

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA ĐIỆN ĐIỆN TỬ
BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG

ĐỒ ÁN 1

**THIẾT KẾ PHẦN CỨNG ĐIỀU KHIỂN THIẾT
BỊ BẰNG GIỌNG NÓI**

NGÀNH HỆ THỐNG NHÚNG VÀ IOT

Sinh viên: **TRẦN TOÀN THẮNG**
MSSV: 20139015

TP. HỒ CHÍ MINH – 05/2023

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA ĐIỆN ĐIỆN TỬ
BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG

ĐỒ ÁN 1

**THIẾT KẾ PHẦN CỨNG ĐIỀU KHIỂN THIẾT
BỊ BẰNG GIỌNG NÓI
ver.1**

NGÀNH HỆ THỐNG NHÚNG VÀ IOT

Sinh viên: **TRẦN TOÀN THẮNG**
MSSV: 20139015

Hướng dẫn: **ThS. TRƯƠNG QUANG PHÚC**

TP. HỒ CHÍ MINH – 05/2023

PHIẾU ĐÁNH GIÁ ĐỒ ÁN

1. Tên đề tài: THIẾT KẾ PHẦN CỨNG ĐIỀU KHIỂN THIẾT BỊ BẰNG GIỌNG NÓI
2. Sinh viên thực hiện: Trần Toàn Thắng MSSV 20139015
3. Giảng viên hướng dẫn: ThS.Trương Quang Phúc

NỘI DUNG NHẬN XÉT:

Stt	Nội dung	Thang điểm	Điểm chấm
1	Tính mới mẻ, mức độ khó của đề tài	10	
2	Tổng quan đề tài nghiên cứu và cơ sở lý thuyết	10	
3	Tính đúng đắn và hợp lý của phương pháp nghiên cứu và phân tích thiết kế	20	
4	Thu thập và phân tích dữ liệu/ đánh giá mô hình hệ thống thiết kế	20	
5	Bố cục, hình thức và cấu trúc trình bày nội dung báo cáo đồ án	10	
6	Các kỹ năng, chuyên môn, thái độ và tính sáng tạo	15	
7	Báo cáo và trả lời câu hỏi	15	
	TỔNG ĐIỂM	100	

Lưu ý: Trường hợp giảng viên phát hiện sinh viên sao chép toàn văn công trình của người khác thì cho tổng điểm bằng 0 (không).

Điểm kết luận quy đổi (Quy về thang điểm 10, không làm tròn):

Tp.HCM, ngày 16 tháng 06 năm 2023

Giảng viên đánh giá

LỜI CẢM ƠN

Lời đầu tiên, tôi xin trân trọng cảm ơn thầy Trương Quang Phúc đã định hướng đề tài và hướng dẫn tôi thực hiện trong quá trình hoàn thành sản phẩm và báo cáo đồ án 1. Thầy là một người rất tận tâm trong công việc giảng dạy và hỗ trợ sinh viên khi gặp khó khăn. Bên cạnh đó, tôi cũng chân thành gửi lời cảm ơn đến các giảng viên của khoa Điện – Điện Tử đã truyền đạt cho tôi những kiến thức nền tảng và những động lực để hoàn thành đề tài. Tôi cũng cảm ơn những tác giả của các dự án mã nguồn mở, những tác giả của các bài viết chia sẻ kinh nghiệm đã mang lại cho tôi những ý tưởng, những kinh nghiệm quý báu cho quá trình phát triển đề tài.

Xin chân thành cảm ơn!

Người thực hiện đề tài

Trần Toàn Thắng

TÓM TẮT

Trong đồ án này, tôi đã thiết kế một sản phẩm sử dụng giọng nói của người dùng để điều khiển bật tắt các thiết bị trong gia đình. Sản phẩm này sẽ là nền tảng cho việc phát triển các thiết bị sử dụng công nghệ nhận dạng giọng nói, các thiết bị trợ lý ảo Tiếng Việt. Trong quá trình phát triển thiết bị, tôi đã sử dụng một chiếc máy tính nhúng nhỏ gọn để đảm nhận trọng trách xử lý giọng nói của người dùng. Vì chiếc máy tính nhúng này không có sẵn micro thu âm, tôi đã sử dụng một module rời để hỗ trợ công việc thu âm, tận dụng cả công loa và đèn led có trên module này để sử dụng cho việc giao tiếp với người dùng. Tôi đã thiết kế một bảng mạch in với mục đích dùng trực tiếp nguồn điện trong gia đình để vừa cung cấp năng lượng cho phần cứng hoạt động và vừa cấp nguồn cho các thiết bị được điều khiển. Dựa vào những kiến thức đã được học, tôi đã xây dựng một trình điều khiển để điều khiển các rơ-le, một từ khóa đặc biệt để gọi thiết bị hoạt động, một chương trình python có thể xử lý giọng nói và điều khiển các rơ-le qua trình điều khiển. Cuối cùng, tôi thực hiện tự động hóa các tác vụ để người dùng có thể dễ dàng sử dụng.

