TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN ĐIỆN TỬ - VIỄN THÔNG

====o0o====

ĐỒ ÁN

TỐT NGHIỆP ĐẠI HỌC

ĐỀ TÀI

ĐIỀU KHIỂN VÀ GIÁM SÁT NGÔI NHÀ BẰNG GIỌNG NÓI VÀ ỨNG DỤNG ĐIỆN THOẠI

Giảng viên hướng dẫn	**: PGS.TS Nguyễn Quốc Trung**
Sinh viên thực hiện	**: Nguyễn Đình Tâm**
MSSV	**: 20122381**
Lớp	**: KT ĐT-TT 07 K57**

Hà nội, 06/2017

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN ĐIỆN TỬ - VIỄN THÔNG

====O0O====

ĐỒ ÁN

TỐT NGHIỆP ĐẠI HỌC

ĐỀ TÀI

ĐIỀU KHIỂN VÀ GIÁM SÁT NGÔI NHÀ BẰNG GIỌNG NÓI VÀ ỨNG DỤNG ĐIỆN THOẠI

Giảng viên hướng dẫn **: PGS.TS Nguyễn Quốc Trung**

Cán bộ phản biện **:**

Sinh viên thực hiện **: Nguyễn Đình Tâm**

MSSV **: 20122381**

Lớp **: KT ĐT-TT 07 K57**

Hà nội, 06/2017

Đánh giá quyển đồ án tốt nghiệp
(Dùng cho giảng viên hướng dẫn)

Giảng viên đánh giá:.....
Họ và tên Sinh viên:..... MSSV:……
Tên đồ án: ....
……..

Chọn các mức điểm phù hợp cho sinh viên trình bày theo các tiêu chí dưới đây:
Rất kém (1); Kém (2); Đạt (3); Giỏi (4); Xuất sắc (5)

Có sự kết hợp giữa lý thuyết và thực hành (20)						
1	Nêu rõ tính cấp thiết và quan trọng của đề tài, các vấn đề và các giả thuyết (bao gồm mục đích và tính phù hợp) cũng như phạm vi ứng dụng của đồ án					
2	Cập nhật kết quả nghiên cứu gần đây nhất (trong nước/quốc tế)					
3	Nêu rõ và chi tiết phương pháp nghiên cứu/giải quyết vấn đề					
4	Có kết quả mô phỏng/thực nghiệm và trình bày rõ ràng kết quả đạt được					
Có khả năng phân tích và đánh giá kết quả (15)						
5	Kế hoạch làm việc rõ ràng bao gồm mục tiêu và phương pháp thực hiện dựa trên kết quả nghiên cứu lý thuyết một cách có hệ thống					
6	Kết quả được trình bày một cách logic và dễ hiểu, tất cả kết quả đều được phân tích và đánh giá thỏa đáng.					
7	Trong phần kết luận, tác giả chỉ rõ sự khác biệt (nếu có) giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai.					
Kỹ năng viết (10)						
8	Đồ án trình bày đúng mẫu quy định với cấu trúc các chương logic và đẹp mắt (bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến trong đồ án, có căn lề, dấu cách sau dấu chấm, dấu phẩy v.v), có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn đúng quy định					
9	Kỹ năng viết xuất sắc (cấu trúc câu chuẩn, văn phong khoa học, lập luận logic và có cơ sở, từ vựng sử dụng phù hợp v.v.)					
Thành tựu nghiên cứu khoa học (5) (chọn 1 trong 3 trường hợp)						
10a	Có bài báo khoa học được đăng hoặc chấp nhận đăng/đạt giải SVNC khoa học giải 3 cấp Viện trở lên/các giải thưởng khoa học (quốc tế/trong nước) từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế				5	
10b	Được báo cáo tại hội đồng cấp Viện trong hội nghị sinh viên nghiên cứu khoa học nhưng không đạt giải từ giải 3 trở lên/Đạt giải khuyến khích trong các kỳ thi quốc gia và quốc tế khác về chuyên ngành như TI contest.				2	
10c	Không có thành tích về nghiên cứu khoa học				0	
Điểm tổng						**/50**
Điểm tổng quy đổi về thang 10						

3. Nhận xét thêm của Thầy/Cô (*giảng viên hướng dẫn nhận xét về thái độ và tinh thần làm việc của sinh viên****)***

.....

.....

.....

.....

.....

Ngày: / /201

Người nhận xét

Đánh giá quyển đồ án tốt nghiệp
(Dùng cho cán bộ phản biện)

Giảng viên đánh giá:.....
Họ và tên Sinh viên:..... MSSV:……
Tên đồ án: ....
Chọn các mức điểm phù hợp cho sinh viên trình bày theo các tiêu chí dưới đây:
Rất kém (1); Kém (2); Đạt (3); Giỏi (4); Xuất sắc (5)

Có sự kết hợp giữa lý thuyết và thực hành (20)						
1	Nêu rõ tính cấp thiết và quan trọng của đề tài, các vấn đề và các giả thuyết (bao gồm mục đích và tính phù hợp) cũng như phạm vi ứng dụng của đồ án					
2	Cập nhật kết quả nghiên cứu gần đây nhất (trong nước/quốc tế)					
3	Nêu rõ và chi tiết phương pháp nghiên cứu/giải quyết vấn đề					
4	Có kết quả mô phỏng/thực nghiệm và trình bày rõ ràng kết quả đạt được					
Có khả năng phân tích và đánh giá kết quả (15)						
5	Kế hoạch làm việc rõ ràng bao gồm mục tiêu và phương pháp thực hiện dựa trên kết quả nghiên cứu lý thuyết một cách có hệ thống					
6	Kết quả được trình bày một cách logic và dễ hiểu, tất cả kết quả đều được phân tích và đánh giá thỏa đáng.					
7	Trong phần kết luận, tác giả chỉ rõ sự khác biệt (nếu có) giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai.					
Kỹ năng viết (10)						
8	Đồ án trình bày đúng mẫu quy định với cấu trúc các chương logic và đẹp mắt (bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến trong đồ án, có căn lề, dấu cách sau dấu chấm, dấu phẩy v.v), có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn đúng quy định					
9	Kỹ năng viết xuất sắc (cấu trúc câu chuẩn, văn phong khoa học, lập luận logic và có cơ sở, từ vựng sử dụng phù hợp v.v.)					
Thành tựu nghiên cứu khoa học (5) (chọn 1 trong 3 trường hợp)						
10a	Có bài báo khoa học được đăng hoặc chấp nhận đăng/đạt giải SVNC khoa học giải 3 cấp Viện trở lên/các giải thưởng khoa học (quốc tế/trong nước) từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế	5				
10b	Được báo cáo tại hội đồng cấp Viện trong hội nghị sinh viên nghiên cứu khoa học nhưng không đạt giải từ giải 3 trở lên/Đạt giải khuyến khích trong các kỳ thi quốc gia và quốc tế khác về chuyên ngành như TI contest.	2				
10c	Không có thành tích về nghiên cứu khoa học	0				
Điểm tổng						**/50**
Điểm tổng quy đổi về thang 10						

3. Nhận xét thêm của Thầy/Cô

.....

.....

.....

.....

.....

.....

Ngày: / /201

Người nhận xét

(Ký và ghi rõ họ tên)

LỜI NÓI ĐẦU

Ngày nay trên thế giới với sự bùng nổ của các ngành công nghệ thông tin, điện tử ứng dụng v.v đã làm cho đời sống của con người ngày càng hoàn thiện. Các thiết bị tự động hóa đã ngày càng trở nên phổ biến và thậm chí là trong đời sống sinh hoạt hằng ngày của môi con người. Do đó một ngôi nhà thông minh có thể trở thành điều hiện thực hóa để nâng cao và phục vụ các tiện ích cho con người hơn.

Qua báo chí và các phương tiện truyền thông, internet chúng ta có thể thấy những mô hình ngôi nhà thông minh đã ra đời. Là một sinh viên Điện tử - Viễn Thông của trường ĐH Bách Khoa Hà Nội, với những kiến thức đã học cùng với mong muốn thiết kế một ngôi nhà được điều khiển giám sát một cách dễ dàng để đáp ứng được nhu cầu sinh hoạt hằng ngày, em đã chọn "***Điều khiển và giám sát ngôi nhà thông minh bằng giọng nói và ứng dụng điện thoại***" làm đề tài tốt nghiệp của mình.

Em xin gửi lời cảm ơn tới PGS. TS Nguyễn Quốc Trung đã nhiệt tình giúp đỡ chỉ bảo em trong quá trình thực hiện đề tài.

Trong quá trình thực hiện báo cáo, em đã cố gắng hết sức để hoàn thiện một cách tốt nhất. Nhưng với kiến thức và sự hiểu biết có hạn nên không tránh khỏi những thiếu sót cũng như chưa thực sự hoàn thiện đề tài, vì vậy mong thầy cô đóng góp ý kiến cho đề tài của em có thể làm tốt hơn.

Em xin chân thành cảm ơn!

Sinh viên thực hiện

Nguyễn Đình Tâm

TÓM TẮT ĐỒ ÁN

Các nhiệm vụ chính đề ra trong báo cáo đồ án tốt nghiệp:

- Nghiên cứu tổng quan và mô tả bài toán.
- Phân tích yêu cầu đề tài
- Thiết kế tổng quan hệ thống.
- Thiết kế chi tiết hệ thống
- Triển khai và kiểm thử

Bố cục báo cáo: Bao gồm phần Mở đầu, Nội dung, Kết luận, Tài liệu tham khảo.

Phần mở đầu: Giới thiệu tóm tắt nhiệm vụ, đề tài, mục tiêu và phạm vi thực hiện đồ án

Phần nội dung: gồm 5 phần chính

- Chương 1*: Giới thiệu tổng quan đề tài*: Chương này mô tả bài toán thiết kế hệ thống điều khiển nhà thông minh bằng giọng nói và ứng dụng điện thoại
- Chương 2: *Phân tích yêu cầu*: Chương này trình bày kết quả đặc tả chức năng của hệ thống
- Chương 3: *Thiết kế tổng quan hệ thống*: Chương này trình bày kết quả phân tích và thiết kế cho hệ thống.
- Chương 4: *Thiết kế chi tiết hệ thống:* Chương này trình bày thiết kế chi tiết từng khối và các công nghệ được sử dụng
- Chương 5: *Triển khai và kiểm thử:* Chương này trình bày các kết quả triển khai hệ thống và các kết quả kiểm tra đạt được
- **Phần kết luận**: Kết luận chung của đồ án tốt nghiệp, đánh giá các công việc đã làm được và chưa làm trong khuôn khổ đồ án, những kiến thức tích lũy được trong việc làm đồ án. Nêu định hướng trong tương lai tiếp tục phát triển hệ thống.

Tài liệu tham khảo

ABSTRACT

The missions of the graduation thesis:

- Introduce general project "Monitor and control smart home by voice and mobile application"
- Analyze system requirements.
- Design general system.
- Design detailed system.
- Deploy and testing system

Report layout: Includes Introduction, Content, Conclusion, References.

Introduction: Introduce generally project, object and range of graduation thesis

Content: It is divided into five sections

- Section 1*: Introducing general project*: This section describe project "Monitor and control smart home by voice and mobile application".
- Section 2: *Analyzing system requirements*: This section present funtions and non-funtions requeriments of system.
- Section 3: *Designing general system*: This section present the general design of system.
- Section 4: *Designing detailed system:* This section present detailed design of system and used technologies.
- Section 5: *Deploying and testing :* This section present the result of deploying and testing system

Conclusion: Conclusion of the graduation project, evaluate the process and the knowledge gained from doing graduation thesis, present the future project to develop a complete system.

References

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH SÁCH CÁC TỪ VIẾT TẮT

AVS	Alexa Voice Service	Dịch vụ nhận dạng giọng nói Alexa
ASK	Alexa Skill Kit	Bộ kỹ năng cho Alexa
AWS	Amazon Web Services	Dịch vụ web Amazon
IoT	Internet of Things	Internet của vạn vật
MQTT	Message Queuing Telemetry Transport	Giao thức truyền nhận tin nhắn từ xa
HTTP	The Hypertext Transfer Protocol	Giao thức truyền dữ liệu siêu văn bản
CPU	Central processing unit	Bộ xử lý trung tâm
GPU	Graphics processing unit	Bộ xử lý đồ họa
RAM	Random-access memory	Bộ nhớ truy cập ngẫu nhiên
LAN	Local Area Network	Mạng nội bộ
GPIO	General-purpose input/output	Đầu ra/đầu vào đa năng
NO	Normal Open	Thường mở
NC	Normal Close	Thường đóng

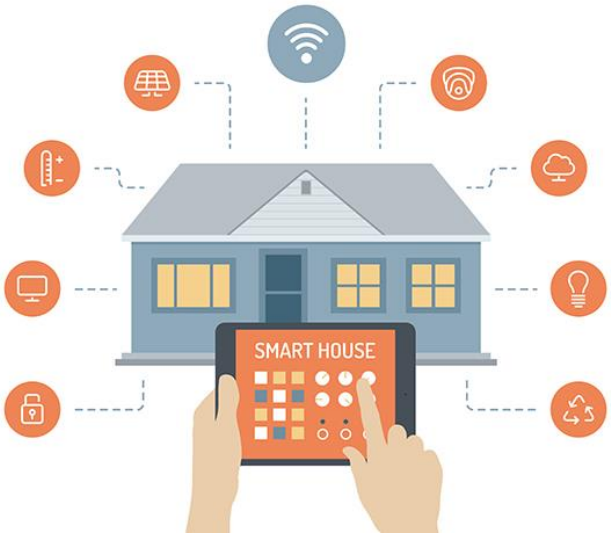
CHƯƠNG 1. GIỚI THIỆU TỔNG QUAN ĐỀ TÀI

1.1 Khái niệm nhà thông minh

Ngôi nhà thông minh là ngôi nhà có các điều kiện kĩ thuật đảm bảo cuộc sống tốt nhất cho con người, được tự động bảo đảm các chỉ tiêu kỹ thuật theo mong muốn của người sử dụng.

Ngôi nhà thông minh là tích hợp của các hệ thống điều khiển và giám sát môi trường như điều khiển đèn chiếu sáng, nhiệt độ, độ ẩm cho phù hợp với môi trường, truyền thông đa phương tiện, an ninh bảo mật….và nhiều tính năng khác nhằm mục đích làm cho cuộc sống ngày càng tiện nghi, an toàn và góp phần sử dụng hợp lý các nguồn tài nguyên như điều khiển bằng giọng nói, điều khiển thông qua ứng dụng di động.

Ngoài ra, cùng với sự phát triển của các thiết bị điện tử cá nhân như máy tính bảng và điện thoại thông minh cùng hạ tầng thông tin ngày càng tiên tiến như internet hoặc các mạng thông tin di động 3G, 4G, ngày nay các hệ thống nhà thông minh còn cung cấp khả năng tương tác với người sử dụng thông qua các thiết bị điện tử cá nhân. Con người có thể điều khiển các thiết bị gia dụng như: hệ thống chiếu sáng, sưởi ấm, máy lạnh, TV, máy tính, âm thanh, camera an ninh,… ở bất cứ đâu, từ trong chính ngôi nhà thông minh đó đến bất kỳ nơi nào trên thế giới thông qua điện thoại hoặc internet.



Hình 1.1. Mô hình ngôi nhà thông minh

1.2 Thực trạng nhà thông minh ở Việt Nam

Trước đây nhà thông minh chỉ hoàn toàn nằm trong trí tưởng tượng hoặc trên phim ảnh. Nhưng từ đầu những năm 1900, "ông tổ" của nhà thông minh – tức các thiết bị điều khiển từ xa đã được bắt đầu nghiên cứu và phát minh, tạo tiền đề cho sự ra đời của chúng sau này. Tuy nhiên cho đến năm 1984, thuật ngữ "Smarthome" - nhà thông minh mới thực sự xuất hiện

Vài năm trở lại đây, khi thế giới đang dần tiến vào kỷ nguyên Internet of Things (IoT), kết nối mọi vật qua Internet, nhà thông minh với khả năng điều khiển trở thành một xu hướng công nghệ tất yếu, là tiêu chuẩn của nhà ở hiện đại. Tại triển lãm lớn nhất về công nghệ điện tử và tiêu dùng diễn ra đầu tháng 1/2015 tại Las Vegas (Mỹ), nhà thông minh là một trong những chủ đề "nóng" nhất. Còn theo hãng tư vấn công nghệ hàng đầu Gartner, công nghệ IoT sẽ bùng nổ kể từ năm 2015 với sự tham gia của hầu hết các hãng công nghệ tên tuổi.

Hình 1.2 Điều khiển ngôi nhà chỉ với một thiết bị thông minh

Việt Nam không nằm ngoài xu hướng này Tại thị trường Việt Nam có sự góp mặt của hàng loạt các thương hiệu như BKAV, Lumi, Hager, Acis, Arkos, Gamma với sự cạnh tranh về giá thành cũng như công nghệ. Ngày càng nhiều khu đô thị áp dụng

giải pháp nhà thông minh trong các căn hộ sang trọng, cao cấp của mình để thỏa mãn nhu cầu của người sử dụng. Hàng loạt khu đô thị cao cấp như Thăng Long Number One, Mandarin Garden, Royal City, Times City, Trung tâm Thương mại Chợ Mơ, Hà Đô Park View, Green Park Tower, Ecopark – khu đô thị sinh thái lớn nhất miền Bắc và hiều biệt thự sang trọng, đẳng cấp từ Bắc vào Nam như Vincom Village, Việt Hưng, Gamuda Gardens, Phú Mỹ Hưng, Phố Đông Village, Thảo Điền, Nam Quan – Quận 7, Tân Phú – Tây Ninh… đang sử dụng giải pháp nhà thông minh.

Hình 1.3 Mô hình nhà thông minh của BKAV

Để có một căn nhà thông minh, ban đầu các công ty sẽ khảo sát thực tế công trình rồi thiết kế phương án theo yêu cầu của gia chủ. Sau khi chốt phương án triển khai, các kỹ sư sẽ tiến hành thi công lắp đặt thiết bị và cấu hình hoạt động cho hệ thống. Vì sử dụng công nghệ truyền thông không dây Zigbee, Wifi... kết nối các thiết bị, nên hệ thống nhà thông minh có thể dễ dàng triển khai với cả những ngôi nhà đang sử dụng.

Tuy nhiên, các giải pháp nhà thông minh hiện tại đa phần mới tập trung ở các công trình biệt thự, chung cư cao cấp. Còn với nhà ở dân dụng, người dùng đã bắt đầu

quan tâm nhưng vẫn đang trong giai đoạn tìm hiểu chứ chưa đầu tư nhiều do còn gặp nhiều khó khăn về thói quen, công nghệ cũng như giá thành còn khá cao đối với những khách hàng bình thường. Ngoài ra cũng chưa tích hợp được giải pháp điều khiển ngôi nhà thông minh bằng giọng nói khi nhu cầu dễ dàng điều khiển ngôi nhà là xu hướng hiện nay.

Từ thực trạng trên, với mong muốn tạo ra một sản phẩm nhà thông minh dễ dàng sử dụng cho người sử dụng và có chi phí giá thành thấp, em đã chọn đề tài "***Điều khiển và giám sát ngôi nhà thông minh bằng giọng nói và ứng dụng điện thoại***" trong phạm vi đồ án tốt nghiệp để tạo bước đệm cho tương lai có thể thiết kế một ngôi nhà thông minh thực sự với nhiều tinh năng hơn.

1.3 Giới thiệu tổng quan về đề tài

Trong phạm vi đề tài đồ án tốt nghiệp, em xin thiết kế mô hình hệ thống nhà thông minh với khả năng điều khiển giám sát bằng giọng nói và qua ứng điện thoại với các đặc tính như sau:

- Phạm vi: Sử dụng cho ngôi nhà với 1 phòng khách, 1 phòng ngủ và 1 phòng bếp.
- Diện tích: Phòng khách: 30m2, phòng ngủ: 16ms, phòng bếp: 14ms. Tổng diện tích 60ms
- Tính năng: Thiết kế ngôi nhà thông minh với các chức năng như:
    - Điều khiển thiết bị qua giọng nói: Đèn chiếu sáng, Tivi, Quạt,
    - Điều khiển thiết bị điện từ máy tính, điện thoại.
    - Điều khiển thiết bị dựa theo cảm biến nhiệt độ, độ ẩm
    - Và một số chức năng khác sẽ phát triển: mở khóa bằng nhận dạng khuôn mặt, phát nhạc…

CHƯƠNG 2. PHÂN TÍCH YÊU CẦU

2.1 Yêu cầu chức năng

2.1.1 ***Điều khiển thiết bị qua giọng nói:***

2.1.1.1 *Đầu vào*

- Lệnh điều khiển của người dùng bằng giọng nói tiếng Anh hoặc tiếng Việt.
- Âm thanh được thu qua mircophone

2.1.1.2 *Xử lý*

Âm thanh được thu qua các microphone và xử lý chuyển đổi thành các tín hiệu điều khiển để bộ xử lý trung tâm truyền tới các thiết bị

2.1.1.3 *Đầu ra*

- Đối với các thiết bị như đèn điện, quạt, bộ xử lý trung tâm sẽ điều khiển bật tắc các thiết bị qua tín hiệu điều khiển tới module Relay.
- Đối với các thiết bị như Tivi, điều hòa, bộ xử lý trung tâm sẽ điều khiển qua tín hiệu tới module hồng ngoại.

2.1.1.4 *Mô tả chức năng*

- Khi người sử dụng nói “Turn off/on the [Living Room]/[Kitchen]/[Bed Room] Light”, “Tắt/bật đèn phòng [khách]/[bếp]/[ngủ]”, “Turn off/on the Fan” thì hệ thống sẽ điều khiển tắt mở đèn, quạt tương ứng.
- Nếu hệ thống thực hiện yêu cầu của người dùng thành công thì sẽ phát âm thanh phản hồi báo thành công hoặc thất bại qua loa.

2.1.2 ***Điều khiển thiết bị bằng máy tính hoặc điện thoại***

2.1.2.1 *Đầu vào*

Thao tác điều khiển của người dùng để điều khiển thiết bị

2.1.2.2 *Xử lý*

Bộ xử lý trung tâm xử lý thao tác điều khiển của người dùng và gửi lệnh điều khiển tới các thiết bị trong ngôi nhà

2.1.2.3 *Đầu ra*

- Tín hiệu điều khiển tới các thiết bị như bóng đèn

- Báo cáo trạng thái hiện tại của các thiết bị lên máy tính và điện thoại

2.1.2.4 *Mô tả chức năng*

Bộ xử lý trung tâm sẽ nhận dữ liệu trạng thái ngôi nhà và hiển thị lên trình duyệt web máy tính hoặc ứng dụng điện thoại, từ đây người dùng có thể thao tác điều khiển các thiết bị một cách dễ dàng bằng cách truy cập qua web server hoặc ứng dụng smartphone để điều chỉnh trạng thái các thiết bị ngay ở trong ngôi nhà của mình

Người dùng có thể kiểm tra các trạng thái nhiệt độ, độ ẩm, trạng thái sử dụng của các thiết bị trong nhà để có thể điều khiển ngôi nhà một cách phù hợp

2.1.3 ***Điều khiển thiết bị dựa theo điều kiện môi trường:***

2.1.3.1 *Đầu vào*

Tín hiệu thu thập từ cảm biến đặt tại các phòng: cảm biến nhiệt độ, độ ẩm...

2.1.3.2 *Xử lý*

- Hệ thống sẽ thu thập dữ liệu môi trường ở trong nhà và lưu trữ
- Truyền thông tin môi trường lên một số thiết bị máy tính và điện thoại của người dùng
- Đặt mức cảnh báo để phát tín hiệu báo động khi có nguy hiểm xảy ra

2.1.3.3 *Đầu ra*

- Tín hiệu điều khiển tới các thiết bị như quạt cho phù hợp với môi trường
- Tín hiệu dữ liệu của môi trường hiện tại để hiển thị lên máy tính và điện thoại
- Tín hiệu cảnh báo nếu có nguy hiểm xảy ra

2.1.3.4 *Mô tả chức năng*

- Hệ thống sẽ lấy dữ liệu môi trường (nhiệt độ, độ ẩm) vào thời điểm hiện tại để hiển thị lên thiết bị của người dùng như máy tính và điện thoại

- Khi nhiệt độ tăng đến một mức giới hạn, hệ thống sẽ phát ra cảnh báo cho người dùng về nguy hiểm có thể xảy ra và có thể sẽ phun nước để dập lửa nếu phát hiện cháy

2.2 Yêu cầu phi chức năng

2.2.1 *Yêu cầu về phần cứng*

- Bộ xử lý trung tâm sử dụng Raspberry Pi 3. Đây là một máy vi tính thu nhỏ với kích thước chỉ bằng một thẻ ATM. Trên bo mạch của Raspberry Pi có CPU, GPU, RAM, khe cắm thẻ microSD, Wi-Fi, Bluetooth, 4 cổng USB 2.0 và 40 chân GPIO để người dùng có thể sử dụng cho các đề tài điện tử.