MỤC LỤC

DANH MỤC HÌNH	vi
DANH MỤC BẢNG	viii
Chương 1 TỔNG QUAN	1
1.1. GIỚI THIỆU	1
1.2. MỤC TIÊU ĐỀ TÀI	2
1.3. TÌNH HÌNH NGHIÊN CỨU	3
1.3.1. Tình hình nghiên cứu ở nước ngoài:	3
1.3.2. Tình hình nghiên cứu hiện tại ở Việt Nam.....	4
1.4. PHƯƠNG PHÁP NGHIÊN CỨU	6
1.5. BỐ CỤC VÀ NỘI DUNG THỰC HIỆN	6
Chương 2 CƠ SỞ LÝ THUYẾT	7
2.1. GIỚI THIỆU PHẦN CỨNG RASPBERRY PI ZERO 2W	7
2.2. GIỚI THIỆU RESPEAKER 2-MICS RASPBERRY PI HAT	9
2.3. GIỚI THIỆU CÔNG NGHỆ XỬ LÝ NGÔN NGỮ TỰ NHIÊN	11
2.4. CÔNG NGHỆ CHUYỂN ĐỔI VĂN BẢN VÀ GIỌNG NÓI.....	12
2.5. LINUX KERNEL VÀ MODULE TRÌNH ĐIỀU KHIỂN.....	13
Chương 3 THIẾT KẾ HỆ THỐNG.....	14
3.1. YÊU CẦU HỆ THỐNG	14
3.2. ĐẶC TẢ HỆ THỐNG	14
3.2.1. Chức năng của thiết bị.....	14

3.2.2.	Mô hình tổng thể	15
3.3.	THIẾT KẾ PHẦN CỨNG.....	16
3.3.1.	Sơ đồ nguyên lý tổng thể	16
3.3.2.	Khởi nguồn.....	18
3.3.3.	Khởi rơ-le điều khiển	18
3.3.4.	Khởi kết nối module.....	19
3.4.	XÂY DỰNG PHẦN MỀM	20
3.4.1.	Xây dựng trình điều khiển cho các rơ-le.....	20
3.4.2.	Tạo một từ khóa đặc biệt để gọi thiết bị hoạt động	28
3.4.3.	Xây dựng chương trình chính và hoàn thiện tính năng	29
Chương 4	KẾT QUẢ	36
4.1.	THI CÔNG THIẾT BỊ.....	36
4.2.	KẾT QUẢ HOẠT ĐỘNG CỦA THIẾT BỊ	39
Chương 5	KẾT LUẬN-HƯỚNG PHÁT TRIỂN.....	40
5.1.	KẾT LUẬN	40
5.2.	HƯỚNG PHÁT TRIỂN	40
	TÀI LIỆU THAM KHẢO.....	42

DANH MỤC HÌNH

Hình 2.1 Thông tin cấu hình cơ bản của Raspberry Pi Zero 2 W	7
Hình 2.2 Tổng quan phần cứng ReSpeaker 2-Mics Pi HAT	10
Hình 3.1 Sơ đồ tổng thể hoạt động của phần cứng	15
Hình 3.2 Sơ đồ nguyên lý của phần cứng	17
Hình 3.3 Sơ đồ nguyên lý khối nguồn	18
Hình 3.4 Sơ đồ nguyên lý khối rơ-le điều khiển	19
Hình 3.5 Sơ đồ nguyên lý khối kết nối module	20
Hình 3.6 Khai báo các thư viện cần thiết cho trình điều khiển	20
Hình 3.7 Khởi tạo các đối tượng cần thiết	21
Hình 3.8 Hàm được gọi khi bật tắt driver hoặc tương tác với tệp proc	21
Hình 3.9 Các đối tượng proc_ops sẽ được dùng khi tạo tệp proc	22
Hình 3.10 Hai hàm thông báo đóng và mở tệp proc lên nhật kí của kernel	22
Hình 3.11 Nội dung của hàm kiểm tra trạng thái rơ-le số 1	23
Hình 3.12 Nội dung của hàm điều khiển rơ-le	24
Hình 3.13 Thực hiện tạo các tệp proc	25
Hình 3.14 Thực hiện kiểm tra tính khả dụng của các gpio	25
Hình 3.15 Yêu cầu sử dụng các gpio sau khi đã kiểm tra	26
Hình 3.16 Đặt chế độ hoạt động và trạng thái ban đầu cho các gpio	26
Hình 3.17 Cho phép debug các gpio và báo hoàn thành trình điều khiển	26
Hình 3.18 Hủy tất cả hành động khi gặp lỗi trong quá trình bật driver	27
Hình 3.19 Hàm thực hiện đóng trình điều khiển	28
Hình 3.20 Nội dung Makefile để build driver	28
Hình 3.21 Các file ghi âm cần thiết và model hotword đầu ra	29
Hình 3.22 Lưu đồ hoạt động cơ bản của chương trình chính	30

Hình 3.23 Các module quan trọng cho chương trình chính.....	30
Hình 3.24 Các đối tượng cần thiết cho chương trình.....	31
Hình 3.25 Đoạn mã phản hồi lại cho người dùng khi phát hiện hotword	32
Hình 3.26 Đoạn mã điều chỉnh tiếng ồn và ghi âm giọng nói người dùng.....	32
Hình 3.27 Đoạn mã chuyển đổi âm thanh thành văn bản có kí tự thường	32
Hình 3.28 Đoạn mã phân tích câu lệnh của người dùng.....	33
Hình 3.29 Đoạn mã cuối cùng của hàm callback.....	33
Hình 3.30 Hoàn thành chương trình và bắt đầu hoạt động	34
Hình 3.31 Cấu hình bật trình điều khiển rơ-le mỗi khi thiết bị khởi động	34
Hình 3.32 Cấu hình khởi động chương trình mỗi khi thiết bị khởi động	35
Hình 4.1 Ảnh PCB thực tế sau khi được gia công.....	36
Hình 4.2 Ảnh phần cứng sau khi hoàn thiện các linh kiện ..	Lỗi! Thẻ đánh dấu không được xác định.
Hình 4.3 Khoanh vùng các khối trong phần cứng	37
Hình 4.4 Ảnh thiết bị khi hoàn thiện.....	38
Hình 4.5 Ảnh thiết bị hoạt động thực tế.....	38

DANH MỤC BẢNG

Bảng 2.1 Bảng so sánh Raspberry Pi Zero 2 W và Raspberry Pi 3 Model B	8
Bảng 3.1 Sơ đồ kết nối Raspberry Pi và rơ-le.....	19

CHƯƠNG 1

TỔNG QUAN

1.1. GIỚI THIỆU

Công nghiệp 4.0 nói chung và Internet of Things (IoT) nói riêng đang đóng vai trò quan trọng trong sự phát triển của nền kinh tế và các ngành nghề hiện nay ở Việt Nam. Với sự kết nối không ngừng và khả năng truyền thông giữa các thiết bị, IoT đã tạo ra nhiều lợi ích đáng kể cho các ngành nghề khác nhau. Một trong những lợi ích lớn nhất của Công nghiệp 4.0 và IoT mang lại là tăng cường hiệu suất sản xuất trong các ngành công nghiệp. Công nghiệp 4.0 đưa vào sử dụng các hệ thống tự động hóa thông minh, giúp tối ưu hóa quá trình sản xuất, giảm thiểu lỗi nhân viên và tăng cường độ chính xác và năng suất. Các thiết bị IoT cho phép ghi nhận dữ liệu từ các cảm biến và máy móc, giúp quản lý sản xuất hiệu quả hơn, từ việc theo dõi lượng nguyên liệu, quá trình sản xuất cho đến lưu trữ và phân tích dữ liệu. Điều này giúp các doanh nghiệp tăng cường khả năng cạnh tranh, giảm chi phí và tạo ra sản phẩm chất lượng cao hơn.