Hình 2.1. Raspberry Pi 3

- Module đóng ngắt thiết bị sử dụng module Relay

Hình 2.2. Module Relay

2.2.2 *Yêu cầu về hệ thống*

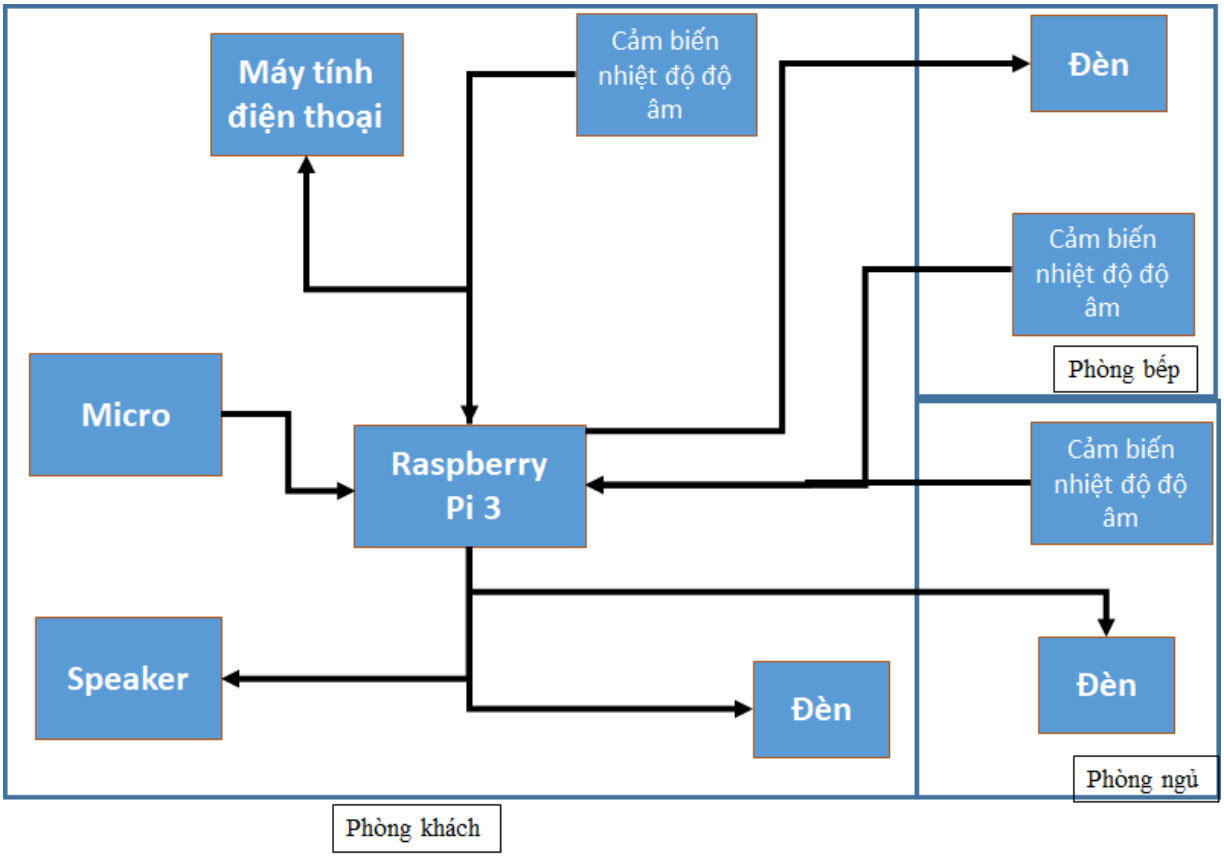
- Đảm bảo điều khiển và xử lý được cho các thiết bị ở trong phòng khách, phòng ngủ và phòng bếp
- Thời gian cập nhật trạng thái môi trường (nhiệt độ, độ ẩm) là 10 phút một lần
- Thời gian xử lý nhận dạng giọng nói: < 6s
- Độ chính xác trong việc điều khiển bằng giọng nói: > 80%
- Nguồn cấp: từ 220V cho bộ xử lý trung tâm và các thiết bị điện
- Chế độ hoạt động: 24/24h
- Thời gian hoàn thành: 3 tháng
- Giá thành: 3.000.000/1 sản phẩm

CHƯƠNG 3. THIẾT KẾ HỆ THỐNG

3.1 Kiến trúc hệ thống

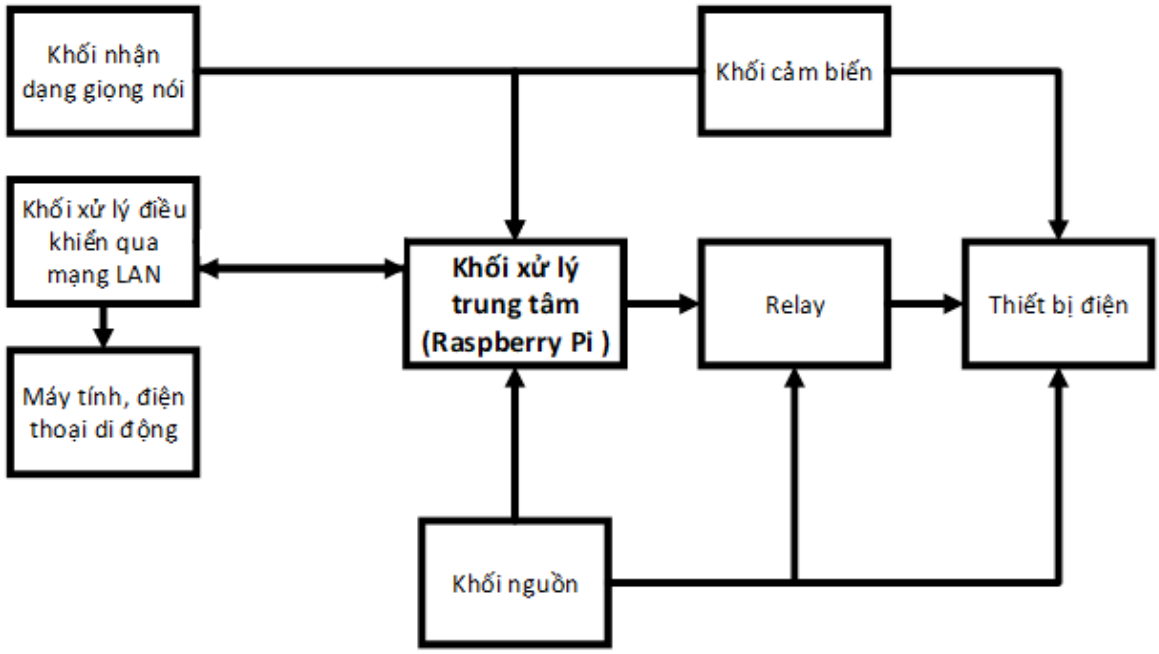
3.1.1 *Mô hình hệ thống nhà thông minh*

Hệ thống được thiết kế cho ngôi nhà với 3 khu vực: Phòng khách, phòng ngủ và phòng bếp. Bộ xử lý trung tâm sẽ được đặt ở phòng khách, nhận thông tin âm thanh từ micro, từ máy tính hoặc điện thoại của người dùng để điều khiển các thiết bị bóng đèn của các phòng sau đó phát phản hồi qua loa. Các cảm biến nhiệt độ độ ẩmm có chức năng thu thập dữ liệu cảm biến và gửi về Raspberry Pi 3, sau đó Raspberry Pi 3 cập nhật dữ liệu lên máy tính điện thoại để thông báo cho người dùng



Hình 3.1 Kiến trúc hệ thống

3.1.2 *Sơ đồ khối*



Hình 3.2. Sơ đồ khối hệ thống

Hệ thống được diễn giải như Hình 3.2 gồm khối điều khiển trung tâm sử dụng board mạch Raspberry Pi 3, được kết nối với các khối cảm biến, khối nhận dạng giọng nói, khối xử lý điều khiển qua mạng Internet, khối hiển thị. Các điều khiển, thông tin cảnh báo, hiển thị thông tin được xử lý song song giữa khối xử lý điều khiển qua mạng Internet và khối xử lý trung tâm

3.2 Mô tả chức năng các khối

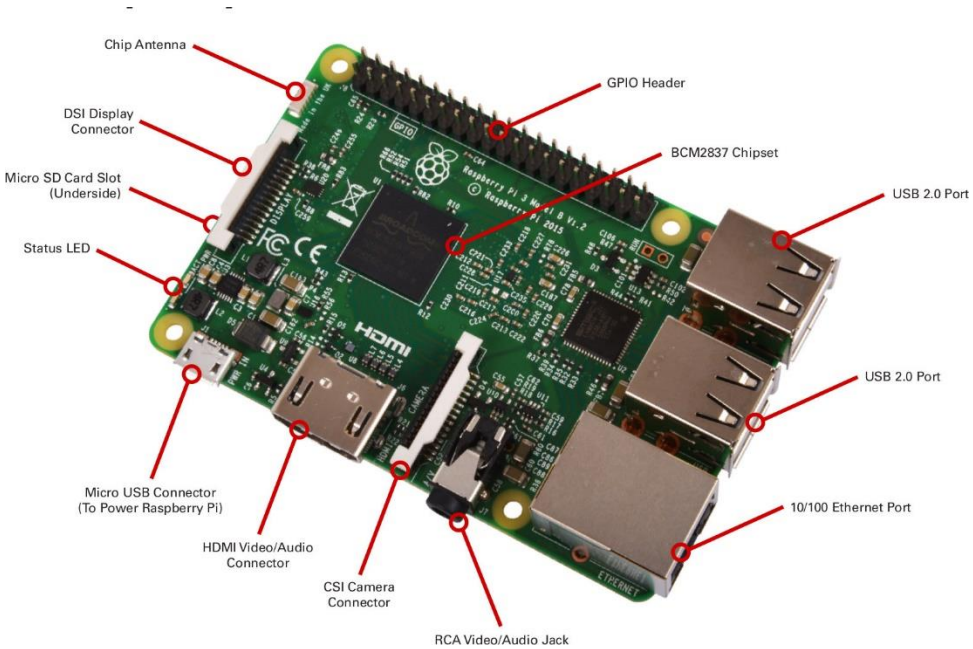
3.2.1 *Khối xử lý trung tâm*

Sử dụng Board Raspberry Pi 3, có khả năng sử dụng như một máy tính chạy hệ điều hành Linux và có khả năng xuất tín hiện ra 40 chân GPIO (General-purpose input/output) để có thể giao tiếp và điều khiển vô số board mạch và ngoại vi bên ngoài. Raspberry Pi 3 xây dựng quanh bộ xử lý SoC Broadcom BCM2835 bao gồm CPU, GPU, bộ xử lý âm thanh/video và các tính năng khác….

Raspberry Pi 3 cho phép lập trình điều khiển cổng vào, ra từ đó kết nối được với thiết bị bên ngoài để điều khiển hoặc giải quyết 1 bài toán thực tế. Các ngôn ngữ được sử dụng trong lập trình Raspberry Pi cho đến nay gồm Python, C, C++, Java, Scratch, Ruby, JavaScript, Html5, Perl, Elang. Phổ biến nhất và nhanh nhất là ngôn ngữ lập trình Python.

Raspberry Pi 3 có thông tin kĩ thuật gồm:

- SoC BCM2837 1.2 GHz với 1 GB RAM .
- 1 cổng HDMI cho đầu ra âm thanh / video số .
- 1 cổng video RCA cho đầu ra video Analog .
- Jack Headphone Stereo 3.5mm cho đầu ra âm thanh Analog .
- 4 cổng USB .
- 01 đầu đọc thẻ nhớ SD để tải hệ điều hành .
- 01 cổng Ethernet LAN.
- 40 chân GPIO
- Tích hợp chuẩn Wifi 802.11n cùng Bluetooth 4.1



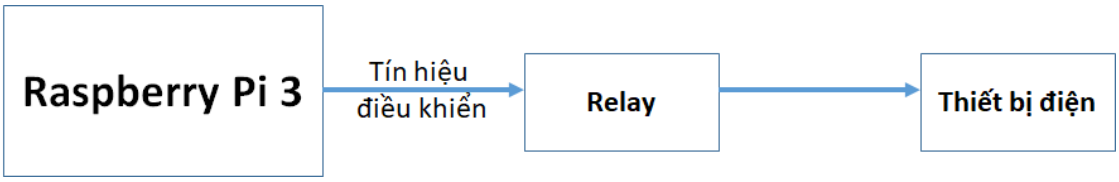
Hình 3.3. Các cổng giao tiếp của Board Raspberry Pi 3

3.2.2 *Khối nguồn*

Đây là khối cung cấp nguồn cho các thiết bị trong nhà cũng như hệ thống để các khối có thể hoạt động:

- Đối với các thiết bị điện, khối xử lý trung tâm (Raspberry pi 3): Sử dụng nguồn xoay chiều 220V
- Đối với các khối cảm biến, khối relay: Sử dụng nguồn 3.3V và 5V của Raspberry Pi 3

3.2.3 *Khối Relay*



Hình 3.4 Sơ đồ khối relay

Khối relay có chức năng trung gian làm công tắc điện tử để truyền tín hiệu điều khiển từ Raspberry Pi 3 tới thiết bị điện để tắt mở theo ý người dùng.

3.2.4 *Khối cảm biến*

Hình 3.5. Sơ đồ khối Cảm biến

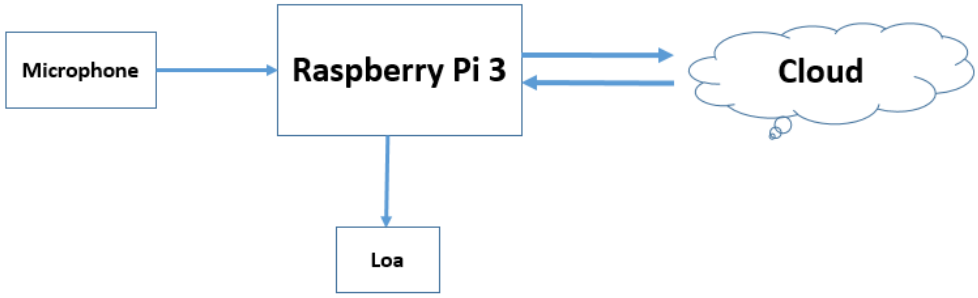
Khối chức năng đo nhiệt độ, truyền thông tin tới bộ xử lý trung tâm để hiển thị cho người dùng thông qua ứng dụng điện thoại hoặc tương tác qua giọng nói và xử lý báo động khi vượt ngưỡng nhiệt độ (cảnh báo cháy)

3.2.5 *Khối xử lý giọng nói*

Nhận thông tin giọng nói qua micro kết nối với Board Raspberry pi 3. Board Raspberry pi 3 sử dụng Google Voice API để chuyển giọng nói tiếng Việt thành dữ liệu văn bản, sau đó so sánh với các câu có sẵn để thực hiện thao tác xử lý tương ứng với lệnh. Board Raspberry pi 3 cũng lưu sẵn các câu thông báo, cảnh báo để phát âm thanh qua Loa

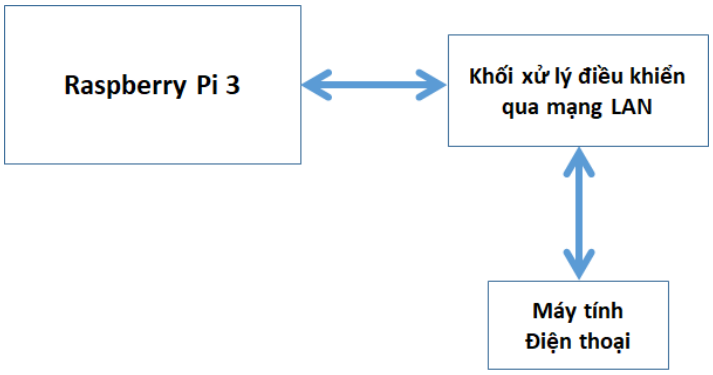
Bên cạnh đó chúng ta có thể sử dụng dịch vụ giọng nói Alexa (Alexa Voice Service - AVS) để có thể điều khiển được ngôi nhà thông minh. AVS cho phép chúng ta cấu hình Raspberry Pi 3 như một thiết bị thông minh đáp ứng được nhu cầu trao đổi, điều khiển thông minh với ngôi nhà.

Bộ xử lý trung tâm sẽ nhận yêu cầu từ microphone, gửi yêu cầu lên Cloud và nhận kết quả trả về để xử lý điều khiển và thông báo kết quả xử lý qua loa.



Hình 3.6. Khối nhận dạng giọng nói

3.2.6 ***Khối xử lý điều khiển qua mạng LAN***



Hình 3.7 Khối xử lý điều khiển qua mạng LAN

Board Raspberry Pi 3 được kết nối với mạng LAN qua cổng Ethernet LAN để truyền dữ liệu trạng thái ngôi nhà, từ đó người dùng có thể sử dụng máy tính, điện thoại để có thể truy cập xem thông tin cũng như thao tác điều khiển các thiết bị của ngôi nhà

Trong phạm vi đề tài này, em sử dụng openHAB, đây là phần mềm miễn phí mã nguồn mở có chức năng là bộ điều khiển trung tâm với khả năng giao tiếp với rất nhiều loại thiết bị khác nhau trong hệ thống ngôi nhà qua giao diện Website hoặc ứng dụng di động kết nối qua mạng LAN

CHƯƠNG 4. THIẾT KẾ CHI TIẾT

4.1 Khối xử lý trung tâm

4.1.1 *Giới thiệu chung.*

Raspberry Pi là một chiếc máy tính tí hon chạy hệ điều hành Linux ra mắt vào tháng 2 năm 2012 với giá chỉ $25. Ban đầu Raspberry Pi được phát triển dựa trên ý tưởng tiến sĩ Eben Upton tại đại học Cambridge muốn tạo ra một chiếc máy tính giá rẻ để học sinh có thể dễ dàng tiếp cận và khám phá thế giới tin học. Với những ưu điểm nổi bật, hơn một triệu board Raspberry Pi đã được bán ra chỉ trong vòng chưa đầy một năm. Hiện nay Raspberry Pi đã được phát triển tới phiên bản thứ 3 với giá chỉ 35$.

Chỉ cần 1 bàn phím, 1 tivi hoặc 1 màn hình có cổng HDMI/DVI, 1 nguồn USB 5V và 1 dây micro USB là đã có thể sử dụng Raspberry Pi như 1 máy tính bình thường. Với Raspberry Pi, ta có thể sử dụng các ứng dụng văn phòng, nghe nhạc, xem phim độ nét cao... Một điều quan trọng là nó rất tiết kiệm điện và khả năng chạy liên tục 24/24.

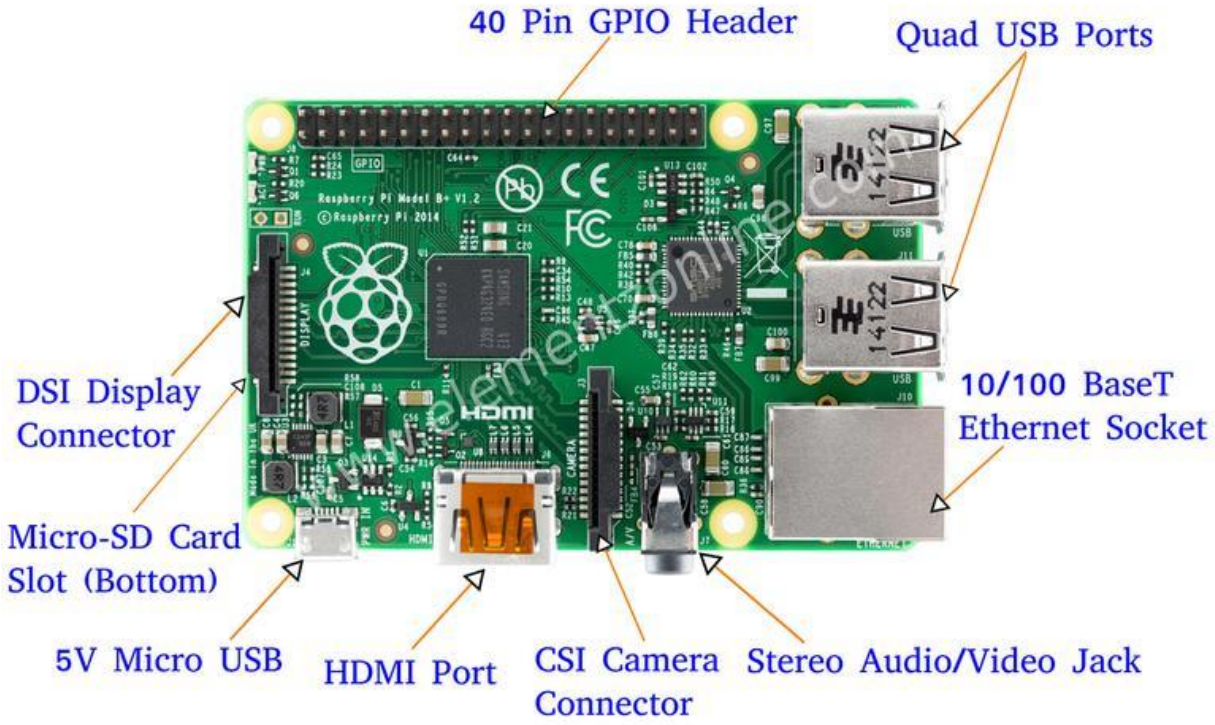
4.1.2 *Phần cứng*

4.1.2.1 *Thông số kỹ thuật*

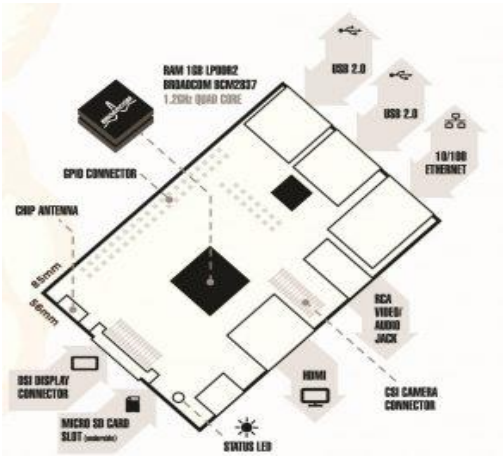
Thông số	**Model B**
System-on-Chip (SoC)	Broadcom BCM2837
CPU	1.2GHz 64-bit quad-core ARMv8 CPU
GPU	Broadcom VideoCore IV
Bộ nhớ (SDRAM)	1GB LPDDR2
USB 2.0 Ports	4× USB 2.0
Video Outputs	Composite RCA hoặc HDMI
Audio Outputs	3.5 mm jack hoặc HDMI
Audio Inputs	Có thể sử dụng microphone USB
Onboard Storage	Secure Digital|SD / MMC / SDIO card slot
Onboard Netword	10/100 Ethernet, 2.4GHz 802.11n wireless
Công suất	700 mA (3.5 W)
Nguồn điện	5V DC qua cổng micro USB hoặc GPIO
Kích thước	85.0 x 56.0 x 17.0 mm
GPIO	40 chân

Bảng 4.1 Thông số kỹ thuật của board Raspberry Pi 3 model B

4.1.2.2 *Cấu tạo*



Hình 4.1. Cấu tạo phần cứng Raspberry Pi 3



Hình 4.2. Sơ đồ chân kết nối Raspberry Pi 3

- Trái tim của Raspberry Pi là chip SoC (System-On-Chip) Broadcom BCM2835 chạy ở tốc độ 1.2GHz. Chip này tương đương với nhiều loại được sử dụng trong smartphone phổ thông hiện nay, và có thể chạy được hệ điều hành Linux. Tích

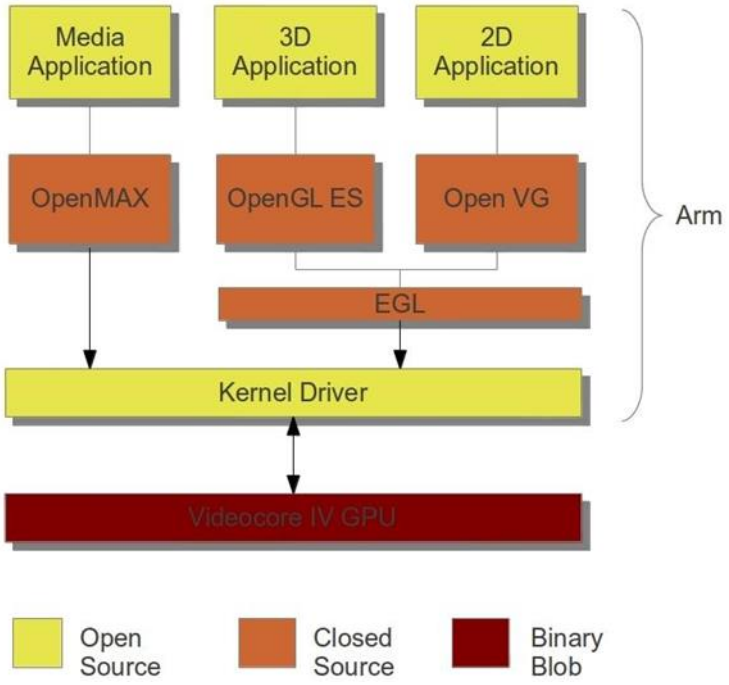
hợp trên chip này là nhân đồ họa (GPU) Broadcom VideoCore IV. GPU này đủ mạnh để có thể chơi 1 số game phổ thông và phát video chuẩn full HD.

- Hệ thống GPIO (General Purpose Input Output): gồm 40 chân chia làm hai hang. Từ đây ta có thể kết nối và điều khiển rất nhiều thiết bị điện tử/cơ khí khác như xuất tín hiệu ra led, thiết bị… hoặc đọc tín hiệu vào từ các nút nhấn, công tắc,cảm biến… Ngoài ra còn có các IO tích hợp các chuẩn truyền dữ liệu UART, I2C và SPI.
- Ngõ HDMI: dùng để kết nối Pi với màn hình máy tính hay tivi có hỗ trợ cổng HDMI.
- Ngõ RCA Video (analog): khi thiết kế Pi người ta cũng tính đến trường hợp người sử dụng ở các nước đang phát triển không có điều kiện sắm một chiếc tivi đời mới tích hợp cổng HDMI. Vì vậy cổng video analog này được thêm vào, giúp Raspberry Pi có thể kết nối với chiếc tivi đời cũ.
- Ngõ audio 3.5mm: kết nối dễ dàng với loa ngoài hay headphone. Đối với tivi có cổng HDMI, ngõ âm thanh được tích hợp theo đường tín hiệu HDMI nên không cần sử dụng ngõ audio này.
- Cổng CSI: khe cắm này là để cắm modul camera vào Raspberry Pi. Khi sản xuất Raspberry Pi thì nhà sản xuất còn sản xuất thêm một modul camera 5MP nhưng người mua không được hỗ trợ mà phải mua thêm. Chúng ta có thể chụp hình, quay phim, ... làm việc tất cả các tác vụ như trên một camera bình thường.
- Cổng DSI: nơi đây sẽ giúp ta có thể kết nối Raspberry Pi với màn hình cảm ứng để hiển thị và sử dụng Raspberry một cách trực quan nhất. Chúng ta có thể thực hiện các tác vụ tương đương như khi sử dụng chuột và bàn phím.
- Cổng USB: một điểm mạnh nữa của Raspberry Pi là tích hợp 2 cổng USB 2.0. Ta có thể kết nối với bàn phím, chuột hay webcam, bộ thu GPS… qua đó có thể mở rộng phạm vi ứng dụng. Vì Raspberry Pi chạy Linux nên hầu hết thiết bị chỉ cần cắm-và-chạy (Plug-and-Play) mà không cần cài driver phức tạp.
- Cổng Ethernet: cho phép kết nối Internet dễ dàng. Cắm dây mạng vào Pi, kết nối với màn hình máy tính hay tivi và bàn phím, chuột là có thể sử dụng dễ dàng.