Bên cạnh những hiệu quả vừa được đề cập ở trên, Công nghiệp 4.0 và IoT cũng mang lại nhiều cơ hội cho lĩnh vực dịch vụ và thương mại. Với sự phát triển của kết nối internet và các ứng dụng di động, việc mua sắm trực tuyến và dịch vụ giao hàng nhanh đã trở thành một xu hướng phổ biến. Công nghiệp 4.0 giúp các doanh nghiệp nắm bắt được xu hướng này và áp dụng các giải pháp công nghệ để cải thiện trải nghiệm khách hàng, tăng cường tính tiện lợi và tăng khả năng cạnh tranh. Ngoài ra, IoT còn đóng vai trò quan trọng trong lĩnh vực y tế, năng lượng, nông nghiệp và quản lý đô thị thông minh, giúp tạo ra các giải pháp sáng tạo và bền vững cho các ngành nghề này.

Trong những năm gần đây, sự phát triển của thiết bị thông minh đã mang đến một cuộc cách mạng trong cuộc sống gia đình. Công nghệ nhận diện giọng nói và các thiết bị trợ lý ảo như Google Assistant, Amazon Alexa và Apple Siri đã trở thành một phần không thể thiếu trong ngôi nhà hiện đại. Với sự phát triển của công nghệ nhận diện giọng nói, người dùng có thể tương tác với các thiết bị thông minh bằng cách sử dụng giọng nói một cách tự nhiên. Các thiết bị trợ lý ảo có thể thực hiện nhiều tác vụ hữu ích như điều khiển ánh sáng, thiết bị gia dụng, kiểm tra thời tiết, phát nhạc và thậm chí trả lời câu hỏi thông qua việc truy cập internet. Điều này giúp tiết kiệm thời gian và nâng cao trải nghiệm người dùng trong gia đình. [1]

Ngoài ra, các thiết bị thông minh khác như hệ thống an ninh, đèn chiếu sáng tự động, máy giặt thông minh và hệ thống điều hòa không khí cũng đang ngày càng phổ biến trong các gia đình. Nhờ vào khả năng kết nối internet, người dùng có thể điều khiển và giám sát các thiết bị này từ xa thông qua điện thoại di động, mang lại sự tiện ích và an ninh cho gia đình. Sự phát triển của các thiết bị thông minh và công nghệ nhận diện giọng nói đã mở ra không gian cho sự tương tác thông minh giữa con người và công nghệ. Điều này đóng góp vào cuộc sống tiện nghi hơn, hiệu quả hơn và an toàn hơn trong gia đình ngày nay. Chính vì thế, tôi chọn đề tài này cho môn Đồ án 1 này với mục đích ứng dụng công nghệ nhận dạng giọng nói Tiếng Việt do Google cung cấp cho việc điều khiển các thiết bị công suất thấp trong gia đình, từ đó đặt nền tảng để mở rộng phát triển các ứng dụng từ giọng nói như trợ lý ảo Tiếng Việt cho cá nhân hoặc cho gia đình.

1.2. MỤC TIÊU ĐỀ TÀI

Với mục đích xây dựng một thiết bị có thể xử lý giọng nói và điều khiển rơ-le bằng Raspberry Pi Zero 2W, cần thực hiện được lần lượt các mục tiêu như sau:

- Thực hiện cấu hình để khởi động thành công Raspberry Pi Zero 2W và có thể điều khiển từ xa qua mạng LAN.

- Xây dựng một trình điều khiển hỗ trợ điều khiển rơ-le bằng các gpio qua Proc file (pseudo file system-hệ thống file ảo).
- Thiết kế, triển khai phần cứng hỗ trợ kết nối các module và điều khiển thiết bị điện gia dụng công suất vừa và nhỏ bằng rơ-le (điển hình là các thiết bị chiếu sáng).
- Thực hiện cài các gói phần mềm và thư viện cần thiết cho nhận diện giọng nói, xây dựng phần mềm nhận diện giọng nói và điều khiển rơ-le bằng trình điều khiển đã tạo.
- Tối ưu phần mềm, hỗ trợ người dùng có thể dễ dàng sử dụng.