- Khe cắm thẻ SD: Raspberry Pi không tích hợp ổ cứng. Thay vào đó nó dùng thẻ SD để lưu trữ dữ liệu. Toàn bộ hệ điều hành Linux sẽ hoạt động trên thẻ SD này vì vậy nó cần kích thước thẻ nhớ tối thiểu 4 GB và dung lượng hỗ trợ tối đa là 32 GB.
- Đèn LED: trên Pi có 5 đèn LED để hiển thị tình trạng hoạt động
  - ACT: Truy cập thẻ SD
  - PWR: Đèn nguồn (Luôn luôn sáng khi có nguồn cắm vào)
  - FDX: Full Duplex Lan
  - LNK: Link/Activity (Khi có hoạt động trao đổi file qua LAN nó sẽ nhấp nháy)
  - 100: Mạng 100Mbps
- Jack nguồn micro USB 5V, tối thiểu 700mA: nhờ thiết kế này mà ta có thể tận dụng hầu hết các sạc điện thoại di động trên thị trường để cấp nguồn điện cho Raspberry Pi.

4.1.3 *Cấu trúc phần mềm*

Các Raspberry Pi sử dụng hệ điều hành dựa trên nền tảng Linux. Phần cứng GPU được truy cập thông qua Image Firmware được nạp vào GPU vào lúc khởi động từ thẻ SD. Image Firmware được gọi là đốm màu nhị phân (Binary Blob), trong khi ARM liên kết với mã trình điều khiển Linux ban đầu được dựa vào nguồn đóng. Một phần của mã điều khiển đã được giải phóng, tuy nhiên nhiều chương trình điều khiển thực tế được thực hiện bằng cách sử dụng mã nguồn đóng GPU. Phần mềm ứng dụng sử dụng các cuộc gọi đến thư viện thời gian chạy nguồn đóng (OpenMax, OpenGL ES hay OpenVG). Nó sẽ gọi một trình điều khiển nguồn mở bên trong lõi Linux, sau đó gọi mã điều khiển nguồn đóng GPU VideoCore IV. Các API của trình điều khiển lõi là cụ thể cho những thư viện đóng. Các ứng dụng Video sử dụng OpenMax, ứng dụng 3D sử dụng OpenGL ES và ứng dụng 2D sử dụng OpenVG và cả hai lần lượt sử dụng EGL. OpenMax và EGL sử dụng trình điều khiển nền tảng mã nguồn mở.



Hình 4.3 Sơ đồ kiến trúc phần mềm

Nhà sản xuất Raspberry sẽ cung cấp một tập hợp các thư viện mã nguồn đóng cho phép chúng ta truy cập vào các tính năng tăng tốc GPU. Các thư viện sẽ có sẵn là:

- OpenGL ES 2.0 (opengl) là một thư viện 3D, rất thường được sử dụng trên máy tính để bàn và các hệ thống nhúng. Nó được định nghĩa bởi Khronos Group.

- OpenVG là một thư viện bản vẽ véc tơ 2D, cũng thường được sử dụng trên máy tính để bàn và các hệ thống nhúng. Một lần nữa, được định nghĩa bởi Khronos Group.

- EGL là một giao diện lập trình ứng dụng giữa Khronos và API như OpenGL ES hay OpenVG và hệ thống cửa sổ nền tảng nguồn gốc cơ bản.

- Openmax cung cấp một tập hợp các API với khái niệm trừu tượng của người dùng cho những thói quen sử dụng trong âm thanh, video, vàxử lý hình ảnh tĩnh. OpenMax định nghĩa ba lớp, đây là lớp IL, cung cấp một giao diện giữa các khuôn khổ đa phương tiện như Gstreamer và một tập hợp các thành phần đa phương tiện (như bảng mã).

4.1.4 ***Hệ điều hành***

4.1.4.1 *Giới thiệu*

Raspberry Pi là một máy tính, để máy tính này hoạt động cần cài đặt hệ điều hành. Trong thế giới nguồn mở linux, có rất nhiều phiên bản hệ điều hành tùy biến (distro) khác nhau. Tùy theo nhu cầu và mục đích, cũng như khả năng học hỏi mà ta sẽ sử dụng distro phù hợp với mình.

Có 5 phiên bản hệ điều hành được cung cấp chính thức cho Raspberry Pi:

- Raspian "wheezy": đây là distro dựa trên Debian wheezy, sử dụng hard-float ABI (tính toán dấu chấm động bằng phần cứng) cho thời gian chạy các ứng dụng nhanh hơn. Có sẵn giao diện đồ họa. Phù hợp với người mới bắt đầu tiếp cận Linux vì tính dễ sử dụng và trực quan.
- Soft-float "wheezy": vẫn được xây dựng dựa trên Debian wheezy nhưng việc xử lý dấu chấm động được thực hiện bằng phần mềm. Việc này giúp có thể sử dụng máy ảo Java (Oracle JVM) trên Raspberry.
- Arch Linux: phiên bản giành cho ARM. Đảm bảo thời gian khởi động trong vòng 10 giây. Chỉ khởi động và load các gói cần thiết. Để sử dụng được Arch Linux cần có kiến thức cơ bản về Linux.
- Pidora: là phiên bản của Fedora được tối ưu cho Raspberry Pi, có sẵn giao diện đồ họa. Giành cho những ai đã quen xài Fedora.
- RISC OS: là hệ điều hành do nhóm phát triển ARM thiết kế riêng. Đây không phải là một phiên bản Linux, do vậy cần làm quen với cấu trúc và câu lệnh đặc trưng cho hệ điều hành này.

Ngoài ra còn nhiều hệ điều hành khác ta có thể cài đặt như: Raspbmc, Android... Trong đề tài này em sử dụng hệ điều hành Raspian, hệ điều hành chính thức của Raspberry Pi, được xây dựng trên nền tảng Debian. Đây cũng là hệ điều hành phổ biến nhất, nhiều người dùng nhất và được hỗ trợ nhiều nhất của Raspberry Pi

Có 2 cách để cài đặt Raspbian.

4.1.4.2 *Sử dụng gói NOOBS*

Đây là cách đơn giản dành cho người mới bắt đầu.

- Bước 1: Download gói NOOBS:

Đây là gói cài đặt đã có sẵn Raspbian được cung cấp bởi nhà phát hành Raspberry Pi.

- Bước 2: Format thẻ SD:

Nếu thẻ SD của bạn mới tinh thì không nhất thiết phải làm bước này.

Phần mềm SD Formatter 4.0 được khuyên dùng cho việc Format thẻ.

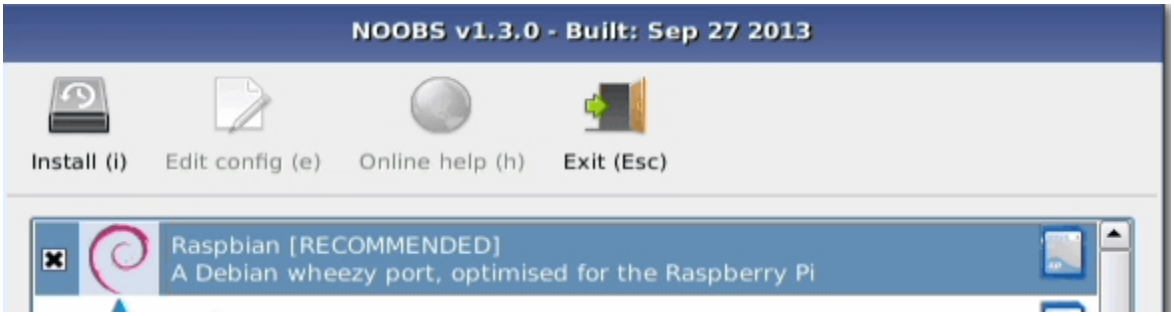
- Bước 3: Giải nén gói NOOBS vào thẻ SD:

Sau khi việc download gói NOOBS và format thẻ hoàn tất.

Bạn chỉ cần giải nén gói NOOBS vào thẻ SD.

- Bước 4: Gắn thẻ vào Raspberry Pi và khởi động:

Hình 4.4 là lần khởi động đầu tiên của Raspbian



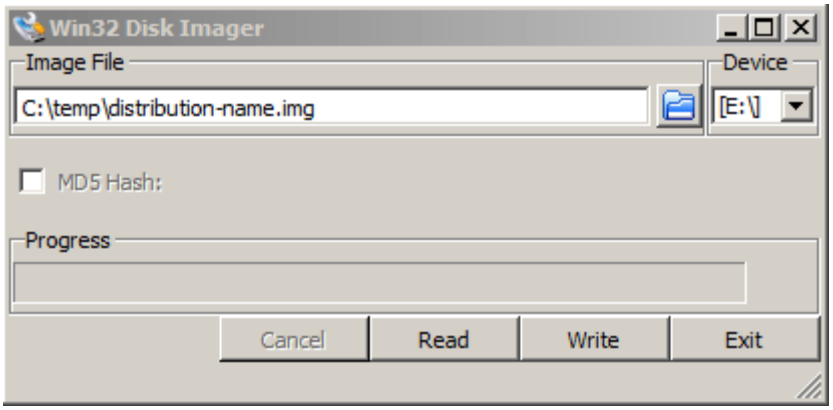
Hình 4.4 Giao diện khi khởi động Raspbian lần đầu

Chúng ta cần chọn Raspbian và nhấn vào biểu tượng Install và đợi việc cài đặt hoàn thành.

4.1.4.3 *Cài đặt Image của Raspbian*

- Bước 1: Cắm thẻ SD vào máy tính và để ý tên ổ đĩa của thẻ (chẳng hạn: E:\)
- Bước 2: Download file image của Raspbian và giải nén (thành file đuôi *.img*).
- Bước 3: Download Win32DiskImager và giải nén.
- Bước 4: Chạy Win32DiskImager (có thể bạn cần Run as administrator).

- Bước 5: Chọn đường dẫn đến file *.img* đã giải nén và chọn ổ đĩa của thẻ SD, tương tự hình 4.5:



Hình 4.5. Chọn đường dẫn tới ổ SD

- Bước 6: Nhấn *Write* và đợi quá trình ghi hoàn thành.
- Bước 7: Thoát chương trình và gỡ thẻ SD khỏi máy tính.
- Bước 8: Gắn thẻ SD vào Raspberry Pi và khởi động.

4.1.4.4 *Lệnh Linux cơ bản để sử dụng Raspberry Pi*

Dòng lệnh là một thế mạnh của HĐH Linux, vì nếu chúng ta thành thạo lệnh Linux, sử dụng lệnh giúp hệ thống chay nhanh, mạnh mẽ hơn nhiều so với sử dụng giao diện đồ họa, đồng thời, thể hiện sự chuyên nghiệp của người dùng

Lệnh	**Ý nghĩa và ví dụ**
man - manual guide	Hiển thị thông tin hướng dẫn sử dụng lệnh trên Linux. Ví dụ: Để xem cách sử dụng lệnh shutdown, ta gõ lệnh man shutdown
startx	Khởi chạy giao diện đồ họa từ giao diện dòng lệnh.
ls - listing	Liệt kê tất cả tập tin và thư mục tại vị trí hiện hành. Ví dụ: Trong thư mục hiện tại là /opt có 2 tập tin excel, để liệt 2 tập tin này lên giao diện dòng lệnh, gõ lệnh ls - la
cd - change directory	Chuyển đến vị trí thư mục khác. Ví dụ: Đang ở vị trí root / muốn chuyển sang thư mục /opt, gõ lệnh cd /opt
mkdir - make directory	Tạo thư mục, giống **New Folder** trong Windows Ví dụ: Muốn tạo một thư mục mới tên hoang, gõ lệnh mkdir hoang
rmdir - remove directory	Xóa một thư mục. Ví dụ: Muốn xóa thư mục /opt/hoang đã tạo, gõ lệnh

	rmdir hoang * Lưu ý, lệnh rmdir chỉ xóa được thư mục trống, với các thư mục có chứa thư mục con hoặc tập tin, để xóa được thư mục, gõ lệnh rm - rf tenthumuc
mv - move	Di chuyển hoặc đổi tên thư mục hay tập tin. Ví dụ: Để di chuyển tập tin test.txt từ thư mục /opt sang thư mục /user/hoang, gõ lệnh move /opt/test.txt /user/hoang/test.txt Nếu muốn vừa di chuyển vừa đổi tên thư mục, gõ lệnh move /opt/test.txt /user/hoang/ten_moi.txt
rm - remove	Xóa một tập tin. Ví dụ: Để xóa tập tin test.txt trong thư mục /user/hoang/ gõ lệnh rm /user/hoang/test.txt
pwd - print working directory	In ra màn hình đường dẫn vị trí hiện tại mà người dùng đang ở đó.
gedit	Mở một tập tin với trình soạn thảo gedit hoặc tạo mới tập tin văn bản. Ví dụ: Muốn chỉnh sửa file test.txt trong thư mục /user/hoang/ gõ lệnh gedit /user/hoang/test.txt
sudo shutdown -h now	Tắt hệ điều hành ngay lập tức, không quan tâm đến các tiến trình đang hoạt động. Với tính bảo mật của Linux, một số lệnh phải có quyền root mới có thể thực thi. Trong **Raspbian**, để thực thi với quyền root, sử dụng lệnh sudo trước đầu mỗi dòng lệnh, sau khi enter, hệ thống sẽ yêu cầu mật khẩu. Ví dụ **sudo reboot**. Sau khi gõ lệnh sudo một lần, trong khoản thời gian 15 phút, có thể sử dụng các lệnh với quyền root mà không cần sử dụng lệnh sudo nữa.
sudo reboot	Khởi động lại hệ điều hành, giống restart trong **Windows**
sudo apt-get install	Cài đặt một gói ứng dụng từ internet. Hệ thống sẽ tự tìm đến các kho chứa gần nhất để tải và cài đặt ứng dụng. Ví dụ: Để cài đặt trình soạn thảo văn bản vim cho hệ điều hành, gõ lệnh sudo apt-get install vim
sudo apt-get remove	Gỡ bỏ một gói ứng dụng. Ví dụ: Để gỡ bỏ trình soạn thảo văn bản vim đã cài đặt, gõ lệnh sudo apt-get remove vim
sudo apt-get purge	Gỡ bỏ hoàn toàn gói ứng dụng, kể cả các cấu hình và nhật

	ký (log) trên hệ điều hành. Ví dụ: Để gỡ bỏ hoàn toàn trình soạn thảo văn bản vim, gồm cả setting và log, gõ lệnh sudo apt-get remove vim
sudo apt-get update	Cập nhật các gói ứng dụng mới nhất cho hệ điều hành.
sudo apt-get upgrade	Nâng cấp các gói ứng dụng lên phiên bản mới nhất.
sudo apt-get autoremove	Quét và xóa những gói ứng dụng không dùng đến.

Bảng 4.2. Các lệnh Linux thông dụng

4.1.5 ***Ngôn ngữ lập trình Python***

4.1.5.1 *Giới thiệu*

Python là ngôn ngữ lập trình hướng đối tượng, bậc cao, mạnh mẽ. Ngoài ra, học Python là khá đơn giản và dễ dàng. Python cũng là một ngôn ngữ thông dịch, tức là ngôn ngữ không cần phải biên dịch ra file chạy mà đọc code đến đâu thì chạy đến đó. Khi chạy lệnh Python ta sẽ có một giao diện dòng lệnh giống của Unix, có thể chạy từng dòng code ngay trực tiếp tại đây.

Guido Van Rossum là người sáng lập ra ngôn ngữ này. Source code của Python mã nguồn mở, do tổ chức phi lợi nhuận Python Software Foundation quản lý. Theo đánh giá của Eris S. Raymond, Python là ngôn ngữ có hình thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện chon người mới học lập trình.

4.1.5.2 *Ưu và nhược điểm*

❖ Ưu điểm:

- Đơn giản : Cú pháp đơn giản giúp cho lập trình dễ dàng đọc và tìm hiểu.
- Tốc độ : Python có tốc độ xử lý nhanh, Python là một ngôn ngữ cho phép việc thông dịch (interprete) ngay trong lúc chạy cho nên mỗi dòng lệnh đều được thông dịch lại. Và trên hết, Python là ngôn ngữ tối giản, đơn giản cho người dùng nên để đáp ứng được điều đó, nó phải đi kèm theo bộ thư viện cồng kềnh để có thể diễn đạt được hết ý người viết sang ngôn ngữ máy.
- Tương tác: Chế độ tương tác cho phép người lập trình thử nghiệm tương tác sửa lỗi của các đoạn mã.
- Chất lượng: Thư viện có tiêu chuẩn cao, Python có khối cơ sở dữ liệu khá lớn nhằm cung cấp giao diện cho tất cả các CSDL thương mại lớn.

- Thuận tiện: Python được biên dịch và chạy trên tất cả các nền tảng lớn hiện nay. Trong đó chúng ta có thể sử dụng cho Raspberry Pi 3
- Mở rộng: Với tính năng này, Python cho phép người lập trình có thể thêm hoặc tùy chỉnh các công cụ nhằm tối đa hiệu quả có thể đạt được trong công việc.

❖ Nhược điểm:

- Python không có các thuộc tính hướng đối tượng như: protected,private hay public, không có vòng lặp do…while và switch….case.
- Python mặc dù nhanh hơn so với PHP, nhưng lại không nhanh hơn so với C++, Java.

4.1.5.3 *Python cho Raspberry Pi 3*

Python được chọn là ngôn ngữ chính thức để lập trình cho Raspberry Pi 3. Với Python chúng ta có thể dễ dàng điều khiển các chân GPIO của Raspberyr Pi 3 cũng như kết nối với các nền tảng khác được sử dụng trong đề tài này.

Để điều khiển được các chân GPIO chúng ta sử dụng thư viện RPi.GPIO, thư viện này đã được cài đặt sẵn cho hệ điều hành Raspbian của Raspberry Pi.

40 chân GPIO bao gồm:

- 26 chân GPIO. Khi thiết lập là chân output, GPIO có thể nhận dữ liệu từ các thiết bị ngoài, khi thiết lập là input, GPIO có thể được sử dụng như chân ngắt, GPIO 14 và 15 được thiết lập sẵn là chân input.
- 1 chân UART, 1 chân I2C, 2 chân SPI, 1 chân PWM (GPIO 4)
- 2 chân nguồn 5V, 2 chân nguồn 3.3V, 8 chân GND
- 2 chân ID EEPROM

Ngoài ra trong đề tài em có sử dụng một số thư viện khác như json, paho.mqtt, httplib, alsaaudio, wave, numpy… để lập trình các chức năng của đề tài.

4.2 Khối Relay

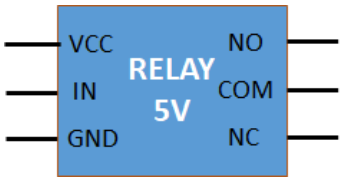
4.2.1 *Giới thiệu chung*

Khối này có chức năng làm công tắc điện tử để điều khiển bật tắt thiết bị điện. Khối được nối giữa khối xử lý trung tâm và các bóng đèn với hai trạng thái đóng mở.

Hiện nay có hai loại module relay: *module relay đóng ở mức thấ*p (nối cực âm vào chân tín hiệu relay sẽ đóng), *module relay đóng ở mức cao* (nối điện áp dương vào chân tín relay sẽ đóng). Trong đề tài này em sử dụng *module relay đóng ở mức cao 5V DC*

4.2.2 *Thông số kĩ thuật*

Module relay đóng ở mức cao có 6 chân. Trong đó 3 chân dùng để cấp nguồn và kích hoạt relay, 3 chân còn lại nối với thiết bị điện công suất cao.

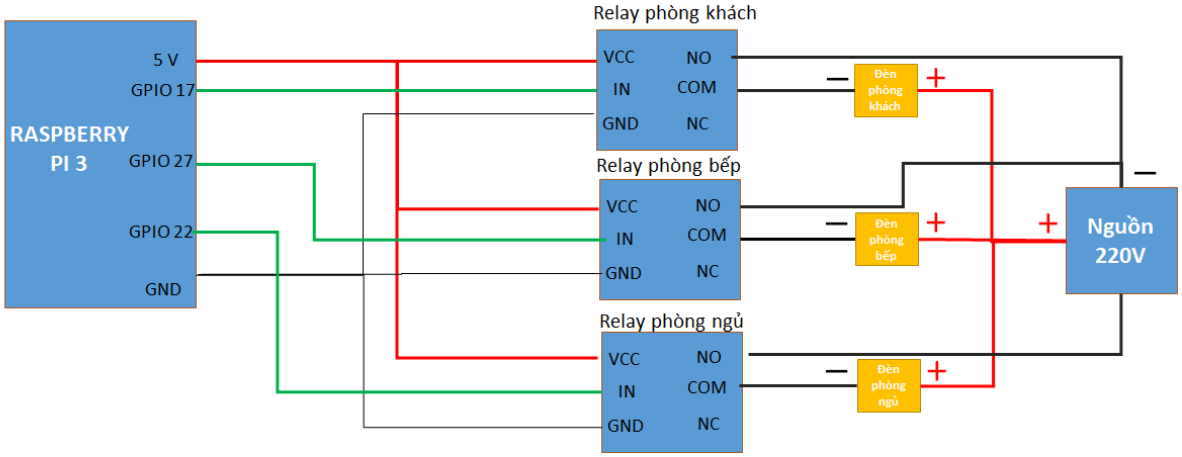


Hình 4.6. Các chân Relay 5V

- Chân VCC cấp nguồn 5V từ Raspberry Pi 3
- Chân GND nối với chân GND của Raspberry Pi 3
- Chân IN là chân để kích hoạt, khi chưa có điện áp đặt vào chân IN (V_{IN} = 0) thì relay sẽ mở, dòng điện không đi qua được. Khi có điện áp lớn hơn 0 (V_{IN} >0) đặt vào chân IN, relay sẽ đóng mạch, dòng điện sẽ chạy qua được giúp chúng ta có thể bật được thiết bị
- Chân NO (Normal Open) chân thường mở, được dùng khi muốn có dòng điện khi relay ở trạng thái OFF.
- Chân NC (Normal Close) chân thường đóng, được dùng khi muốn có dòng điện khi relay ở trạng thái ON.
- Chân COM là chân chung, nó luôn được kết nối với 1 trong 2 chân còn lại. Còn việc nó kết nối chung với chân nào thì phụ thuộc vào trạng thái hoạt động của rơ le.

4.2.3 *Kết nối với Raspberry Pi*

Relay được cấp nguồn 5V từ Raspberry Pi 3, 3 chân IN của các Relay phòng khách, phòng bếp và phòng ngủ được nối lần lượt với các chân GPIO 17, GPIO 27, GPIO 22 để điều khiển bật tắt. Ba chân còn lại của relay, chân NO được nối vào chân mát của nguồn xoay chiều 220V, chân COM được nối với cực âm của đèn. Cực dương của đèn được nối với chân nóng nguồn điện.



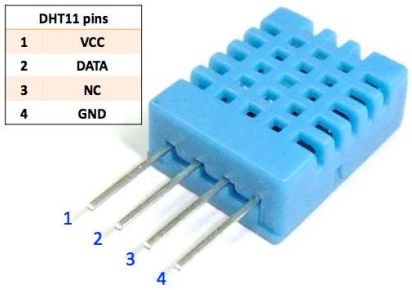
Hình 4.7. Sơ đồ kết nối relay với Raspberry pi 3 và thiết bị điện

4.3 Khối cảm biến

Trong đề tài này em sử dụng cảm biến DHT11 cho 3 phòng khách, phòng ngủ, phòng bếp để giám sát nhiệt độ, độ ẩm của căn nhà

4.3.1 *Giới thiệu*

Cảm biến DHT11 là cảm biến dùng để đo nhiệt độ và độ ẩm, được tích hợp trong một mạch duy nhất, chúng ta chỉ việc nối dây nguồn (Vcc, GND) và dây tín hiệu (data) vào các chân GPIO của Raspberry Pi tương ứng.

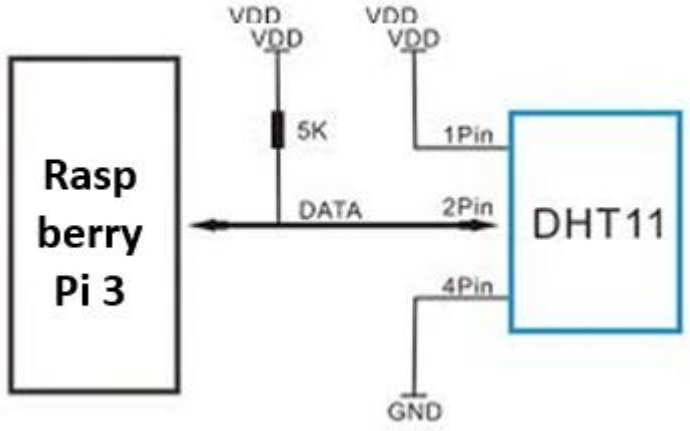


Hình 4.8 Cảm biến nhiệt độ độ âm DHT11

4.3.2 ***Thông số kĩ thuật***

- Điện áp hoạt động: 3-5.5V DC
- Ngưỡng độ ẩm: 20 - 90%
- Sai số độ ẩm: ± 5%
- Ngưỡng nhiệt độ: 0 - 55^{o}C
- Sai số nhiệt độ: ± 2^{oC}
- Kích thước: 15.5mm x 12mm x 5.5mm
- Tần số lấy mẫu: 1Hz , nghĩa là 1 giây DHT11 lấy mẫu một lần.
- 4 chân: VCC(cực (+) nguồn), DATA(chân tín hiệu), NC, GND(cực (-) nguồn)

4.3.3 ***Nguyên lý hoạt động***



Hình 4.9. Sơ đồ kết nối DHT11 với Raspberry Pi 3

DHT11 gửi và nhận dữ liệu với một dây tín hiệu DATA, với chuẩn dữ liệu truyền 1 dây này, chúng ta phải đảm bảo sao cho ở chế độ chờ (idle) dây DATA có giá trị ở mức cao, nên trong mạch sử dụng DHT11, dây DATA phải được mắc với một trở kéo bên ngoài (thông thường giá trị là 4.7kΩ).

Dữ liệu truyền về của DHT11 gồm 40bit dữ liệu theo thứ tự:

8 bit biểu thị phần nguyên của độ ẩm + 8 bit biểu thị phần thập phân của độ ẩm + 8 bit biểu thị phần nguyên của nhiệt độ + 8 bit biểu thị phần thập phân của nhiệt độ + 8 bit check sum

Ví dụ: ta nhận được 40 bit dữ liệu như sau:

0011 0101 0000 0000 0001 1000 0000 0000 0100 1101

Tính toán:

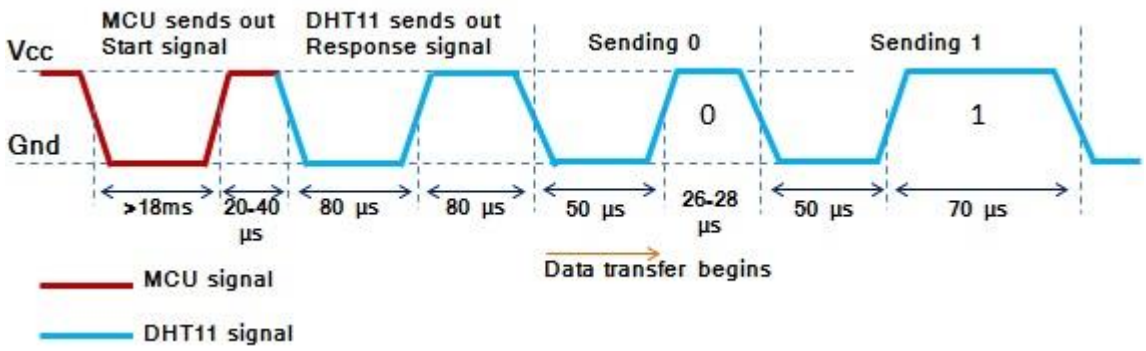
8 bit high humidiy	+	8 bit low humidity	+	8 bit high temp	+	8 bit low temp	=	8 bit check sum
0011 0101		0000 0000		0001 1000		0000 0000		0100 1101

Độ ẩm: 0011 0101 = 35H = 53% (ở đây do phần thập phân có giá trị 0000 0000, nên ta bỏ qua không tính phần thập phân)

Nhiệt độ: 0001 1000 = 18H = 24°C (ở đây do phần thập phân có giá trị 0000 0000, nên ta bỏ qua không tính phần thập phân)

Để có thể giao tiếp với DHT11 theo chuẩn 1 chân vi xử lý thực hiện theo 2 bước chính:

- ❖ Gửi tín hiệu muốn đo (Start) tới DHT11, sau đó DHT11 xác nhận lại.
- ❖ Khi đã giao tiếp được với DHT11, Cảm biến sẽ gửi lại 5 byte dữ liệu và nhiệt độ đo được.



Hình 4.10. Quy trình gửi nhận tín hiệu với DHT11 [3]

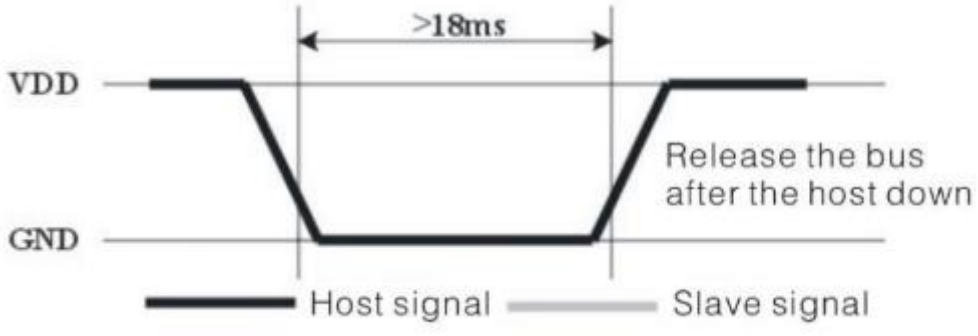
Quy trình làm việc với DHT11 như sau

- ❖ Bước 1: Gửi tín hiệu start tới DHT11 sau đó DHT11 xác nhận lại.

- Sau khi cấp nguồn cho cảm biến, DHT11 sẽ tiến hành đo nhiệt độ và độ ẩm, ghi lại dữ liệu. Chân DATA sẽ ở trạng thái Input và mức cao để chờ người dùng gửi tín hiệu bắt đầu

- Rapspberry Pi thiết lập chân GPIO nối với chân DATA là output, kéo chân DATA xuống 0 trong khoảng thời gian >18ms. Khi đó DHT11 sẽ hiểu bộ xử lý trung

tâm muốn đo giá trị nhiệt độ và độ ẩm. Chúng ta đưa chân DATA lên 1, sau đó thiết lập lại là chân đầu vào.



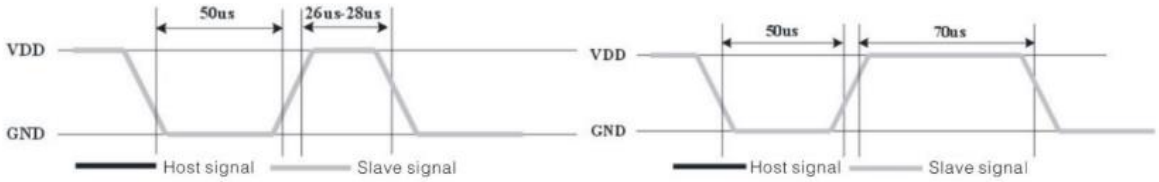
Hình 4.11. Raspberry Pi gửi tín hiệu Start [3]

- Sau khoảng 20-40us, DHT11 sẽ kéo chân DATA xuống thấp. Nếu >40us mà chân DATA không được kéo xuống thấp nghĩa là ko giao tiếp được với DHT11.
- Chân DATA sẽ ở mức thấp 80us sau đó nó được DHT11 kéo nên cao trong 80us. Bằng việc giám sát chân DATA, bộ xử lý trung tâm có thể biết được có giao tiếp được với DHT11 không. Nếu tín hiệu đo được DHT11 lên cao, khi đó hoàn thiện quá trình giao tiếp của Rapspberry Pi với DHT.

❖ Bước 2: đọc giá trị trên DHT11

- DHT11 sẽ trả giá trị nhiệt độ và độ ẩm về dưới dạng 5 byte. Trong đó:
  - Byte 1: giá trị phần nguyên của độ ẩm (RH%)
  - Byte 2: giá trị phần thập phân của độ ẩm (RH%)
  - Byte 3: giá trị phần nguyên của nhiệt độ (TC)
  - Byte 4 : giá trị phần thập phân của nhiệt độ (TC)
  - Byte 5: kiểm tra tổng.

  Nếu Byte 5 = (8 bit) (Byte1 +Byte2 +Byte3 + Byte4) thì giá trị độ ẩm và nhiệt độ là chính xác, nếu sai thì kết quả đo không có nghĩa.
- Đọc dữ liệu: Sau khi giao tiếp được với DHT11, DHT11 sẽ gửi liên tiếp 40 bit 0 hoặc 1 về chân kết nối với Raspberry Pi, tương ứng chia thành 5 byte kết quả của nhiệt độ và độ ẩm.

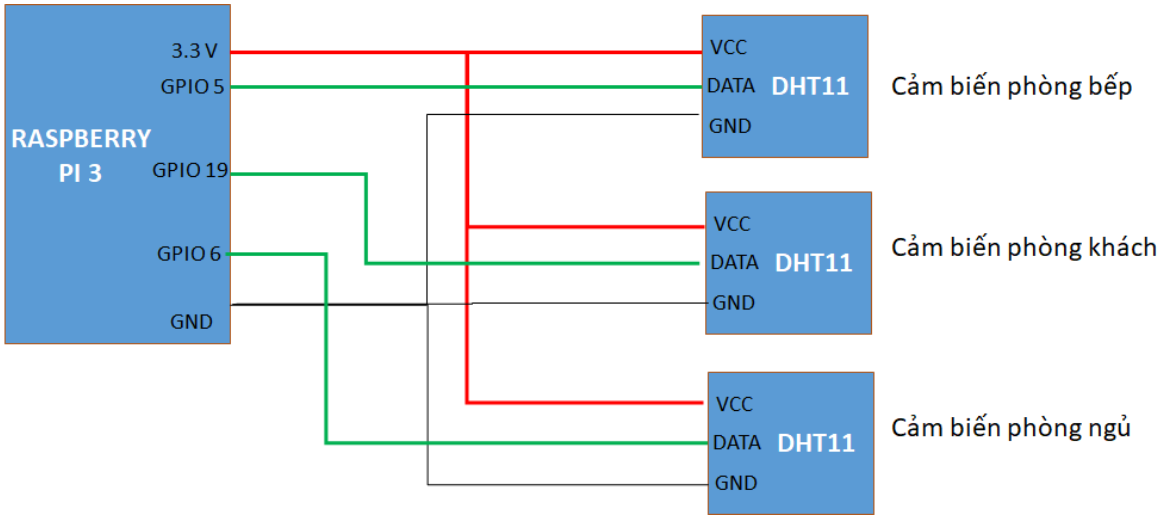


Hình 4.12. Định dạng bit 0 và bit 1 được gửi [3]

- Sau khi tín hiệu được đưa về 0, ta đợi chân DATA của MCU được DHT11 kéo lên 1. Nếu chân DATA là 1 trong khoảng 26-28 us thì là 0, còn nếu tồn tại 70us là 1. Do đó trong lập trình ta bắt sườn lên của chân DATA, sau đó độ trễ 50us. Nếu giá trị đo được là 0 thì ta đọc được bit 0, nếu giá trị đo được là 1 thì giá trị đo được là 1. Cứ như thế chúng ta sẽ đọc các bit tiếp theo để thu thập nhiệt độ và độ ẩm.

4.3.4 *Kết nối với Raspberry Pi*

Trong đề tài này em sẽ sử dụng 3 cảm biến nhiệt độ ẩm ở 3 phòng trong căn nhà. Ba cảm biến sử dụng nguồn 3.3V của Raspberry Pi, các chân DATA kết nối với các chân GPIO 5, GPIO 19, GPIO 6 để truyền nhận dữ liệu.



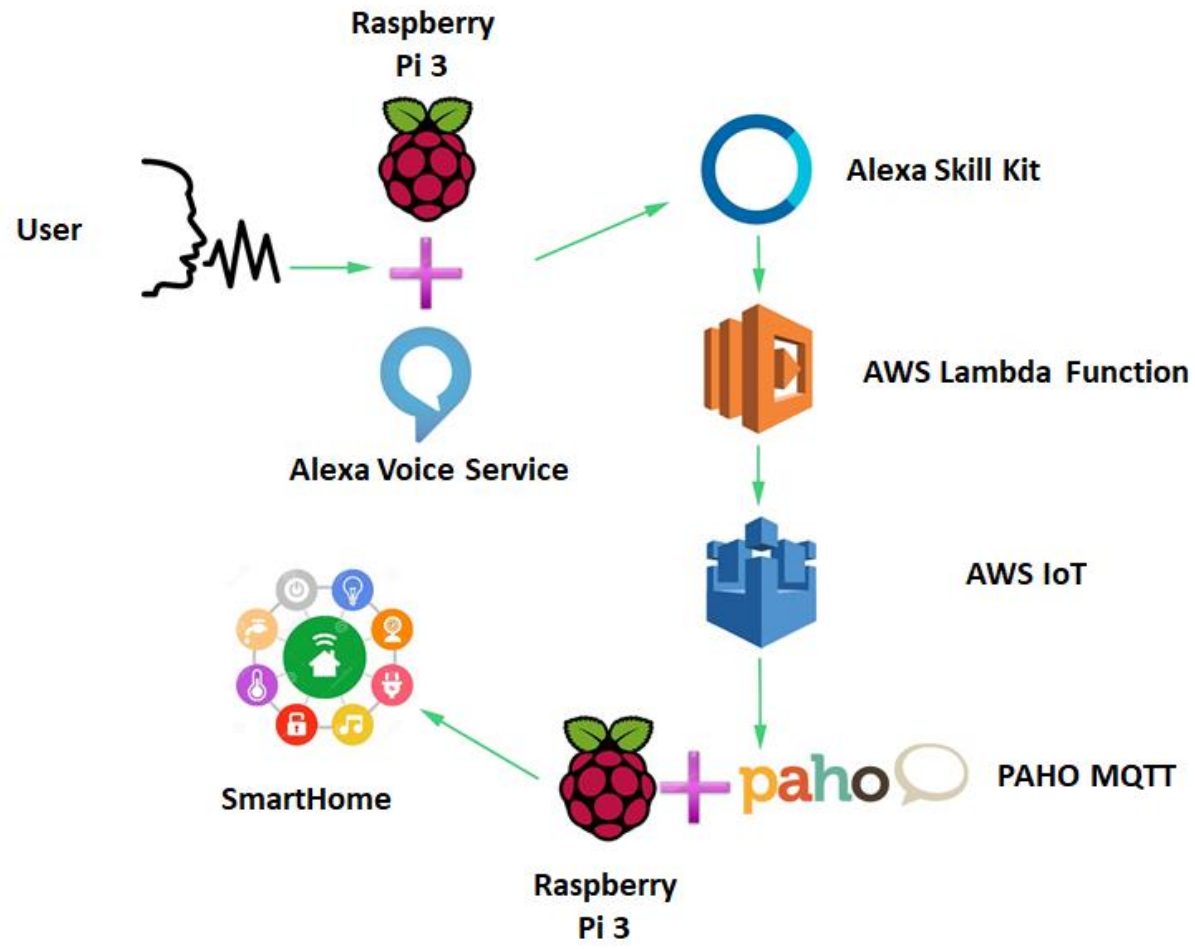
Hình 4.13. Sơ đồ kết nối các cảm biến DHT11 với Raspberry Pi

4.4 Khối xử lý giọng nói – Sử dụng các dịch của Amazon

4.4.1 *Sơ đồ khối tổng quát*

Chúng ta có thể sử dụng các dịch vụ của Amazon để sử dụng giọng nói điều khiển các thiết bị đèn của phòng khách, phòng bếp và phòng ngủ cũng như đọc thông tin nhiệt độ độ ẩm:

- Alexa Voice Service (AVS): Được tích hợp với Raspberry, người dùng có thể sử dụng giọng nói với các câu lệnh điều khiển thiết bị, hỏi đáp kiến thức.
- Alexa Skill Kit (ASK): Đây là dịch vụ của Amazon, từ đây người dùng có thê to xác định các yêu cầu có thể có của người dùng, cụ thể trong đề tài này là các câu lệnh điều khiển thiết bị, từ đây chúng ta có thể xác định được các yêu cầu, các trường hợp nói nào ứng với các yêu cầu đó.
- Message Queuing Telemetry Transport (MQTT): Là một giao thức gửi dưới dạng *publish*/*subscribe* sử dụng cho các thiết bị Internet of Things với băng thông thấp, độ tin cậy cao và khả năng sử dụng được sử dụng trong mạng lưới không ổn định. Trong đề tài này, chúng ta sử dụng giao thức MQTT với thư viện Python paho để nhận lệnh điều khiển từ AWS IoT khi có sự thay đổi về trạng thái yêu cầu lệnh và xử lý điều khiển thiết bị.
- AWS IoT: Đây là dịch vụ đám mây của Amazon, thiết bị của người dùng có thể kết nối tới, gửi trạng thái hoặc nhận lệnh điều khiển từ đây
- AWS Lambda Function: Dịch vụ điện toán đám mây, là một máy chủ ảo cho phép người dùng thực hiện, xử lý các mã lệnh. Đây là cầu nối giữa ASK và AWS IoT để tiếp nhận và xử lý các yêu cầu của người dùng từ AVS và gửi yêu cầu tới AWS IoT



Hình 4.14 Kiến trúc khối xử lý giọng nói sử dụng AVS

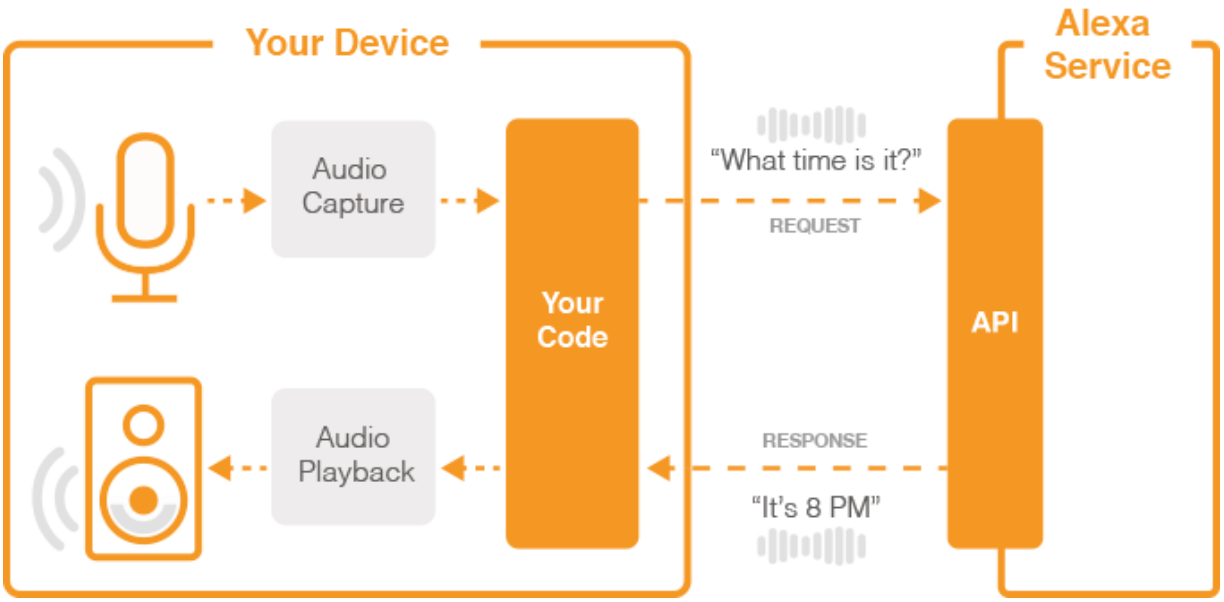
4.4.2 ***Alexa Voice Service***

Là dịch vụ đám mây thông minh của Amazon cho phép chúng ta có thể kết nối với các thiết bị có microphone và loa. Khi tích hợp với AVS, chúng ta có thể tích hợp với các khả năng cốt lõi của Alexa và các thư viện khác của bên thứ ba đang được phát triển.

Chúng ta có thể dễ dàng cài đặt và kết nối Raspberry Pi với Alexa Voice Service qua một ứng dụng được cung cấp bởi Amazon. Bằng cách tích hợp, chúng ta có một số lợi ích sau:

- Dễ dàng cho người dùng kiểm soát thiết bị bằng cách trực quan nhất với giao diện thoại bằng ngôn ngữ tự nhiên, người sử dụng không cần phải mất nhiều thao tác để tiếp cận điều khiển thiết bị nữa

- Alexa Voice Service dựa trên dịch vụ đám mây và được sử dụng theo nhu cầu của bạn, chúng ta khong cần phải lo lắng về việc cung cấp phần cứng và quản lý cơ sở hạ tầng về việc xử lý dữ liệu thoại.
- Không cần phải có kinh nghiệm trong việc xử lý giọng nói tự nhiên. Chỉ cần tích hợp AVS vào thiết bị, chúng ta có thể sử dụng một cách dễ dàng
- Sử dụng AVS với khả năng tự tạo khả năng hội thoại cho riêng bạn một cách miễn phí



Hình 4.15. Dịch vụ Alexa Voice Service

Người dùng tương tác với AVS bằng cách nhấn nút bấm trên của Raspberry Pi 3 hoặc có thể sử dụng từ kích hoạt "Alexa", "Amazon", "Computer"… để bắt đầu ra lệnh. Raspberry Pi sẽ ghi lại câu lệnh được yêu cầu và gửi nó đến dịch vụ AVS. Sau khi nhận được yêu cầu, AVS sẽ xử lý và phản hồi thông qua loa của Raspberry Pi. Ngoài các kịch bản hội thoại có sẵn của AVS, chúng ta có thể sử dụng Alexa Skills Kit (ASK) để có thể tự thiết lập và tạo ra các kịch bản chính bạn hoặc sử dụng của các nhà phát triển khác. Người dùng có thể tạo ra các khả năng trả lời các câu hỏi kiến thức chung, cung cấp dự báo thời tiết, chơi nhạc tìm hiểu kiến thức Wikipedia hay điều khiển các thiết bị nhà thông minh….

Amazon cung cấp cho chúng ta một ứng dụng Java, một Node.js server. Chúng ta sử dụng Node.js server để lấy mã token đăng nhập với Amazon bằng cách sử dụng trình duyệt web của Raspberry. Người dùng phải có tài khoản phát triển của Amazon để tạo ra mã định danh: *ClientId, ClientSecret, ProductID.* Khi mở ứng dụng Java, ứng dụng sẽ yêu cầu chúng ta mở trình duyệt để kết nối tới Node.js server, sau khi xác thực thông tin, chúng ta sẽ được chuyển tiếp tới trang đăng nhập của Amazon và được trả về một mã token truy nhập để sử dụng cho các yêu cầu tới Alexa Voice Service.

4.4.3 ***Alexa Skill Kit***

Alexa Skill Kit (ASK) là một tập ở các công cụ, tài liệu, mã lệnh mà giúp chúng ta tự khai báo và dễ dàng thêm các kĩ năng thoại tới Alexa.

Để làm việc với ASK, chúng ta cần thiết lập các thông số sau

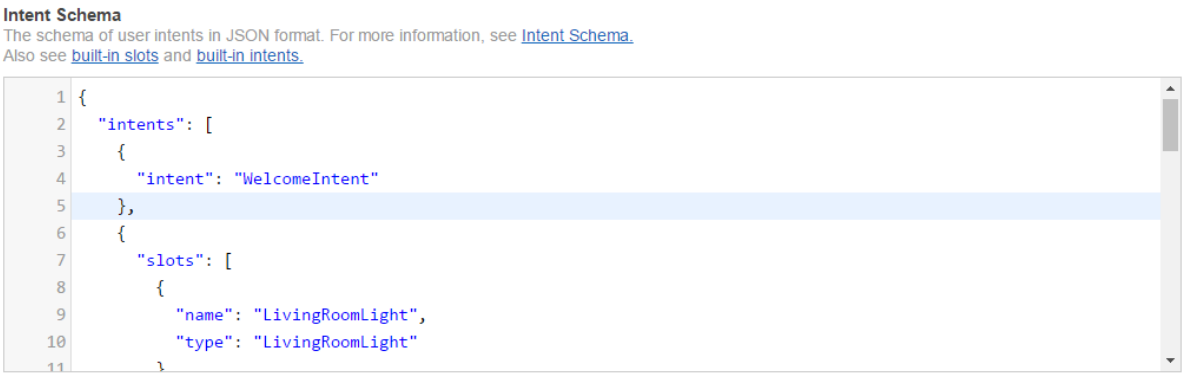
4.4.3.1 *Invocation Name:*

Tên của kĩ năng để người dùng có thể sử dụng khi giao tiếp với Alexa. Trong đề tài này em chọn tên Smart Home, khi muốn yêu cầu tới ASK, chúng ta có thể nói vào micro

- "Alexa, Tell Smart Home to …"
- "Alexa, Ask Smart Home to …"

4.4.3.2 *Intent Schema:*

Là tập hợp các *intents* – tên đại diện cho các hành động đáp ứng yêu cầu của người dùng. Các *intents* này bao gồm thành phần là *slots* để chỉ rõ hành động và thiết bị mà người dùng yêu cầu. *Intent Schema* được viết dưới dạng cấu trúc JSON



Hình 4.16. Cấu hình Intent Schema

Chúng ta có các *intents* sau để phân loại yêu cầu của người dùng:

- WelcomeIntent: Khi người dùng kết nối, AVS sẽ liên kết tới *intents* này của ASK
- LivingRoomLightIntent, BedRoomLightIntent, KitchenLightIntent: Khi người dùng ra lệnh yêu cầu tắt mở các thiết bị đèn, AVS sẽ liên kết tới các *intents* này.
- StopIntent: Khi người dùng kết thúc phiên hoạt động, intent này sẽ được gọi để phản hồi cho người dùng

4.4.3.3 *Sample utterances:*

Là tập hợp các câu lệnh có thể có của người dùng khi ra lệnh tới AVS, ứng với từng *Intents* sẽ có các câu lệnh được định nghĩa tương ứng

Hình 4.17. Cấu hình Sample Utterances

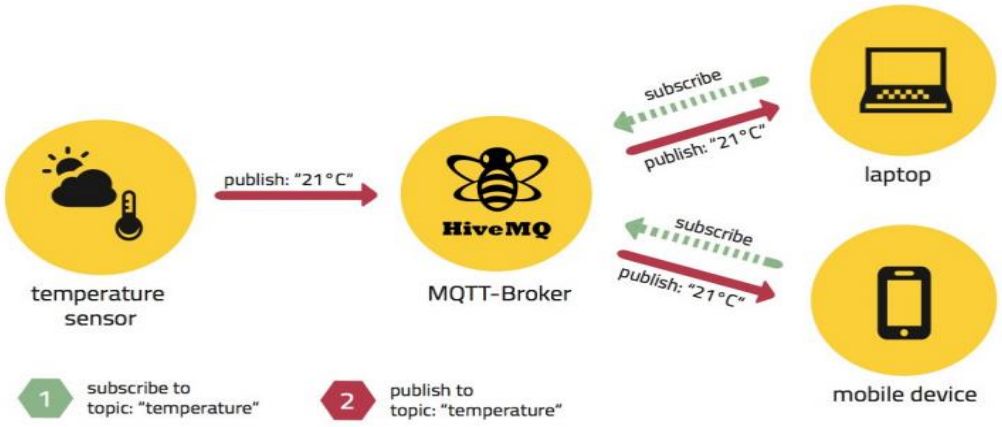
4.4.4 ***MQTT***

4.4.4.1 *MQTT là gì?*

MQTT (Message Queuing Telemetry Transport) xuất hiên lần đầu vào năm 1999 và được IBM đưa vào sử dụng vào năm 2013, là một giao thức với cơ chế *publish/subscribe/topic* sử dụng cho các thiết bị Internet of Things với băng thông thấp, độ tin cậy cao và khả năng được sử dụng trong mạng lưới không ổn định, giá băng thông mạng đắt đỏ hay khi chạy trên thiết bị nhúng bị giới hạn về tài nguyên tốc độ và bộ nhớ để truyền nhận dữ liệu.

4.4.4.2 *Mô hình giao thức MQTT*

MQTT theo mô hình *Client-Server* truyền thống, nghĩa là sẽ có 1 MQTT server hay còn gọi là *Broker* làm trung gian giữ vai trò quản lý các kết nối giữa các MQTT *client*. Đây là giao thức cho phép các MQTT *client* xuất bản (*publish*) và đăng ký(*subscribe*) vào bản tin được truyền qua mạng TCP/IP. Ví dụ ứng dụng trong hệ thống SmartHome như hình dưới, từ điện thoại chúng ta muốn giám sát nhiệt độ của phòng khách thì điện thoại sẽ đăng kí *Subscribe* Topic nhiệt độ từ phòng khác và phòng ngủ với server. Còn 2 cảm biến nhiệt độ ở phòng khách sẽ liên tục gửi dữ liệu nhiệt độ khi nhiệt độ phòng khách thay đổi. Từ đó *broker* sẽ thông báo thay đổi nhiệt độ này cho điện thoại.



Hình 4.18. MQTT với giám sát nhiệt độ

Một số khái niệm của MQTT

- *Topic*: Địa chỉ kênh để *client* có thể đăng ký, các kênh được phân cấp, giống như một hệ thống tập tin (ví dụ kitchen/oven/temperature). Ký tự "/" được sử dụng để phân tách các cấp của *topic*

Khi khai báo một topic, chúng ta cần chú ý các nguyên tắc tên sau:

- Một topic nên có ít nhất 1 kí tự.
- Topic name là phân biệt chữ hoa, chữ thường. Ví dụ, ACCOUNTS and Accounts là 2 cái tên khác nhau.
- Topic name có thể bao gồm dấu cách. Ví dụ, Accounts payable là hợp lệ.

- Việc bắt đầu bằng dấu "/" sẽ tạo ra một topic khác. Ví dụ, /finance sẽ khác với finance. /finance sẽ matches "+/+" and "/+", nhưng không matches với "+".
- Không đưa kí tự null (Unicode \x0000) vào trong bất kì topic nào.

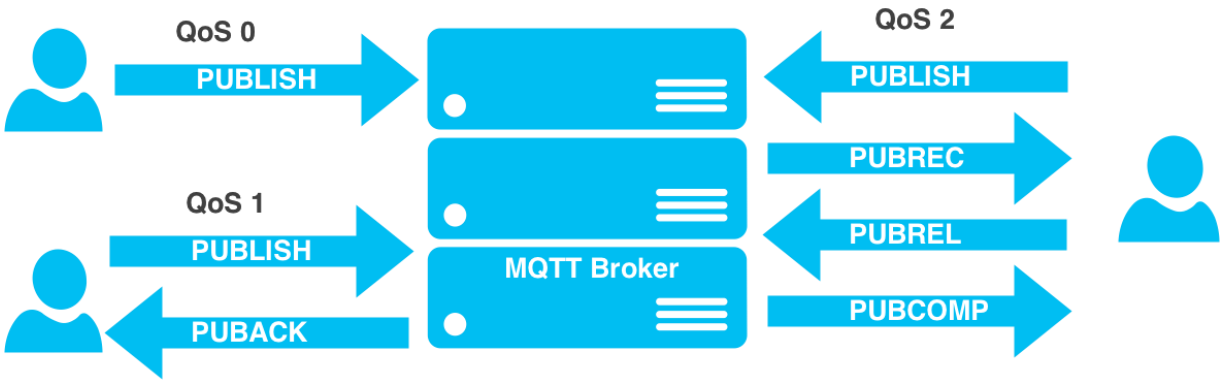
- *Publish*: là phương thức khi *client* gửi dữ liệu tới *topic* để thông báo thay đổi về trạng thái hay yêu cầu điều khiển

- *Subscribe*: là phương thức khi *client* muốn đăng ký một *topic* nào đó, *client* sẽ nhận được dữ liệu bất cứ khi nào *client* khác gửi dữ liệu tới *topic* đã đăng ký

- *Payload*: Là nội dung chứa trong bản tin được gửi.

- *QoS*: *QoS (Qualities of Service)* là tùy chọn về phương thức khi đăng ký hay gửi tới topic

- *QoS0: Broker/Client* sẽ gửi dữ liệu đúng 1 lần, quá trình gửi được xác nhận bơi giao thức TCP/IP
- *QoS1: Broker/Client* sẽ gửi dữ liệu với ít nhất một lần xác nhận từ phía nhận, có nghĩa là có thể nhiều hơn một lần xác nhận đã nhận được dữ liệu
- *QoS2: Broker/Client* đảm bảo khi gửi dữ liệu thì phía nhận chỉ nhận được đúng 1 lần, quá trình này trải qua 4 bước bắt tay



Hình 4.19. Các tùy chọn QoS

Một gói tin có thể được gởi ở bất kỳ QoS nào, và các *client* cũng có thể *subscribe* với bất kỳ yêu cầu QoS nào. Có nghĩa là *client* sẽ lựa chọn QoS tối đa mà nó có để nhận tin. Ví dụ, nếu 1 gói dữ liệu được *publish* với QoS2, và *client* đăng ký với QoS0, thì gói dữ liệu được nhận về *client* này sẽ được *broker* gởi với QoS0, và 1 *client* khác đăng ký cùng kênh này với QoS 2, thì nó sẽ được *broker* gởi dữ liệu với QoS2.

Một ví dụ khác, nếu 1 *client subscribe* với QoS2 và gói dữ liệu gởi vào kênh đó *publish* với QoS0 thì *client* đó sẽ được *broker* gởi dữ liệu với QoS0. QoS càng cao thì càng đáng tin cậy, đồng thời độ trễ và băng thông đòi hỏi cũng cao hơn.

- *Retain:* là một trường trong bản tin MQTT để quy định về việc lưu trữ bản tin MQTT. Nếu *retain* bằng 1, khi gói tin được *publish* từ *client, broker* phải lưu trữ lại gói tin với QoS, và nó sẽ được gởi đến bất kỳ *client* nào đăng ký cùng kênh *topic* trong tương lai. Khi một *client* kết nối tới *broker* và đăng ký, nó sẽ nhận được gói tin cuối cùng có *retain* = 1 với bất kỳ topic nào mà nó đăng ký trùng. Tuy nhiên, nếu *broker* nhận được gói tin mà có QoS = 0 và *retain* = 1, nó sẽ huỷ tất cả các gói tin có *retain* = 1 trước đó. Và phải lưu gói tin này lại, nhưng hoàn toàn có thể huỷ bất kỳ lúc nào.

4.4.4.3 *Định dạng bản tin MQTT*

❖ Phần header cố định

Tất cả các bản tin MQTT đều chứa phần cố định dưới đây:

bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 1	Loại bản tin				Cờ DUP	Cờ QoS level		Cờ RETAIN

Bảng 4.3 Cấu trúc header bản tin MQTT

- 4 bit đầu quy định loại bản tin MQTT

Phân loại	**Giá trị**	**Miêu tả**
Reserved	0	Chưa dùng
CONNECT	1	*Client* yêu cầu kết nối
CONNACK	2	Kết nối được chấp nhận
PUBLISH	3	Bản tin *publish*
PUBACK	4	Bản tin *publish* được chấp nhận
PUBREC	5	*publish* đã được nhận (đảm bảo nhận được phần 1)
PUBREL	6	Bản tin *publish* phần 2 (sau khi nhận được PUBREC)
PUBCOMP	7	Xuất bản hoàn thành (sau khi nhận được PUBREL)
SUBSCRIBE	8	Yêu cầu đăng ký từ *client*
SUBACK	9	Yêu cầu đăng ký được chấp nhận

Phân loại	Giá trị	Miêu tả
UNSUBSCRIBE	10	Yêu cầu hủy đăng ký
UNSUBACK	11	Yêu cầu hủy đăng ký được chấp nhận
PINGREQ	12	Yêu cầu PING
PINGRESP	13	Trả lời yêu cầu PING
DISCONNECT	14	Client đang mất kết nối
Reserved	15	Reserved

Bảng 4.4 Các giá trị của 4 bit đầu bản tin MQTT

- 4 bit còn lại của byte đầu tiên tương ứng sẽ mô tả trạng thái của DUP, QoS, RETAIN

Vị trí bit	Tên viết gọn	Miêu tả
3	DUP	Truyền lặp lại
2-1	QoS	Quality of Service (mức độ tin cậy)
0	RETAIN	cờ RETAIN

Bảng 4.5. 4 bit còn lại của phần header bản tin MQTT

- DUP: Cờ này được bật khi *client* hoặc *server* đang cố gửi lại một gói tin loại *publish, pubrel, subscribe* hoặc *unsubscribe*. Giá trị này được sử dụng trong các bản tin có QoS lớn hơn 0 và yêu cầu ACK phản hồi. Phía nhận sẽ tự kiểm tra xem gói tin ở trên đã được chuyển đến trước đó hay chưa và sẽ quyết định truyền lặp lại
- QoS: Cờ này sẽ cho biết mức độ tin cậy của việc chuyển bản tin *publish*. Giá trị của QoS được miêu tả trong bảng dưới đây.

Giá trị QoS	bit 2	bit 1	Miêu tả		
0	0	0	Chỉ gửi 1 lần	Chỉ gửi 1 lần, không quan tâm kết quả	<=1
1	0	1	Ít nhất 1 lần	Xác nhận bằng ACK	>=1
2	1	0	Chính xác 1	Nhận đảm bảo qua quá trình bắt tay	=1

Giá trị QoS	bit 2	bit 1	Miêu tả
			lần
3	1	1	Chưa dùng

Bảng 4.6. Các giá trị QoS

- RETAIN: Cờ này chỉ được sử dụng ở bản tin *publish* Khi client gửi 1 bản tin *publish* đến server, nếu cờ Retain được set (1), thì server hiểu rằng phải lưu trữ message này sau khi chuyển nó đến các client đã đăng ký.

❖ Phần nội dung bản tin (payload)

Là các byte còn lại chứa nội dung của bản tin. Ví dụ khi muốn gửi một lệnh bật bóng đèn tới *broker, client* sẽ gửi bản tin có payload là "ON"

4.4.5 ***AWS IoT***

4.4.5.1 *AWS IoT là gì*

Amazon Web Service Internet of Things (AWS IoT) là một nền tảng điện toán đám mây cho phép các nền tảng khác kết nối dễ dàng để quản lý các thiết bị khác. Trong phạm vi đề tài này, em sử dụng AWS IoT thu thập và quản lý các dữ liệu trạng thái của ngôi nhà thông minh và từ đó chúng ta có thể dễ dàng thiết lập và sử dụng chúng qua các giao thức HTTP, WebSockets và MQTT…

4.4.5.2 *Đăng ký Thing*

Trước tiên, để có thể lưu trữ được các trạng thái và yêu cầu đối với thiết bị chúng ta cần đăng ký với AWS IoT, *Thing* là thuật ngữ đại diện cho thiết bị hoặc một thực thể logic trên đám mây với một số mô tả sau:

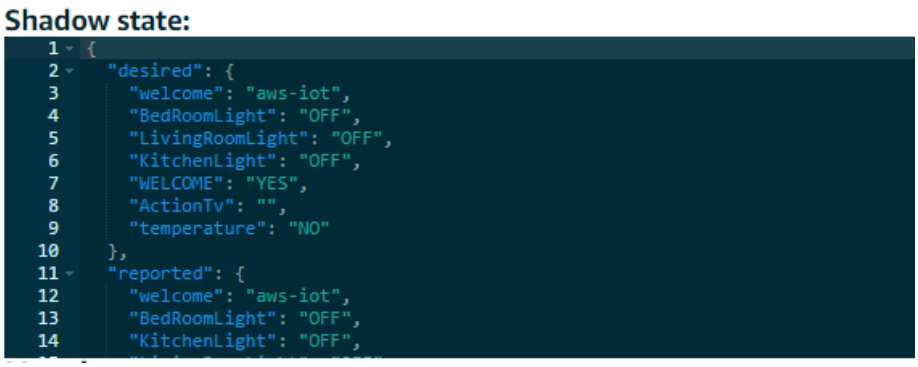
- *Thing name:* Tên của thiết bị, đối với dự án này là: RaspberryPi
- *Thing shadow: Thing shadow* là một đoạn mã JSON được lưu trữ và truy xuất thông tin trạng thái hiện tại và yêu cầu điều khiển cho thiết bị Raspberry Pi của chúng ta. Để sử dụng *thing shadow* chúng ta có thể truy cập thông qua giao thức MQTT hoặc HTTP

- *Certificate:* để truy cập được vào AWS IoT cũng như đảm bảo an toàn cho dữ liệu của chúng ta, AWS IoT cung cấp cho chúng ta chứng chỉ khi đăng ký thiết bị.

4.4.5.3 *Tìm hiểu Thing Shadow Document*

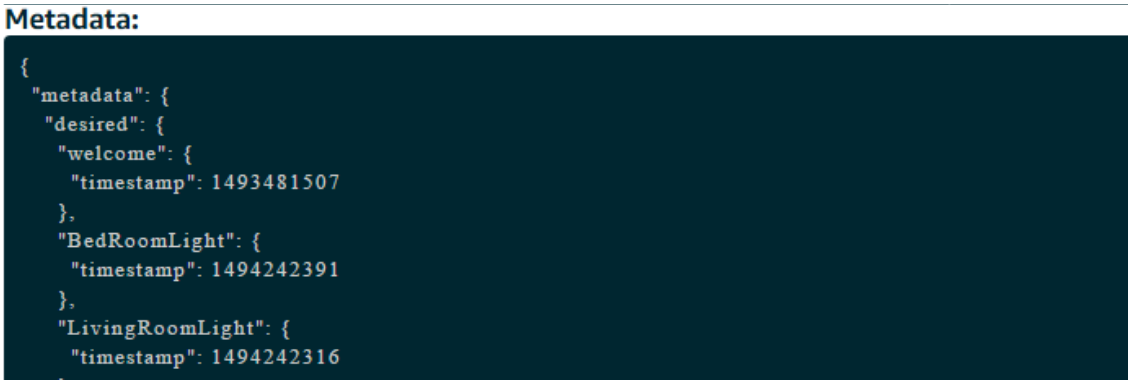
Shadow Document là nơi lưu trữ dữ liệu của thiết bị dưới dạng JSON và có các thuộc tính sau:

- *Shadow state*: đoạn mã JSON chứa hai thuộc tính *desired* và *reported. Desired* mô tả cho các yêu cầu đối với thiết bị thông qua AWS IoT, tương tự như vậy, *reported* mô tả cho việc báo cáo trạng thái hiện tại của thiết bị. Người dùng có thể cập nhật cũng như truy xuất để thực hiện mà không cần kết nối trực tiếp tới thiết bị.



Hình 4.20. Shadow State

- *Metadata:* Đoạn mã JSON chứa thông tin về dữ liệu về thời gian cập nhật đối với từng thuộc tính trong phần *shadow state*, dựa vào đây chúng ta có thể xác định thời điểm mỗi thuộc tính được cập nhật.



Hình 4.21. Metadata của Thing Shadow Document

- *Version:* Mỗi khi *Thing Shadow Document* được cập nhật, số hiệu phiên bản sẽ được tăng lên, chúng ta sẽ xác định được phiên bản nào là phiên bản được cập nhật gần đây nhất.
- *ClientToken:* Một đoạn chuỗi cho thiết bị để xác nhận rằng đang trong phiên làm việc của một *client*, tạo ra sự liên kết giữa việc đáp ứng và yêu cầu từ *client* với AWS IoT

4.4.5.4 *Làm việc với Thing Shadows*

AWS IoT cung cấp 3 phương thức cho việc làm việc với *thing shadows:*

- UPDATE: Tạo một *thing shadows* nếu trước đó chưa tồn tại hoặc cập nhật nội dung của *thing shadows* dữ liệu được cung cấp. Dữ liệu sẽ được lưu trữ với thông tin thời gian để chỉ rõ thời điểm UPDATE cuối cùng. Bản tin sẽ được gửi cho tới tất cả các *client* đã đăng ký với hai trường khác nhau là *desired* (yêu cầu) và *reported* (báo cáo). Ví dụ khi người dùng muốn điều khiển bật bóng đèn phòng khách, người dùng ra lệnh với AVS, AVS liên kết với ASK sẽ gửi yêu cầu UPDATE chứa trường *desired*, sau khi có sự thay đổi ở trường *desired*, Raspberry Pi sẽ thực hiện yêu cầu của người dùng và gửi bản tin UPDATE chứa trường *reported* tới AWS IoT
- GET: gửi yêu cầu tới AWS IoT để nhận trạng thái mới nhất được lưu ở *thing shadows.* Phương thức này sẽ trả về một đoạn mã JSON của *thing shadows* với đầy đủ các thông số. Chúng ta sẽ sử dụng phương thức này để lấy dữ liệu thông số nhiệt độ, hoặc lấy trạng thái hiện tại của thiết bị đèn để gửi lệnh điều khiển phù hợp
- DELETE: Xóa tất cả nội dung của một *thing shadows.* Nó sẽ xóa thông tin trong mã JSON. Chúng ta sẽ không thể khôi phục *thing shadows* đã xóa nhưng có thể tạo mới *thing shadows* với cùng tên.

Để sử dụng 3 phương thức trên chúng ta có thể sử dụng MQTT hoặc RESTful API thông qua giao thức HTTPS, trong đề tài này em sử dụng giao thức MQTT

Thông qua MQTT, chúng ta có thể cập nhật, truy xuất hoặc xóa thông tin trạng thái của *thing shadows* thông qua việc gửi các bản tin tới kênh *topic* với tiền tố bắt đầu là *$aws/things/thingName/shadow*. Thông tin các *topic* nằm ở bảng sau đây

STT	Topic	Hành động	Mô tả
1	/update	*Publish/ Subscribe*	*Publish* một yêu cầu trạng thái tới *shadow documents* để cập nhật *thing shadow*AWS IoT trả lời bằng việc *publish* tới topic */update/accepted* hoặc */update/rejected* tương ứng
2	/update/accepted	*Subscribe*	AWS IoT *publish* tới topic này khi yêu cầu thay đổi *thing shadow* được chấp thuận
3	/update/documents	*Subscribe*	AWS IoT *publish* tới topic này một đoạn mã dưới dạng JSON bao gồm 2 phiên bản trước và sau khi cập nhật thành công
4	/update/rejected	*Subscribe*	AWS IoT *publish* tới topic này khi yêu cầu thay đổi *thing shadow* bị từ chối
5	/update/delta	*Subscribe*	AWS IoT *publish* tới topic này một đoạn mã dưới dạng JSON bao gồm những thành phần của hai trường *desired* và *reported* mà có sự thay đổi.
6	/get	*Publish/Subscribe*	*Publish* một yêu cầu lấy dữ tới *shadow documents* để lấy thông tin *thing shadow* hiện tại. AWS IoT sẽ trả lời bằng việc *publish* tới topic /get/*accepted* hoặc /get/*rejected* tương ứng
7	/get/accepted	*Subscribe*	AWS IoT *publish* tới topic này khi yêu cầu lấy dữ liệu *thing shadow* được chấp thuận
8	/get/rejected	*Subscribe*	AWS IoT *publish* tới topic này khi yêu cầu lấy dữ liệu *thing shadow* bị từ chối
9	/delete	*Publish/Su*	Để xóa một *thing shadow*, chúng ta *publish* một

		bscribe	bản tin không có nội dung tới topic này
10	/delete/accepted	*Subscribe*	AWS IoT *publish* tới topic này khi yêu cầu xóa *thing shadow* được chấp thuận
11	/delete/rejected	*Subscribe*	AWS IoT *publish* tới topic này khi yêu cầu xóa *thing shadow* bị từ chối

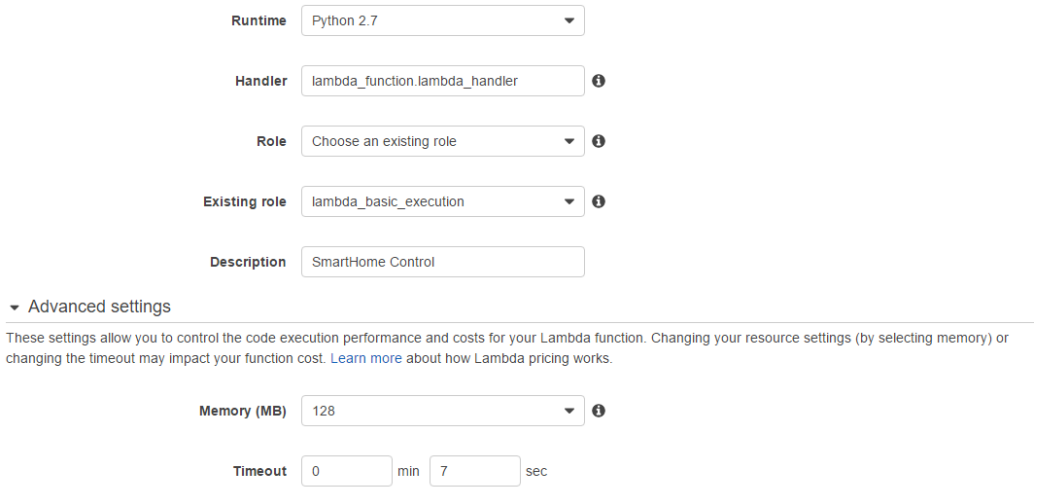
Bảng 4.7. Các topic của Thing shadow

Như vậy chúng ta có thể làm việc với AWS IoT, người dùng có thể truy xuất và thay đổi dữ liệu ở AWS IoT từ ASK. Cầu nối trung gian giữa chúng là AWS Lambda Function

4.4.6 ***AWS Lambda Function***

4.4.6.1 *AWS Lambda Function là gì*

AWS Lambda Function là dịch vụ máy tính và cho phép chúng ta chạy mã lập trình của bạn mà không cần phải cung cấp hay quản lý các máy chủ. *AWS Lambda* thực hiện chạy mã của bạn chỉ khi có yêu cầu được gửi tới với tính sẵn sàng cao và thực hiện tất cả các việc quản lý các tài nguyên tính toán bao gồm bảo trì hệ thống máy chủ, cung cấp và cân chỉnh tài nguyên, theo dõi và ghi lại báo cáo về hoạt động của *Lambda Funtion*. Chúng ta chỉ cần cung cấp mã nguồn bằng một trong các ngôn ngữ như Node.js, Java, C# và Python. Trong đồ án này, em sử dụng ngôn ngữ Python trên AWS Lambda Function



Hình 4.22. Cấu hình môi trường làm việc cho AWS Lambda

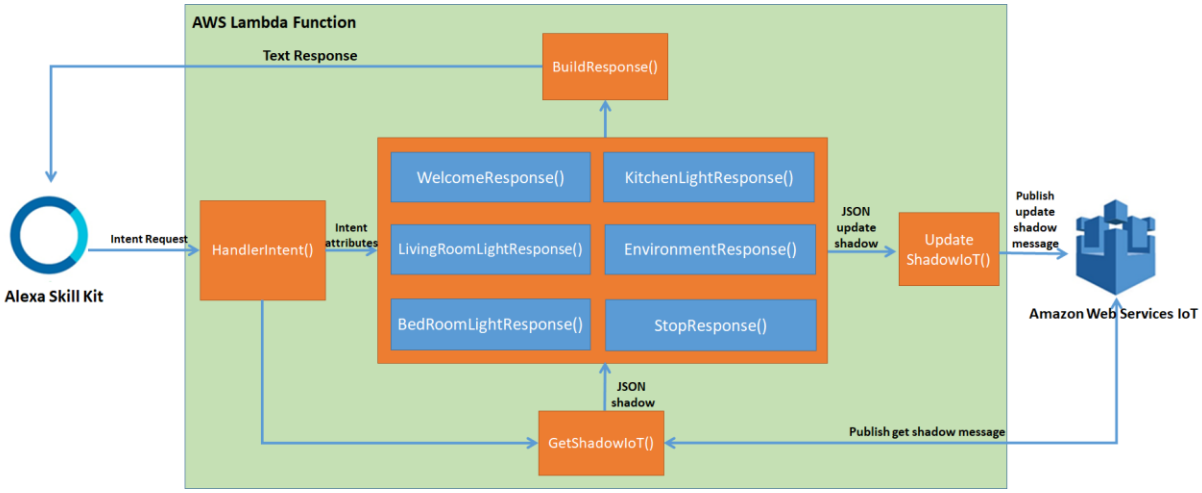
4.4.6.2 *Thiết kế chi tiết*

❖ Sơ đồ kiến trúc phần mềm

Hình 4.23. Sơ đồ luồng dữ liệu AWS Lambda Function

AWS Lambda Function nhận được *Intent Request* từ ASK bao gồm Intent và Slot, qua quá trình xử lý AWS Lambda sẽ *publish* một bản tin *update shadow* tới AWS IoT, bên cạnh đó AWS Lambda cũng sẽ *publish* một bản tin *get shadow* để xác định trạng thái hiện tại của ngôi nhà để tạo đoạn *text response* gửi tới ASK phản hồi cho người dùng

❖ Sơ đồ thiết kế chi tiết



Hình 4.24. Sơ đồ thiết kế chi tiết AWS Lambda Function

Sau khi nhận được lệnh từ người dùng, Alexa Skill Kit sẽ gửi *Intent Request* dưới dạng JSON bao gồm các thông tin *session* và *request* tới AWS Lambda Function. Hàm *HandlerIntent()* sẽ phân tích và xử lý để gửi *Intent Attributes* đến các hàm xử lý tương ng gồm hàm:

- *WelcomeRespone*(): hàm xử lý khi người dùng mở ASK

- *LivingRoomLightResponse*(), *BedRoomLightResponse*(), *KichenLightResponse*(): các hàm xử lý yêu cầu bật tắt đèn các phòng
- *EnvironmentResponse*(): hàm xử lý yêu cầu về môi trường trong các phòng
- *StopReponse*(): Khi người dùng lựa chọn kết thúc, hàm này sẽ được gọi

Bên cạnh đó, hàm *HandlerIntent()* cũng sẽ gọi hàm *GetShadowIoT()* truy xuất lấy trạng thái hiện tại của *thing shadows* để có thể điều khiển một cách chính xác nhất

Sau khi các hàm trên được xử lý, AWS Lambda Function sẽ tạo một đoạn mã JSON chứa yêu cầu của người dùng và gọi hàm *UpdateShadowIoT()* để gửi một yêu cầu cập nhật *thing shadows* và một đoạn mã JSON khác cho hàm *BuildResponse()* để gửi phản hồi yêu cầu cho Alexa Skill Kit, từ đó ASK thông qua dịch vụ AVS phản hồi kết quả cho người dùng.

4.4.6.3 *Kết nối với AWS IoT*

Với ngôn ngữ lập trình Python, để làm việc được với AWS IoT, chúng ta cần sử dụng thư viện AWS IoT Device SDK được cung cấp bởi Amazon

Trước tiên, chúng ta phải tạo *thing shadow* ở AWS IOT và tải về các khóa chứng chỉ cần thiết để xác thực và làm việc.

Dưới đây là đoạn mã cấu hình kết nối với AWS IoT với *thing shadow* được đặt tên là raspberry

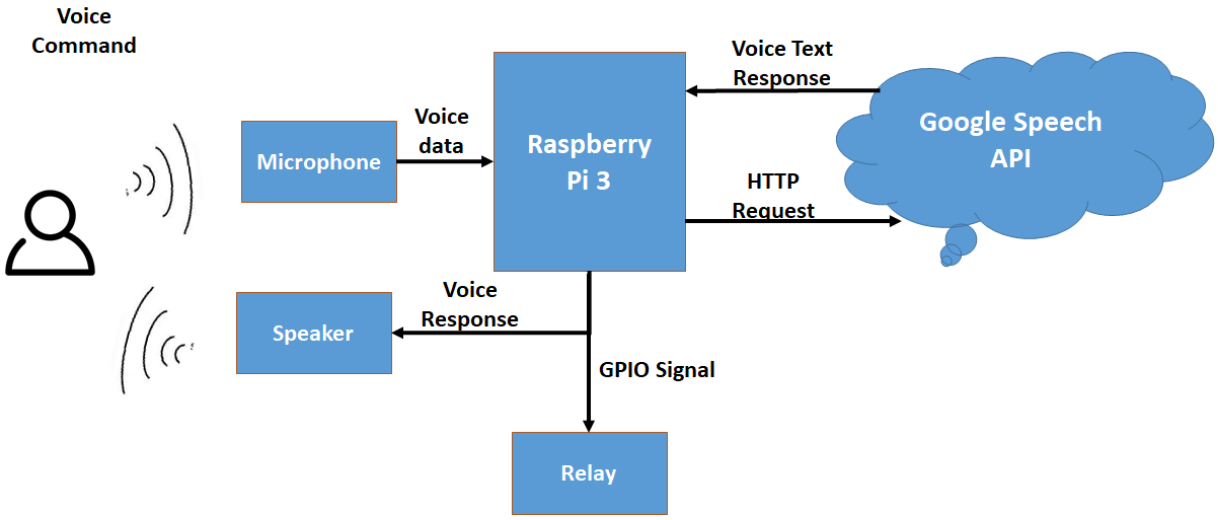
```
# Import SDK packages
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
# For certificate based connection
myAWSIoTMQTTShadowClient = AWSIoTMQTTShadowClient(shadowClient)
myAWSIoTMQTTShadowClient.configureEndpoint(a3oosh7oql9nlc.iot.us-east-
1.amazonaws.com, 8883)
myAWSIoTMQTTShadowClient.configureCredentials("YOUR/ROOT/CA/PATH",
"PRIVATE/KEY/PATH", "CERTIFICATE/PATH")
# For Websocket, we only need to configure the root CA
# myMQTTClient.configureCredentials("YOUR/ROOT/CA/PATH")
# AWSIoTMQTTShadowClient configuration
myAWSIoTMQTTShadowClient.configureConnectDisconnectTimeout(10)  # 10 sec
myAWSIoTMQTTShadowClient.configureMQTTOperationTimeout(5)  # 5 sec
myDeviceShadow =
myAWSIoTMQTTShadowClient.createShadowHandlerWithName("raspberry", True)
myDeviceShadow.shadowGet(IoTShadowCallback_Get, 5)
myDeviceShadow.shadowUpdate(myJSONPayload, IoTShadowCallback_Update, 5)
myDeviceShadow.shadowDelete(IoTShadowCallback_Delete, 5)
myDeviceShadow.shadowRegisterDeltaCallback(IoTShadowCallback_Delta)
```

- Để *publish* một bản tin GET, chúng ta gọi hàm *myDeviceShadow.shadowGet(IoTShadowCallback_Get, 5)* trong đó hàm *IoTShadowCallback_Get* là hàm để xử lý kết quả trả về từ AWS IoT với thời gian cho phép là 5 giây
- Để *publish* một bản tin GET, chúng ta gọi hàm *myDeviceShadow.shadowUpdate(IoTShadowCallback_Update, 5)* trong đó hàm *IoTShadowCallback_Update* là hàm để xử lý kết quả trả về từ AWS IoT với thời gian cho phép là 5 giây
- Để *subscribe* kênh *topic delta* để nhận kết quả tự động trả về bất cứ khi nào có sự thay đổi ở hai trường *desired* và *reported* của *Thing Shadow,* chúng ta gọi hàm *myDeviceShadow.shadowRegisterDeltaCallback (IoTShadowCallback_Delta)* trong đó hàm *IoTShadowCallback_Delta* là hàm để xử lý kết quả trả về từ AWS IoT

4.5 Khối xử lý giọng nói – Sử dụng Google Speech API

Việc xử lý nhận dạng giọng nói để điều khiển thiết bị với Alexa Voice Service đã được hoàn thành tuy nhiên hiện tại AVS vẫn chưa hỗ trợ về nhận dạng giọng nói tiếng Việt. Chính vì vậy em đã thiết kế thêm khối xử lý giọng nói sử dụng Google Speech API để hữu dụng với nhiều đối tượng người Việt Nam.

4.5.1 *Sơ đồ khối tổng quát*



Hình 4.25. Sơ đồ khối khối xử lý giọng nói sử dụng Google Voice API

Người dùng ra lệnh cho bộ xử lý trung tâm (Raspberry Pi 3) thông qua Microphone. Dữ liệu thoại được Raspberry Pi 3 ghi lại và lưu dưới dạng tập tin âm thanh flac, qua quá trình xử lý gửi một HTTP Request tới Google Speech API để chuyển đổi dữ liệu âm thanh sang dữ liệu văn bản. Từ đây bộ xử lý trung tâm sẽ phân tích dữ liệu văn bản trả về để thực hiện lệnh điều khiển tương ứng.

4.5.2 ***Google Speech API.***

4.5.2.1 *Giới thiệu Google Speech API*

Google Speech API là dịch vụ chuyển từ giọng nói sang văn bản. Google Speech API nhận dạng giọng nói rất chính xác và hỗ trợ hơn 80 ngôn ngữ để hỗ trợ người dùng toàn cầu. Chúng ta có thể sử dụng để lấy văn bản từ giọng nói người dùng, điều khiển mọi thứ thông qua giọng nói hoặc chuyển đổi các tập tin âm thanh. Người dùng chỉ cần gửi yêu cầu tới dịch vụ kèm với một tập tin âm thanh, Google Speech sẽ

xử lý và trả về đoạn mã JSON chứa các thông tin được dịch. Chúng ta cũng không cần lo ngại về việc xử lý tín hiệu nhiễu của tập tin âm thanh, Google Speech API sau khi nhận được sẽ xử lý để nhận diện một cách tốt nhất.

4.5.2.2 *Làm việc với Google Speech API [8]*.

- Gửi yêu cầu tới Google Speech API

Để gửi tập tin âm thanh và nhận dữ liệu văn bản đã được chuyển đổi, bạn cần phải gửi tập tin âm thanh theo giao thức HTTP tới địa chỉ: https://www.google.com/speech-api/v2/recognize với các giá trị dưới đây làm tham số URL:

- client=*chromium*
- lang=*language_choice*
- key=*a_developer_key*

Trong đó *language_choice* là từ khóa viết tắt cho từng ngôn ngữ

Ví dụ để yêu cầu chuyển đổi ngôn ngữ là tiếng anh, chúng ta đặt giá trị lang=en_US, đối với tiếng Việt, chúng ta đặt giá trị lang=vi_VN.

Giá trị *a_developer_key* là một đoạn mã được Google cung cấp cho các Developer khi khởi tạo một dự án ở nền tảng đám mây Google. Từ đây chúng ta có thể kích hoạt và bắt đầu sử dụng các dịch vụ như Google Speech API.

Tiêu đề HTTP Request

- Content-Type: trong dự án này chúng ta sẽ để giá trị là “audio/x-flac; rate=44100;”

Và cuối cùng, dữ liệu trong thân HTTP Request là dữ liệu tập tin âm thanh dưới dạng mã hóa flac. Tập tin này được ghi âm bởi micro với tần số lấy mẫu 44100Hz và mã hóa 32 bit

❖ Nhận phản hồi từ Google Speech API

Khi yêu cầu được gửi tới Google Speeech API thành công, Google sẽ trả về một đoạn mã JSON có cấu trúc như sau đây với kết quả nhận diện giọng nói ở trường *transcript*

```
{
   "result":[
      {
         "alternative":[
            {
               "transcript":"good morning Google how are you feeling today"
            }
         ],
         "final":true
      }
   ],
   "result_index":0
}
```

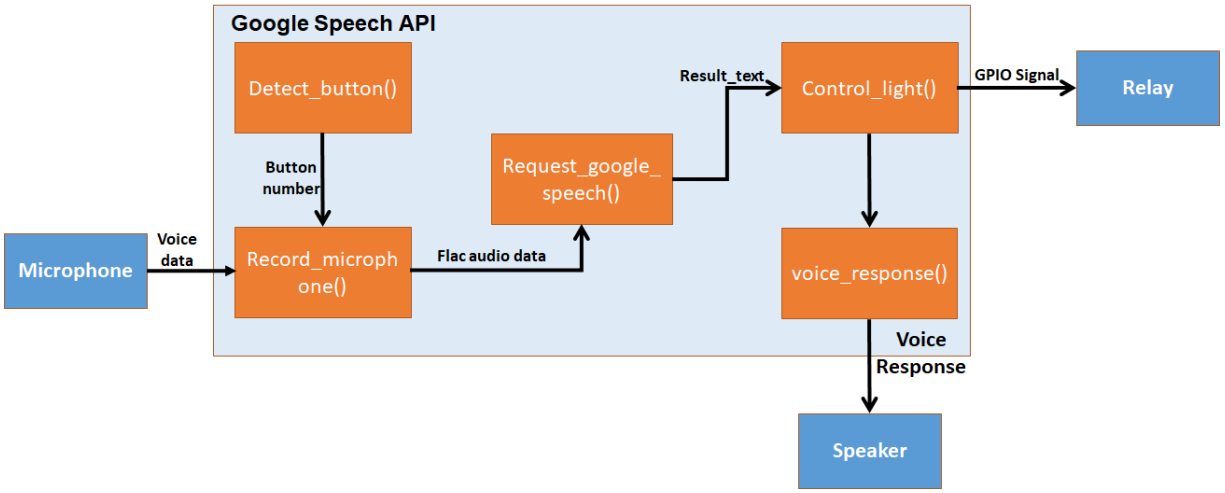

Khi có nhiễu hoặc có nhiều kết quả được chuyển đổi, Google sẽ trả về thêm một số trường ở trong đoạn mã JSON


```
{
  "result":[
    {
      "alternative":[
        {
          "transcript":"this is a test",
          "confidence":0.97321892
        },
        {
          "transcript":"this is a test for"
        }
      ],
      "final":true
    }
  ],
  "result_index":0
}
```


Kết quả trả về sẽ được chúng ta xử lý và đưa ra hành động phù hợp với lệnh điều khiển của người dùng

4.5.3 *Thiết kế chi tiết*

Để sử dụng Google Speech API, chúng ta sử dụng 2 nút bấm để kích hoạt chương trình. Một nút bấm cho việc ghi âm và gửi yêu cầu câu lệnh tiếng Anh, một nút bấm cho việc ghi âm và gửi yêu cầu câu lệnh bằng tiếng Việt và thực hiện theo các bước sau đây:



Hình 4.26. Thiết kế chi tiết khối xử lý giọng nói sử dụng Google Speech API

- Khi người dùng bấm nút, chương trình sẽ phát hiện và gọi hàm *record_microphone()* để ghi âm, người dùng có 4s để thực hiện câu lệnh của mình (tắt/bật đèn).
- Chương trình sẽ ghi âm và gửi yêu cầu tới Google Speech API bằng cách truyền tập tin âm thanh flac vào hàm *Request_google_speech()*, nếu thành công sẽ trả mã JSON về.
- Sau khi gửi yêu cầu và được phản hồi thành công, hàm sẽ truyền kết quả tới hàm Control_light() để chương trình phân tích so sánh với các mẫu câu có sẵn để đưa ra hành động phù hợp
- Hàm control_light sẽ gửi tín hiệu tới chân GPIO relay tương ứng với đèn các phòng và chạy các tập tin âm thanh phản hồi bằng cách gọi hàm voice_response()

4.6 Khối xử lý điều khiển qua mạng LAN

Trong khối này em sử dụng OpenHAB là phần mềm miễn phí nguồn mở có chức năng làm bộ điều khiển trung tâm để giao tiếp với rất nhiều các loại thiết bị khác nhau (kể cả các thiết bị thương mại) của rất nhiều hãng sản xuất vào trong một hệ thống Smart Home hoàn chỉnh.

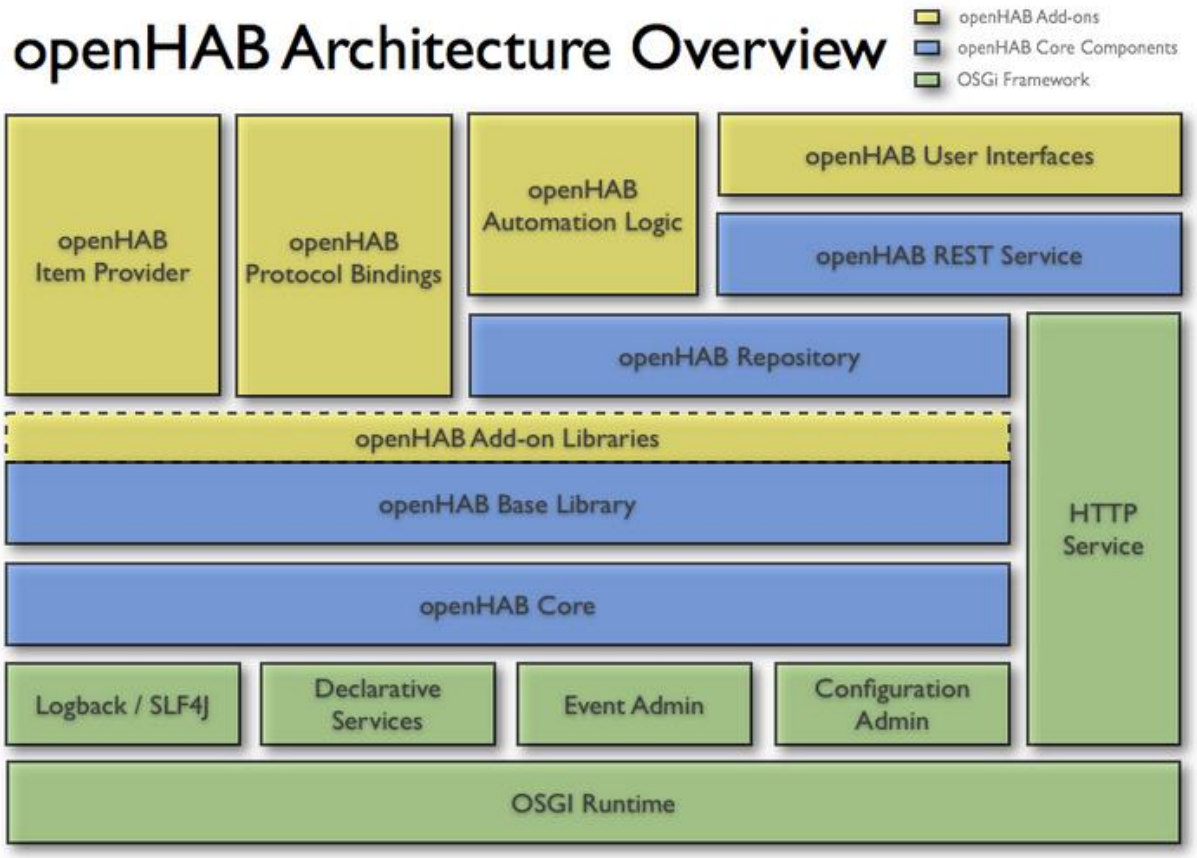
4.6.1 ***Tìm hiểu OpenHAB***

4.6.1.1 *OpenHAB là gì*

Openhab là một phần mềm mã nguồn mở có chức năng là bộ điều khiển trung tâm, với khả năng kết nối giao tiếp và điều khiển tới nhiều loại thiết bị khác nhau trong hệ thống SmartHome. OpenHab cung cấp nhiều giao diện người dùng (trang web, ứng dụng android, ứng dụng ios….) giúp cho quá trình làm việc với OpenHab dễ dàng và thuận tiện hơn. Diễn giải theo quan điểm cá nhân: OpenHab như một phần mềm quản lí, nó cung cấp các tiện ích giúp hỗ trợ việc kết nối các thiết bị phần cứng vào một cách dễ dàng nhất (tuy nhiên bị giới hạn trong các phần cứng mà nó hỗ trợ). Sau khi kết nối, thì OpenHab và các thiết bị sẽ giao tiếp vỡi nhau, và OpenHab sẽ có được thông tin trạng thái gửi từ các thiết bị, cùng với đó, các thiết bị có thể nhận được các lệnh điều khiển từ

OpenHab. OpenHAB 2 được dựa trên mã nguồn mở Eclipse SmartHome và được hoàn toàn viết bằng Java. OpenHAB có thể cài đặt trên nhiều nền tảng như Windows, Linux, Mac OSX và các nền tảng nhúng như Raspberry Pi, BeagleBone Black, UDOO, Cubietruck, do đó bạn có thể cài đặt trên Raspberry Pi

4.6.1.2 *Cấu trúc OpenHAB*



Hình 4.27. Cấu trúc Openhab

OpenHab Runtime được triển khai dựa trên OSGi Framework (nền tảng này vận hành theo cơ chế pub/sub), nó sử dụng nền tảng Java solution và cần JVM để chạy. Dựa trên OSGi, OpenHab tích hợp các kiến trúc mô-đun hóa, và tích hợp rất nhiều thành phần vào nhau

Chúng ta có thể giao tiếp với openHAB thông qua Event bus. Đây là là dịch vụ lõi của OpenHAB, có nhiệm vụ thông báo cho các thành phần khác trong hệ thống về các sự kiện và cập nhật từ các gói bên ngoài. Có 2 kiểu Event chính:

- Câu lệnh để thực hiện các hành động, hoặc thay đổi trạng thái của item/device
- Cập nhật trạng thái và thông áo về thay đổi trạng thái của một số thiết bị.

Các giao thức binding (kết nối tới phần cứng) đều giao tiếp qua Event Bus.

Trong đề tài này, em sử dụng giao thức MQTT để giao tiếp qua Event Bus để cập nhật trạng thái ngôi nhà cũng như gửi lệnh điều khiển các thiết bị từ giao diện máy tinh và điện thoại của người dùng

4.6.1.3 *Các thành phần cấu hình openHAB*

(a) Things

Things là các thực thế được thêm vào hệ thống và có khả năng nhiều chức năng. Cần phải chú ý rằng Things không phải là thiết bị, nhưng có thể đại diện cho một dịch vụ web hay bất kỳ một nguồn thông tin và chức năng có thể quản lý khác. Chúng có liên quan tới quá trình thiết lập và cấu hình, nhưng không cho quá trình hoạt động

(b) Items

Items là các thực thể đại diện cho các thành phần sẽ hiển thị trong giao diện cũng như trong cài đặt. Chúng sẽ có trạng thái và được sử dụng qua các sự kiện, chúng có thể được đọc hoặc thay đổi qua tương tác. Các items có thể là các thiết bị trong nhà như bóng đèn, quạt… hay các thông số của nhiệt độ, độ ẩm….

Để khai báo Items chúng ta có thể thiết lập ở trong các tệp tin có đuôi .items ở trong thư mục items của openhab.

Cú pháp cấu hình:

itemtype itemname ["labeltext"] [<iconname>] [(group1, group2, ...)] [["tag1", "tag2", ...]]

Ví dụ: *Number LivingRoom_Temperature "The Temperature is [%.1f °C]" <temperature> (gTemperature, gLivingRoom) ["TargetTemperature"]*

Diễn giải các thành phần trong cấu hình items

- *itemtype*: Loại item được dùng trong openhab (number, switch, dimmer….)
- *itemname:* Tên của Item để có thể sử dụng trong cấu hình
- *["labeltext"]:* chữ hiển thị của item trên giao diện
- *[(group1, group2, ...)]:* tên nhóm của item

(c) Sitemaps

Trong openHAB, một tập hợp của Things và Items là đại diện của các thực thể vật lý và logic theo cài đặt ngôi nhà của người dùng. *Sitemaps* được sử dụng để chọn

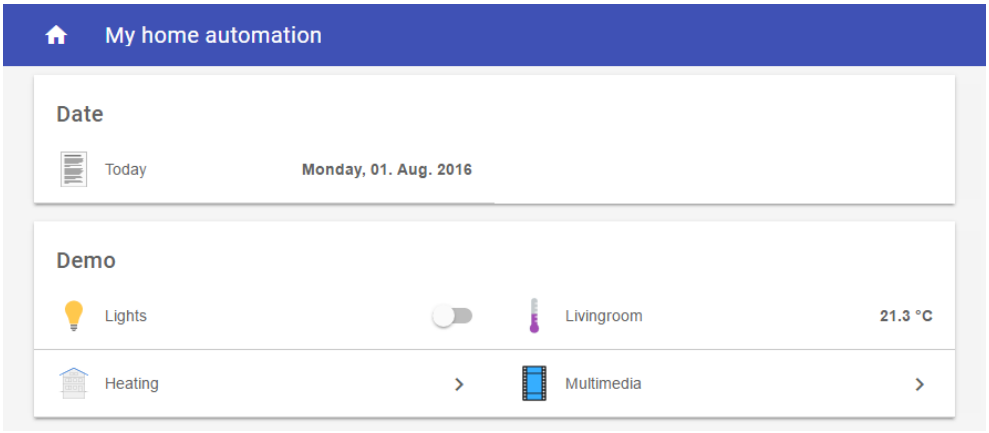
và liên kết các thực thể với nhau để hiển thị trên các giao diện sử dụng theo định hướng của người dùng.

Người dùng có thể định nghĩa *Sitemaps* theo một cú pháp rõ ràng trong một tệp có đuôi .sitemap và lưu trong thư mục *openhab*. Các *Things* và *Items* đã được khai báo sẽ được liên kết với nhau để hiển thị một cách khoa học trên UI hiển thị cho người dùng.

Ví dụ chúng ta có tập tin demo.sitemap sau đây:

```
sitemap demo label="My home automation" {
    Frame label="Date" {
        Text item=Date
    }
    Frame label="Demo" {
        Group item=Heating
        Switch item=Lights icon="big_bulb" mappings=[OFF="All Off"]
        Text item=Temperature
valuecolor=[>25="orange",>15="green",<=15="blue"]
        Text item=Multimedia_Summary label="Multimedia" icon="video" {
            Selection item=TV_Channel mappings=[0="off", 1="DasErste",
2="BBC One", 3="Cartoon Network"]
            Slider item=Volume
        }
    }
}
```

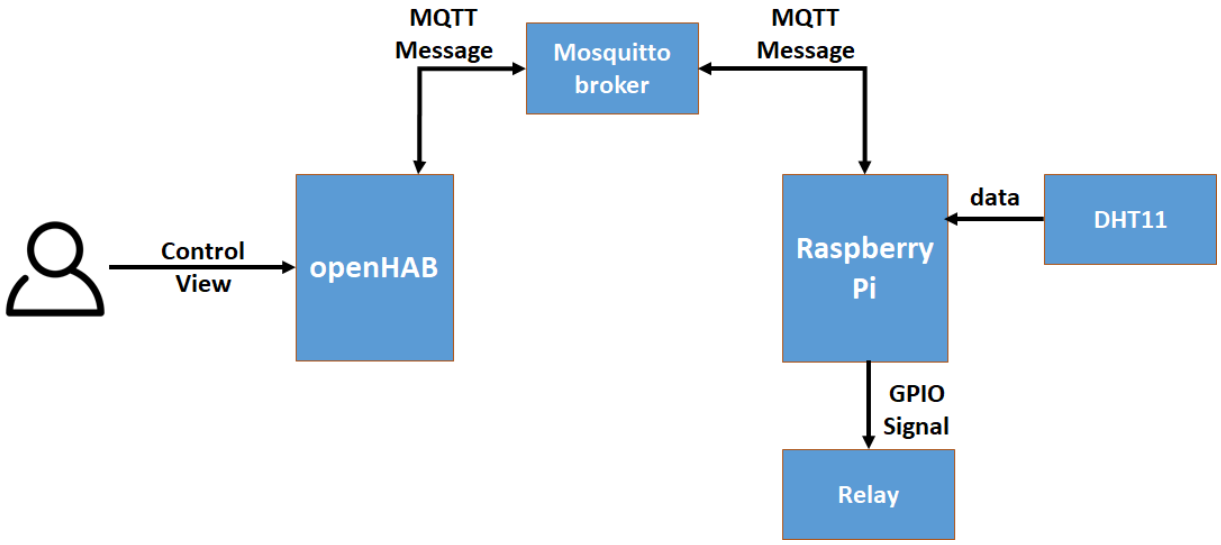
Chúng ta sẽ được giao diện sau khi cấu hình như Hình 4.28



Hình 4.28 Giao diện khi cấu hình

4.6.2 *Thiết kế chi tiết*

Trong đề tài này em sử dụng giao thức MQTT để điều khiển thiết bị điện, cập nhật trạng thái môi trường các phòng.



Hình 4.29. Sơ đồ thiết kế khối xử lý điều khiển qua mạng LAN

Trong sơ đồ Hình 4.29 trên, *Mosquitto broker* là một trung gian cho phép ta cài đặt và thực thi giao thức MQTT. Đây là một mã nguồn mở và có thể cài đặt, sử dụng trên Raspberry Pi để làm cầu nối trung gian giữa openHAB và bộ xử lý trung tâm để tiếp nhận, xử lý và chuyển tiếp các bản tin đến các thiết bị trong hệ thống của chúng ta thông qua các kênh. Sau khi cài đặt, Mosquitto sẽ tự khởi chạy và sử dụng cổng 1883.

Khối điều khiển qua mạng LAN có 2 chức năng chính:

- Cập nhật nhiệt độ, độ ẩm: ở trong openHAB đã cấu hình địa chỉ của các *items* nhiệt độ, độ ẩm của các phòng theo một kênh *topic* nhất định, nhiệm vụ của Raspberry Pi là đọc nhiệt độ từ DHT11 và *publish* lên *topic* tương ứng mà openHAB đã *subscribe*
- Điều khiển bật tắt thiết bị điện: người dùng ra lệnh điều khiển thông qua giao diện điện thoại hoặc trang web, openHAB sẽ *publish* một bản tin với *payload* chứa lệnh (ON hoặc OFF) tới topic, Raspberry Pi đã đăng ký sẵn topic này và tạo tín hiệu GPIO để điều khiển.

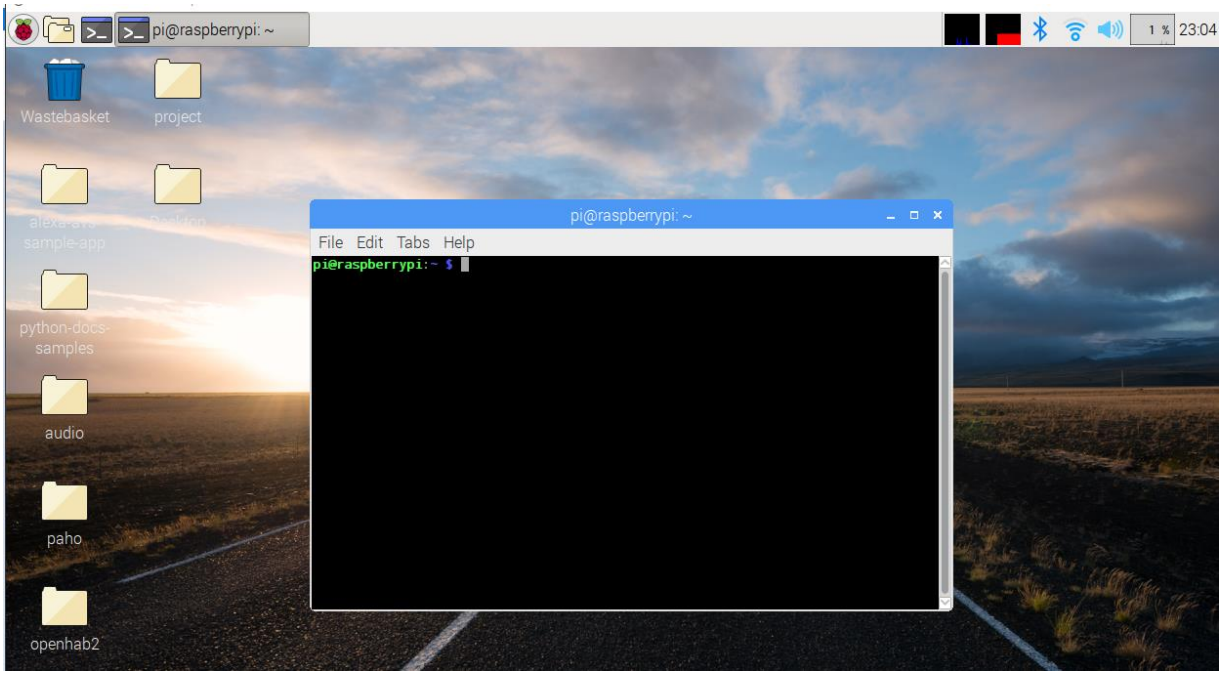
CHƯƠNG 5. TRIỂN KHAI VÀ KIỂM THỬ

5.1 Triển khai

5.1.1 *Cài đặt các công cụ cần thiết*

5.1.1.1 *Cài đặt hệ điều hành Raspbian*

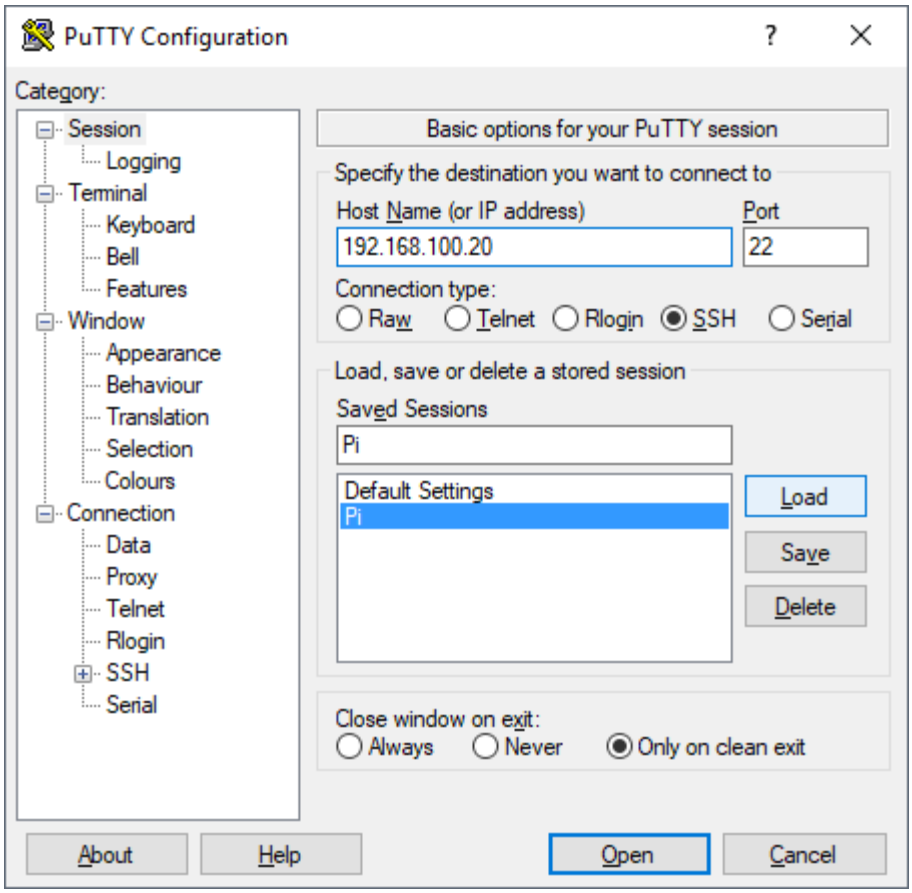
Như đã trình bày ở chương thiết kế chi tiết, chúng ta tiến hành cài đặt hệ điều hành Raspbian Jessie cho Raspberry Pi 3. Sau khi cài đặt chúng có được giao diện và terminal làm việc như hình



Hình 5.1. Hệ điều hành Raspbian và terminal làm việc

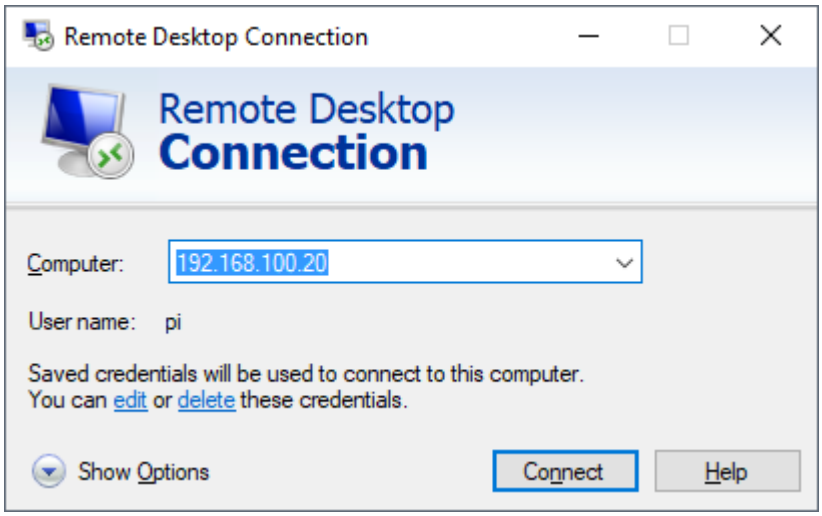
Một vấn đề đặt ra khi không có sẵn màn hình, bàn phím hoặc chuột để sử dụng Raspberry Pi, ta có thể kết nối Raspberry Pi với máy tính chạy Windows thông qua SSH. Đây là một giao thức mạng dùng để thiết lập một kết nối bảo mật giữa hai thiết bị cùng lớp mạng. Phần mềm được sử dụng là Putty.

Để SSH tới Raspberry Pi, ta khởi động Putty và điền IP của Raspberry Pi vào ô Host Name của phần mềm rồi chọn Open. Ở đây em đặt địa chỉ IP cố định cho Raspberry Pi là 192.168.100.20 cổng 22



Hình 5.2. Phần mềm Putty

Ngoài ra, ta có thể dùng Remote Desktop Connection có sẵn trên Windows để điều khiển từ xa Raspberry Pi trên giao diện đồ họa.



Hình 5.3. Phần mềm Remote Desktop Connection

Khởi động Remote Desktop Connection từ Windows, điền địa chỉ IP của Raspberry Pi vào ô Computer, sau đó bấm Connect. Cửa sổ xuất hiện yêu cầu tên đăng nhập và mật khẩu. Ta sử dụng tên đăng nhập và mật khẩu mặc định của Raspberry Pi là pi/raspberry.

5.1.1.2 *Cài đặt các thư viện python*

❖ Cài đặt MQTT Broker Mosquitto

```
wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
sudo apt-key add mosquitto-repo.gpg.key
sudo wget http://repo.mosquitto.org/debian/mosquitto-wheezy.list
sudo apt-get update
sudo apt-get install mosquitto mosquitto-clients python-mosquitto
```

❖ Cài đặt thư viện MQTT paho

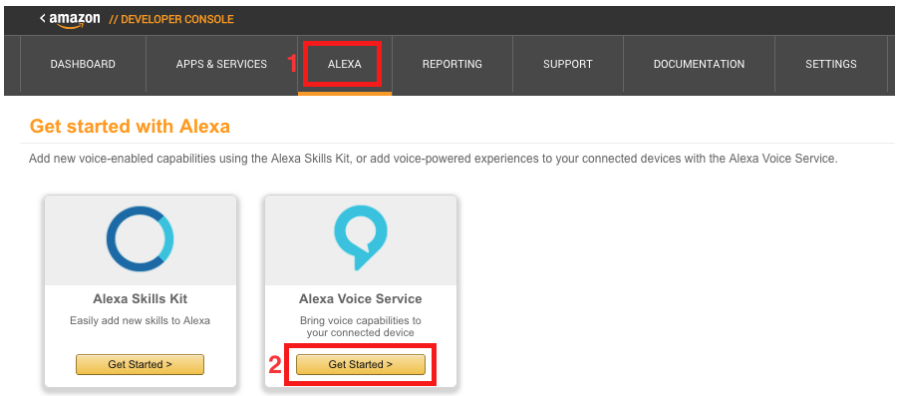
```
git clone https://github.com/eclipse/paho.mqtt.python
cd paho.mqtt.python
python setup.py install
```

❖ Cài đặt thư viện AWS IoT Device SDK để có thể làm việc với Thing Shadow của AWS IoT

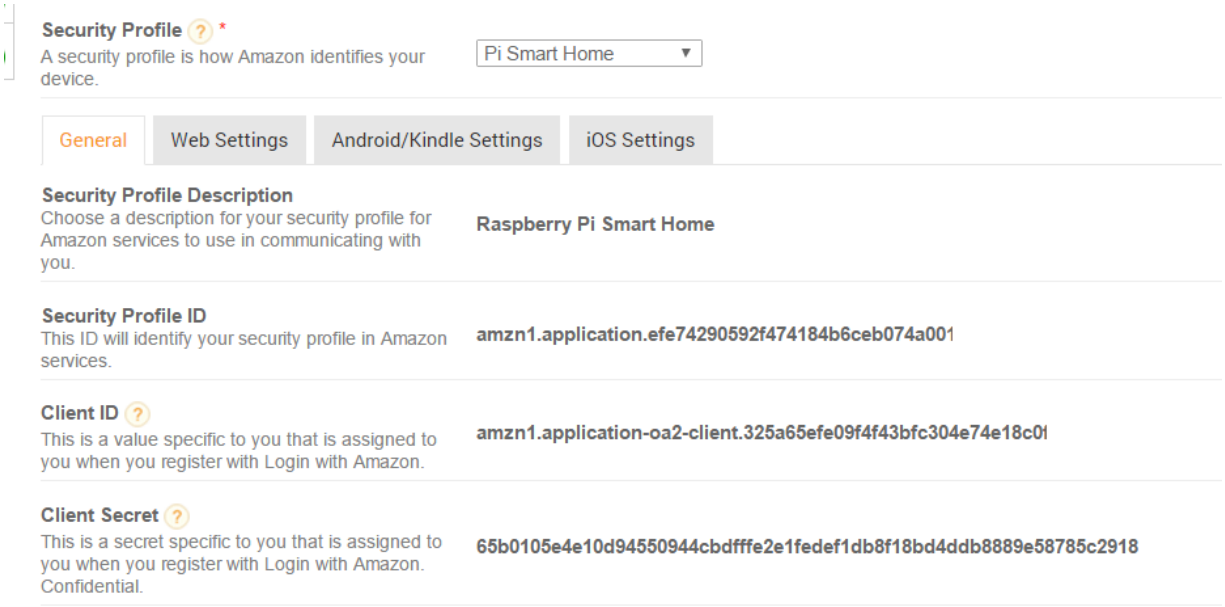
```
git clone https://github.com/aws/aws-iot-device-sdk-python.git
cd aws-iot-device-sdk-python
python setup.py install
```

5.1.1.3 *Cài đặt ứng dụng của Alexa Voice Service*

❖ Bước 1: Trước tiên chúng ta phải có tài khoản Amazon Developer và tạo thiết bị, các mã định danh cần thiết



Hình 5.4 Đăng ký dịch vụ Alexa Voice Service



Hình 5.5. Khởi tạo thiết bị và các mã định danh cần thiết

Bước 2: Sao chép ứng dụng mẫu từ GitHub và cài đặt từ terminal

```
cd Desktop
git clone https://github.com/alexa/alexa-avs-sample-app.git
cd ~/Desktop/alexa-avs-sample-app
nano automated_install.sh
```

Chương trình sẽ tiến hành cài đặt các phần mềm cần thiết và mất khoảng 30 phút làm việc.

Bước 3: Khởi chạy ứng dụng

Để khởi chạy ứng dụng, chúng ta cần phải mở 3 cửa sửa terminal để chạy các chương trình cần thiết

- Terminal 1: Chạy dịch vụ web cho việc xác thực và ủy quyền giữa ứng dụng mẫu với AVS

```
cd ~/Desktop/alexa-avs-sample-app/samples
cd companionService && npm start
```

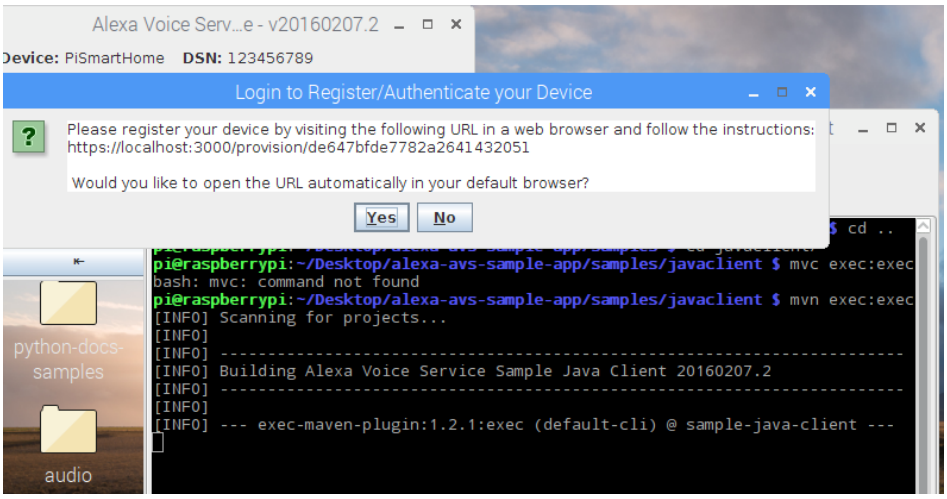
- Terminal 2: Chạy ứng dụng mẫu để giao tiếp với AVS

```
cd ~/Desktop/alexa-avs-sample-app/samples
cd javaclient && mvn exec:exec
```

- Terminal 3: Mở công cụ từ đánh thức để có thể tương tác với ứng dụng bằng cụm từ “Alexa” :

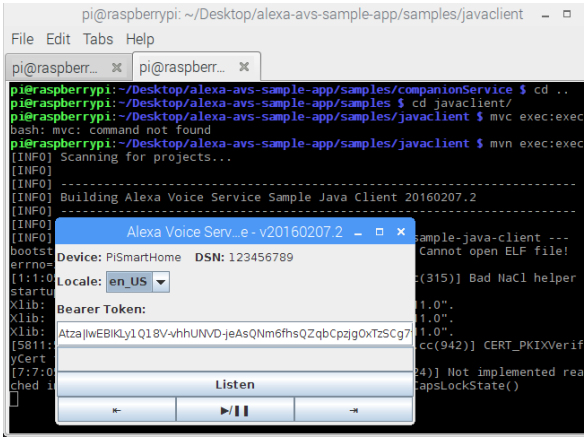
```
cd ~/Desktop/alexa-avs-sample-app/samples
cd wakeWordAgent/src && ./wakeWordAgent -e sensory
```

Khi ứng dụng khởi động, nó sẽ yêu cầu mở trình duyệt web để xác thực với AVS và nhận mã token cho ứng dụng có thể tích hợp với AVS



Hình 5.6. Ứng dụng yêu cầu mở trình duyệt web để xác thực với AVS

Sau khi nhận mã token từ AVS, chúng ta có thể gọi ứng dụng hoạt động bằng cách sử dụng từ đánh thức Alexa hoặc bấm chọn Listen để ra lệnh điều khiển thông qua microphone của Raspberry Py



Hình 5.7. Ứng dụng sau khi đã nhận mã token có thể hoạt động

5.1.1.4 *Cài đặt openHAB*

Tải và cài đặt bản mới nhất của openHAB

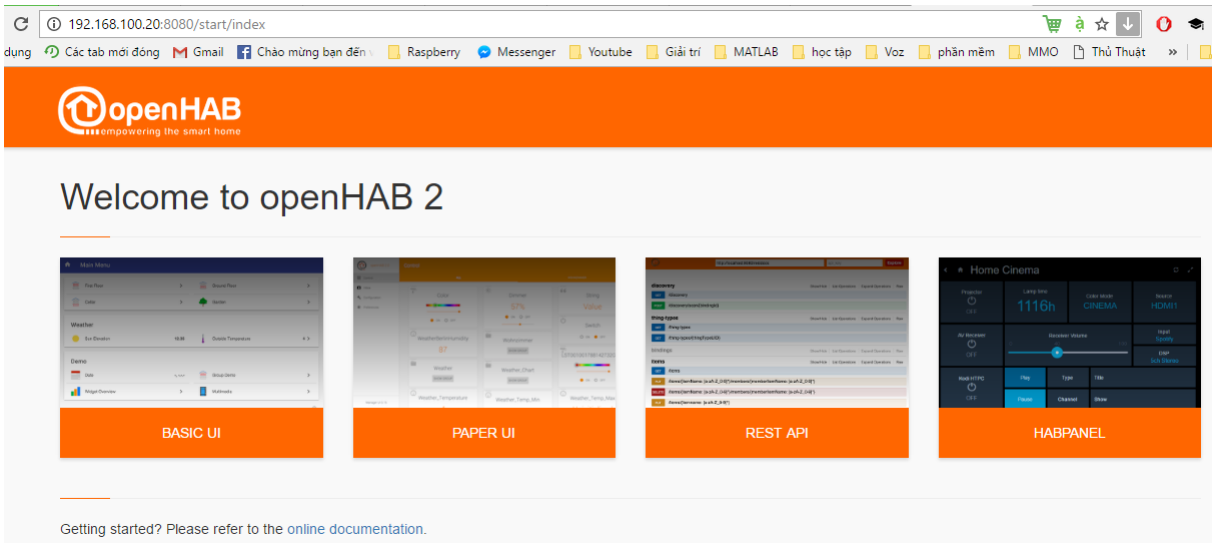
```
sudo apt-get install openhab2-offline
```

cấu hình để openHAB luôn chạy khi Raspberyr Pi khởi động

```
sudo /etc/init.d/openhab2 start
sudo /etc/init.d/openhab2 status
```

```
sudo update-rc.d openhab2 defaults
```

Sau khi cài đặt thành công, chúng ta có thể truy cập openHAB để cấu hình theo địa chỉ sau http://192.168.100.20:8080 trong đó 192.168.100.20 là địa chỉ IP của Raspberry Pi được đặt sẵn



Hình 5.8. Giao diện openHab khi chúng ta truy cập lần đầu

Do phạm vi đề tài nên chúng ta chỉ đi cấu hình mục BASIC UI bằng cách tạo các tập tin *items, sitemaps*

❖ Cấu hình *items*

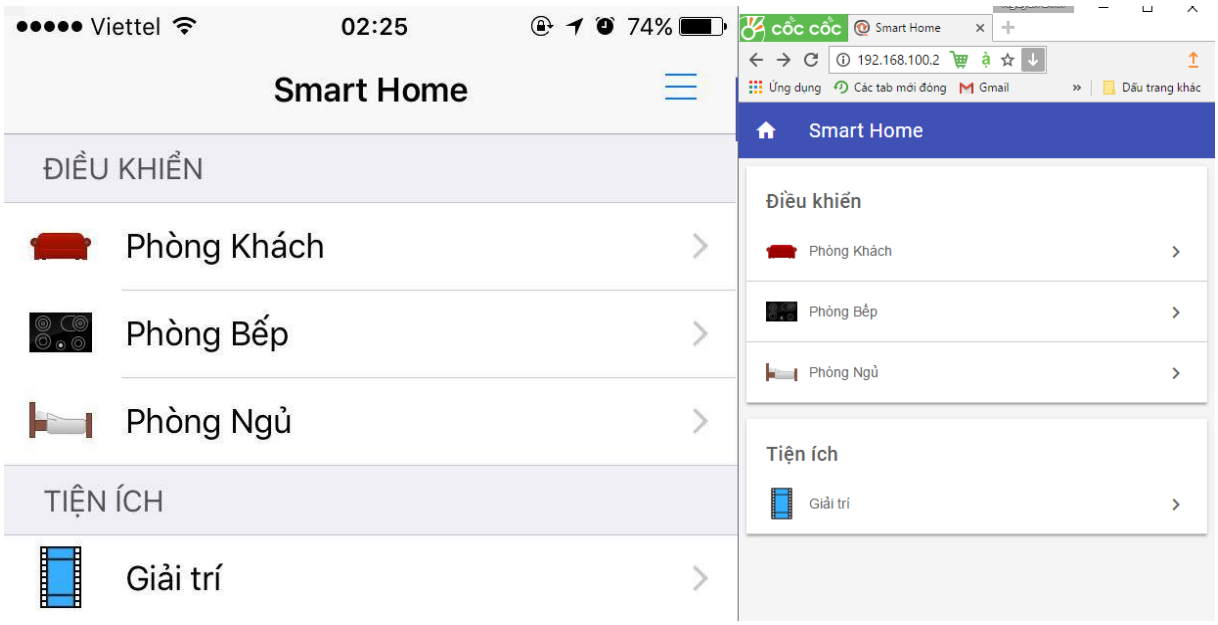
```
Group All
Group Living "Phòng Khách" <Living> (All)
Group Kitchen "Phòng Bếp" <Kitchen> (All)
Group Bedroom "Phòng ngủ" <Bedroom> (All)
//Living room
Number Temperature_Living "Nhiệt độ[%.1f C]" <temperature> (Living) {mqtt=
"<[mymosquitto:/Living/Temperature:state:default]"}
Number Humidity_Living "Độ ẩm [%.1f]" <humidity> (Living)
{mqtt="<[mymosquitto:/Living/Humidity:state:default]"}
Switch Light_Living "Đèn phòng khách" <light> (Living) {
mqtt=">[mymosquitto:/Living/Light:command:off:OFF],>[mymosquitto:/Living/Light:comma
nd:on:ON],<[mymosquitto:/Living/Light/Status:state:default]"}
//Kitchen
Number Temperature_Kitchen "Nhiệt độ [%.1f C]" <temperature> (Kitchen)
{mqtt="<[mymosquitto:/Kitchen/Temperature:state:default]"}
Number Humidity_Kitchen "Độ ẩm [%.1f]" <humidity> (Kitchen)
{mqtt="<[mymosquitto:/Kitchen/Humidity:state:default]"}
Switch Light_Kitchen "Đèn phòng bếp" <light> (Kitchen) {
mqtt=">[mymosquitto:/Kitchen/Light:command:off:OFF],>[mymosquitto:/Kitchen/Light:com
mand:on:ON],<[mymosquitto:/Kitchen/Light/Status:state:default]"}
//Main Bedroom
```

```
Number Temperature_Bedroom "Nhiệt độ [%.1f C]" <temperature> (Bedroom)
{mqtt="<[mymosquitto:/Bedroom/Temperature:state:default]"}
Number Humidity_Bedroom "Độ ẩm [%.1f]" <humidity> (Bedroom)
{mqtt="<[mymosquitto:/Bedroom/Humidity:state:default]"}
Switch Light_Bedroom "Đèn phòng ngủ" <light> (Bedroom) {
mqtt=">[mymosquitto:/Bedroom/Light:command:off:OFF],>[mymosquitto:/Bedroom/Light:com
mand:on:ON],<[mymosquitto:/Bedroom/Light/Status:state:default]"}
//Location
Location BkLocation           "Đại học Bách Khoa Hà Nội"
```

❖ Cấu hình sitemaps

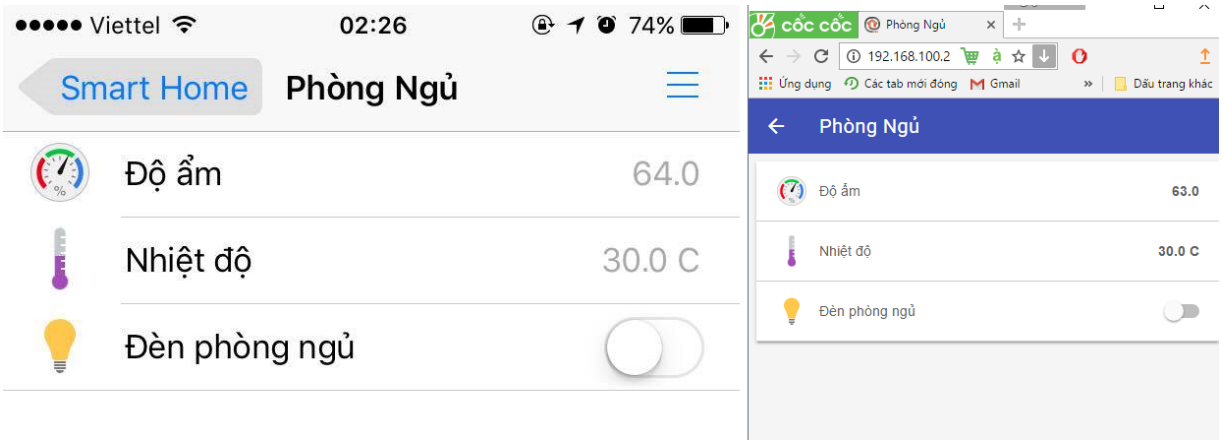
```
sitemap default label="Smart Home"
{
Frame label="Điều khiển" {
       Group item=Living label="Phòng Khách" icon="sofa"
       Group item=Kitchen label="Phòng Bếp" icon="kitchen"
       Group item=Bedroom label="Phòng Ngủ" icon="bedroom"
}
Frame label="Tiện ích" {
Text label="Giải trí" icon="video" {
       Text label="Đọc báo" icon="video" {
              Frame label="Đọc báo" {
                     Webview url="http://dantri.com.vn/" height=8

              }
       }
}
```

Sau khi cấu hình hoàn chỉnh, chúng ta có được giao diện khi mở BASIC UI trên trình duyệt web và mở ứng dụng openHAB trên điện thoại. Chúng ta sẽ quản lý 3 phòng và sẽ phát triển thêm các tiện ích giải trí khác



Hình 5.9. Giao diện khi khởi động

Giao diện khi mở phòng ngủ khi mở BASIC UI trên trình duyệt web và mở ứng dụng openHAB trên điện thoại. Thông số độ ẩm, nhiệt độ và trạng thái đèn được hiển thị ở đây.



Hình 5.10. Giao diện khi mở phòng ngủ

5.1.2 ***Thiết kế mô hình***

Trong đề tài này, em sử dụng vật liệu Fomex để làm mô hình với các đặc điểm trọng lượng nhẹ, chống thấm nước và không độc hại.

5.1.2.1 *Hộp đựng cho bộ xử lý trung tâm – Raspberry Pi 3*

Hình 5.11 Hộp đựng bảo vệ cho bộ xử lý trung tâm

Hộp đựng Raspberry pi được đặt chung với hệ thống dây điện và nguồn 220V

Hình 5.12. Ngăn chứa các thành phần của khối xử lý trung tâm và khối nguồn

5.1.2.2 *Microphone và Loa*

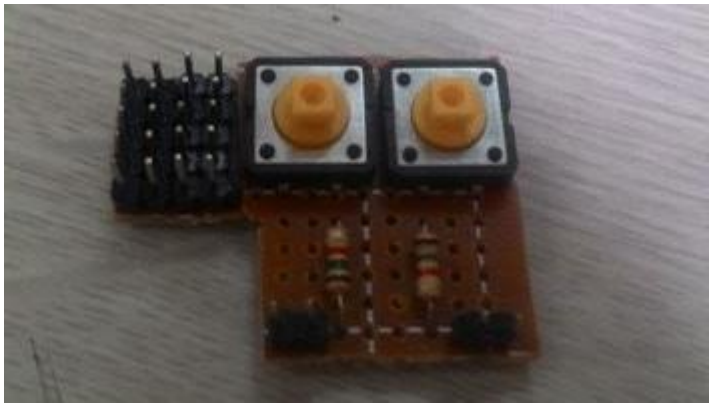
❖ Microphone USB 2.0 được sử dụng để làm đầu vào thu âm thanh người dùng. Giá 160.000 VNĐ 1 sản phẩm, được đặt ở phía trước mô hình ngôi nhà

Hình 5.13. Microphone USB và vị trí đặt trong mô hình

❖ Sử dụng Loa Mini USB Loyfun để phát âm thanh từ Raspberry Pi 3 với giắc 3.5mm và giắc USB cấp nguồn 5V từ Raspberry Pi. Giá thành 140.000 VNĐ 1 sản phẩm, được đặt ở phòng khách của mô hình

Hình 5.14 Loa USB và vị trí đặt trong mô hình

5.1.2.3 *Mạch 2 nút bấm cho khối xử lý giọng nói – sử dụng Google Speech API*



Hình 5.15. Mạch nút bấm khối xử lý giọng nói – sử dụng Google Speech API

5.1.2.4 *Mô hình ngôi nhà*

Mô hình ngôi nhà được thiết kế bằng vật liệu fomex với kích thước 40x40cm được chia cho 3 phòng khách, phòng ngủ, phòng bếp, ngăn chứa bộ xử lý trung tâm và khối nguồn. Mỗi phòng được bao gồm 1 bóng đèn 12W – 220V và một cảm biến nhiệt độ DHT11

Hình 5.16. Mô hình ngôi nhà sau khi hoàn thành

5.1.2.5 *Giá thành sản phẩm*

Sau khi hoàn thành mô hình, sản phẩm có giá thành tổng cộng 2.210.000 VNĐ, đây là chi phí không quá đắt và cũng không vượt qua giá bán sản phẩm đề xuất ban đầu. Chi tiết các thành phần được liệt kê trong Bảng 5.1

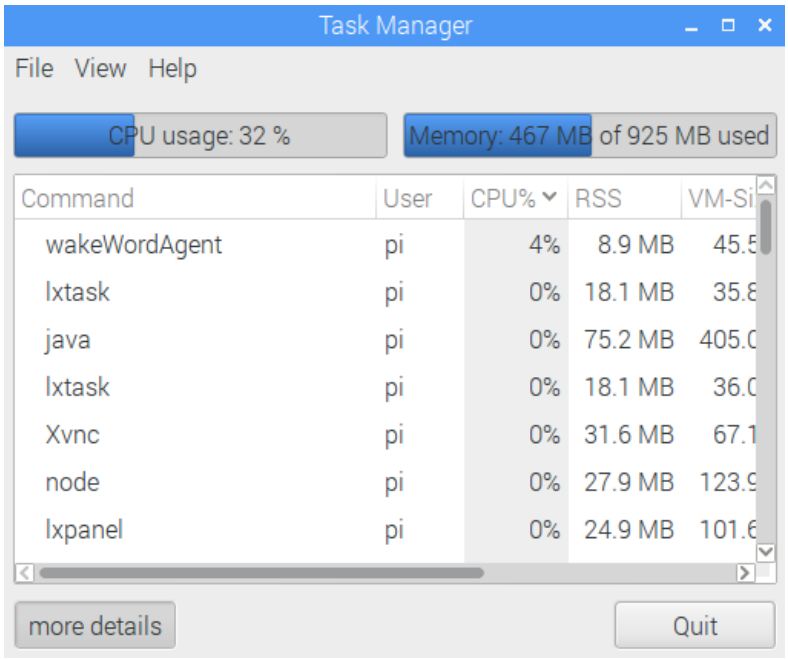
STT	Tên	Số lượng	Đơn giá	Giá thành
1	Raspberyr Pi 3 + Nguồn + thẻ nhớ	1	1.550.000 đ	1.550.000 đ
2	Module Cảm biến DHT11	3	40.000 đ	120.000 đ
3	Bộ bóng đèn	3	20.000 đ	60.000 đ
4	Microphone	1	160.000 đ	160.000 đ
5	Loa	1	140.000 đ	140.000 đ
6	Bộ nguồn 220V	1	80.000 đ	80.000 đ
7	Nguyên liệu mô hình	1	100.000 đ	100.000 đ
Tổng cộng				2.210.000 đ

Bảng 5.1 Giá thành bộ sản phẩm

5.2 Kiểm thử

5.2.1 ***Hiệu năng***

Sau khi hoàn thành cài đặt các công cụ và thiết kế mô hình, em tiến hành thử nghiệm các tính năng. Khi cho triển khai các chương trình chạy trong đề tài, chiếm 467MB/925Gb Ram và tiêu thụ 32% CPU



Hình 5.17. Lượng tài nguyên tiêu thụ khi chạy chương trình

5.2.2 *Các chức năng*

5.2.2.1 *Chức năng nhận dạng giọng nói sử dụng Alexa Voice Service*

Với chức năng này, người dùng có thể sử dụng các câu lệnh tiếng anh

- "Alexa, open Smart Home": trả về lời chào mừng người dùng, giới thiệu các tính năng của chương trình
- "Alexa, Turn on living room / bedroom / kitchen light": Ra lệnh bật đèn các phòng tương ứng, chương trình sẽ kiểm tra trạng thái hiện tại của đèn để điều khiển và phản hồi một cách chính xác. Ví dụ đèn đang tắt thì sẽ được bật, phản hồi bật đèn thành công, nhưng nếu đèn đang bật thì sẽ phản hồi với nội dung đèn đã được bật sẵn rồi.
- "Alexa, Turn off living room / bedroom / kitchen light": Tương tự như trên người dùng ra lệnh tắt đèn các phòng, chương trình sẽ kiểm tra trạng thái hiện tại của đèn để điều khiển và phản hồi một cách chính xác. Ví dụ đèn đang bật thì sẽ được tắt, phản hồi bật đèn thành công, nhưng nếu đèn đang tắt thì sẽ phản hồi với nội dung đèn đã được tắt sẵn rồi.
- "Alexa, temperature / humidity, living room / bedroom / kitchen light": Ra lệnh để AVS trả về giá trị nhiệt độ, độ ẩm của các phòng tương ứng

Việc điều khiển được thử nghiệm từ ba người, 2 người nam quê ở Nghệ An và 1 người nữ quê ở Quảng Ninh. Mỗi người đọc 50 lần để ra lệnh tương ứng với các câu lệnh trên được kết quả thử nghiệm sau đây

STT	Người Thực hiện	Số Lần	Đúng	Sai
1	Tâm	50	47	3
2	Bình	50	43	7
3	Hạnh	50	44	6
Tổng		150	134	16

Bảng 5.2 Kết quả thử nghiệm điều khiển bằng giọng nói tiếng Anh

Kết quả đúng 134/150 lần, đạt 89%, đây là một kết quả khả quan bởi do việc thử nghiệm sử dụng tiếng Anh có thể phát âm chưa chuẩn hay do lỗi về đường truyền cũng như chương trình vẫn chưa được tối ưu.

5.2.2.2 *Chức năng nhận dạng giọng nói sử dụng Google Voice API*

Với chức năng này, người dùng có thể sử dụng các câu lệnh tiếng Việt khi bấm nút.

- "Bật đèn phòng khách / bếp / ngủ": Chương trình sẽ bật đèn tương ứng ở các phòng và phản hồi thành công. Nếu đèn đã bật sẵn, chương trình sẽ chỉ thông báo đèn đã được bật rồi.
- "Tắt đèn phòng khách / bếp / ngủ": Chương trình sẽ tắt đèn tương ứng ở các phòng và phản hồi thành công. Nếu đèn đã tắt sẵn, chương trình sẽ chỉ thông báo đèn đã được tắt rồi.
- "Tắt / Bật tất cả đèn: Chương trình sẽ tắt /bật tất cả các đèn trong các phòng và thông báo

Việc điều khiển được thử nghiệp từ ba người, 2 người nam quê ở Nghệ An và 1 người nữ quê ở Quảng Ninh. Mỗi người đọc 50 lần để ra lệnh tương ứng với các câu lệnh trên được kết quả thử nghiệm sau đây

STT	Người Thực hiện	Số Lần	Đúng	Sai
1	Tâm	50	43	7
2	Bình	50	41	9
3	Hạnh	50	40	10
Tổng		150	124	26

Bảng 5.3 Kết quả thử nghiệm điều khiển bằng giọng nói tiếng Việt

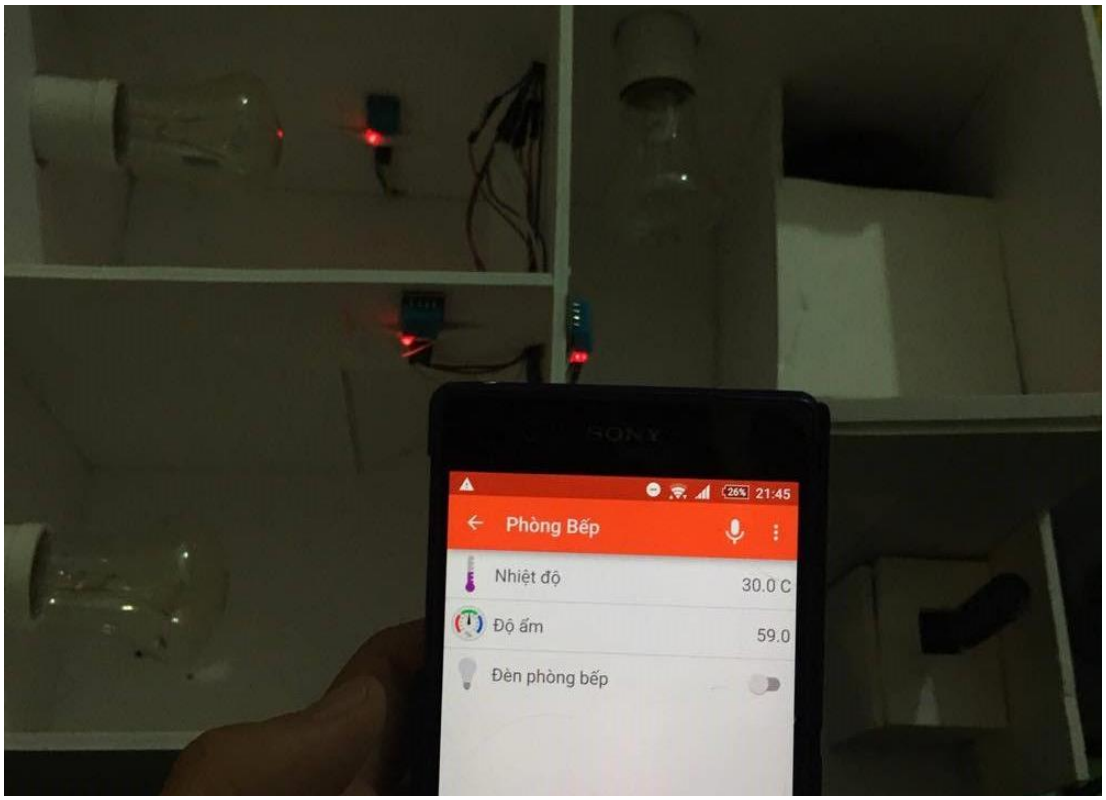
Kết quả đúng 129/150 lần, đạt 83%, đây là một kết quả khả quan vì việc nhận dạng giọng nói tiếng Việt là rất khó, ngữ âm tiếng Việt có sự khác nhau về các vùng miền, ngoài ra còn khó khăn về đường truyền hay thao tác thử nghiệm không được chính xác.

5.2.2.3 *Chức năng điều khiển qua máy tính, điện thoại*

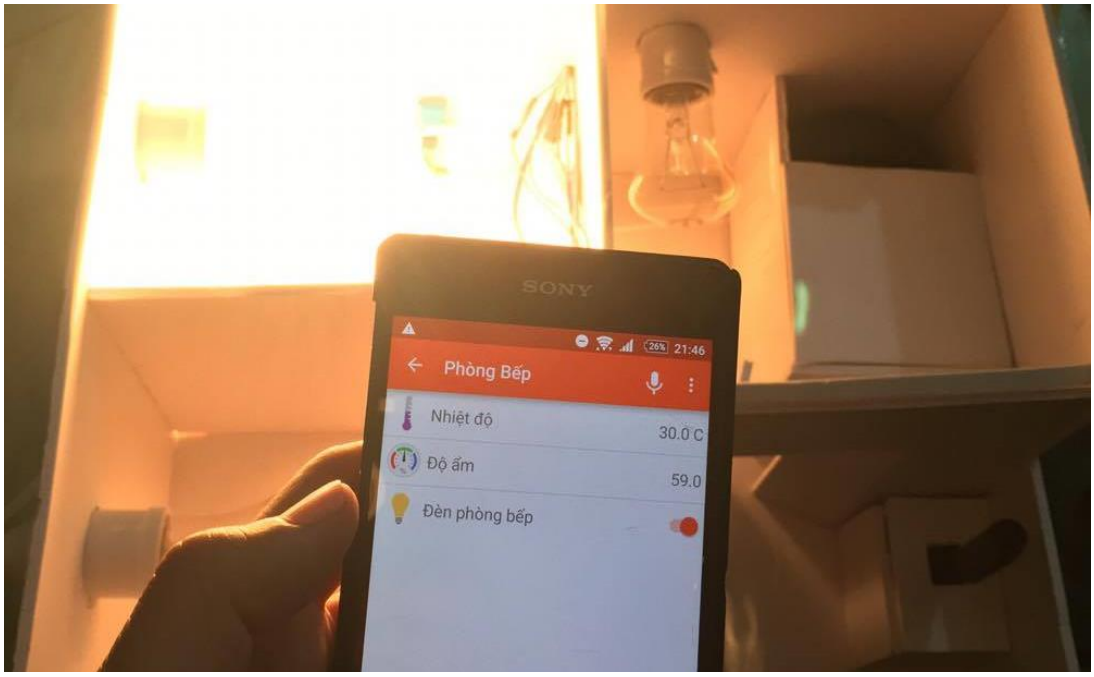
Với chức năng này, người dùng có thể thông qua truy cập địa chỉ của openHAB qua trình duyệt web hoặc cài đặt ứng dụng openHAB trên điện thoại sử dụng hai hệ điều hành IOS và Android. Từ đây, với giao diện đã được cấu hình, người dùng có thể ra lệnh bật tắt đèn các phòng cũng như theo dõi trạng thái môi trường.

Qua quá trình kiểm thử, người dùng đã có thể thao tác xử lý chính xác lên tới 95%, dữ liệu nhiệt độ, độ ẩm được gửi lên cập nhật theo khoảng thời gian nhất định, trạng thái của các bóng đèn nếu có sự thay đổi cũng được cập nhật tức thì để người dùng có thể kiểm soát ngôi nhà một cách tốt nhất.

Trong Hình 5.18, khi tất cả các đèn đều đang tắt, chúng ta truy cập ứng dụng openHab và vào mục quản lý phòng bếp, hiện tại công tắc gạt của phòng bếp đang ở chế độ off. Khi người dùng bấm chuyển công tắc như Hình 5.19, đèn phòng bếp đã được bật. Ngoài ra ứng dụng cũng hiển thị thông tin nhiệt độ và độ ẩm của các phòng vào thời điểm hiện tại là 30^0C và 59% như hai Hình 5.18 và 5.19.



Hình 5.18 Đèn phòng bếp hiện đang tắt



Hình 5.19. Đèn phòng đã bếp được bật

5.2.2.4 *Chức năng điều khiển thiết bị dựa theo điều kiện môi trường*

Với chức năng này, do có sự hạn chế về mặt thời gian cũng như kiến thức, em chỉ phát triển được chức năng thu thập dữ liệu nhiệt độ, độ ẩm ngôi nhà trong khoảng thời gian 10 phút mỗi lần, lưu trữ và hiển thị cho người dùng. Bên cạnh đó, nếu nhiệt độ lớn hơn 50 độ C, độ ẩm thấp dưới 20% thì chương trình sẽ tự động phát âm thanh báo động cảnh báo cho người dùng qua loa. Tuy nhiên độ chính xác cũng như tính ổn định của tính năng này là chưa tốt, em sẽ cải thiện và thêm các tính năng khác nữa trong tương lai.

KẾT LUẬN

"***Điều khiển và giám sát ngôi nhà thông minh bằng giọng nói và ứng dụng điện thoại***" là một đề tài đòi hỏi nhiều thời gian nghiên cứu bởi trong một ngôi nhà thì có rất nhiều phương pháp điều khiển để ngôi nhà của chúng ta ngày càng thông minh hơn. Trong đó việc điều khiển bằng giọng nói để tạo sự thuận lợi cho người dùng là một điều cần thiết cũng như việc giám sát ngôi nhà qua máy tính và ứng dụng điện thoại.

Trên đây là những trình bày chi tiết của em về đề tài"***Điều khiển và giám sát ngôi nhà thông minh bằng giọng nói và ứng dụng điện thoại***", do giới hạn về trình độ nên em đã gặp khá nhiều khó khăn trong việc tiếp cận công nghệ, phát triển các chức năng một số vấn đề kĩ thuật khác…. Ngoài ra về thời gian cũng như nhân lực nên em chưa phát triển được nhiều tính năng cũng như tối ưu được hết hiệu năng làm việc của chương trình, trong thời gian sắp tới em sẽ cố gắng hoàn thiện thêm nữa để có thể mang sản phẩm của bản thân đóng góp cho xã hội và cộng đồng.

Trong quá trình thực hiện đồ án tốt nghiệp lần này em cũng không tránh khỏi những hạn chế. Em mong nhận được sự góp ý của thầy cô để có thể xây dựng hoàn chỉnh một ngôi nhà thông minh với nhiều chức năng hơn trong tương lai.

Em xin chân thành cảm ơn!

Sinh viên

Nguyễn Đình Tâm

TÀI LIỆU THAM KHẢO

[1] Raspberry Pi - Teach, Learn, and Make with Raspberry Pi [Online]. Available: https://www.raspberrypi.org/

[2] SmartHome by Bkav [Online]. Available: http://www.smarthome.com.vn/

[3] DHT11 Humidity & Temperature Sensor - D-Robotics UK [Online]. Available: http://www.micropik.com/PDF/dht11.pdf

[4] Getting Started with the Alexa Voice Service [Online]. Available: https://developer.amazon.com/public/solutions/alexa/alexa-voice-service/getting-started-with-the-alexa-voice-service

[5] Getting Started with the Alexa Skills Kit [Online]. Available: https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/getting-started-guide

[6] AWS Lambda Developer Guide [Online]. Available: http://docs.aws.amazon.com/lambda/latest/dg/welcome.html

[7] AWS IoT Documentation[Online]. Available: https://aws.amazon.com/documentation/iot/

[7] openHAB Documentation[Online]. Available: http://docs.openhab.org/

[8] Using Google Speech API from Python https://progfruits.wordpress.com/2014/05/31/using-google-speech-api-from-python/