1.3. TÌNH HÌNH NGHIÊN CỨU

1.3.1. Tình hình nghiên cứu ở nước ngoài:

Công nghệ nhận dạng giọng nói và công nghệ trợ lý ảo là hai trong những ứng dụng phổ biến của trí tuệ nhân tạo (AI) trong thời đại 4.0. Những công nghệ này đã mang lại nhiều tiện ích cho người dùng trong việc tìm kiếm thông tin, thực hiện các thao tác cơ bản hoặc phức tạp chỉ bằng giọng nói. Ngoài ra, chúng còn có khả năng đọc hiểu văn bản, dịch thuật, phân tích dữ liệu và hỗ trợ học tập, làm việc hiệu quả.

Trên thế giới hiện nay, có rất nhiều dự án liên quan đến công nghệ nhận dạng giọng nói và công nghệ trợ lý ảo được phát triển bởi các ông lớn trong lĩnh vực công nghệ. Một số ví dụ điển hình như:

- Siri của Apple: là một trợ lý ảo được tích hợp trên các thiết bị thuộc hệ sinh thái của Apple như iPhone, iPad, Macbook, Apple Watch... Siri có thể nghe và hiểu các lệnh bằng giọng nói của người dùng và thực hiện các chức năng như gọi điện thoại, gửi tin nhắn, đặt lịch hẹn, mở ứng dụng, tra cứu thông tin trên mạng...
- Google Assistant của Google: là một trợ lý ảo được tích hợp trên các thiết bị sử dụng hệ điều hành Android hoặc các thiết bị thông minh khác của Google như Google Home, Google Nest... Google Assistant có thể nghe và hiểu các

lệnh bằng giọng nói của người dùng và thực hiện các chức năng như tìm kiếm thông tin trên Google, điều khiển các thiết bị thông minh trong nhà, chơi nhạc, xem video, đặt hàng online...

- Alexa của Amazon: là một trợ lý ảo được tích hợp trên các thiết bị thông minh của Amazon như Echo, Fire TV, Kindle... Alexa có thể nghe và hiểu các lệnh bằng giọng nói của người dùng và thực hiện các chức năng như mua sắm trên Amazon, điều khiển các thiết bị thông minh trong nhà, chơi nhạc, xem video, đọc sách điện tử... [2]

Cong nghệ nhận dạng giọng nói và công nghệ trợ lý ảo mang lại nhiều lợi ích cho người dùng như tiết kiệm thời gian, tăng cường sự tiện lợi và thoải mái trong giao tiếp với các thiết bị công nghệ. Tuy nhiên, chúng cũng có một số hạn chế và thách thức cần được khắc phục như:

- Sự không chính xác hoặc sai sót trong quá trình nhận diện giọng nói do ảnh hưởng của tiếng ồn xung quanh, giọng nói không rõ ràng hoặc có âm thanh khác xen vào.
- Sự thiếu an toàn và bảo mật cho người dùng do việc thu thập và xử lý dữ liệu giọng nói có thể bị rò rỉ hoặc lạm dụng bởi các bên thứ ba không đáng tin cậy.
- Sự thiếu đa dạng và phong phú trong ngôn ngữ và văn hóa do việc phát triển công nghệ này chủ yếu tập trung vào một số ngôn ngữ phổ biến như tiếng Anh, tiếng Trung... mà bỏ qua các ngôn ngữ khác. [3]

1.3.2. Tình hình nghiên cứu hiện tại ở Việt Nam

Việt Nam hiện nay là một trong những quốc gia có tốc độ phát triển kinh tế nhanh nhất khu vực và thế giới. Để thích ứng với xu hướng toàn cầu hóa và số hóa, Việt Nam đã đẩy mạnh việc ứng dụng các công nghệ mới như trí tuệ nhân tạo (AI), điện toán đám mây, internet vạn vật (IoT)... Trong số đó, công nghệ nhận dạng giọng nói và công nghệ trợ lý ảo là hai lĩnh vực có tiềm năng lớn và được các tập đoàn lớn trong và ngoài nước quan tâm và đầu tư. Các tập đoàn này nhìn thấy tiềm năng và ưu

điểm của công nghệ này trong việc nâng cao trải nghiệm người dùng và thúc đẩy sự tiện ích trong cuộc sống hàng ngày.

Các tập đoàn công nghệ đa quốc gia như FPT, VNPT, Viettel đang đẩy mạnh nghiên cứu và phát triển công nghệ nhận dạng giọng nói và trợ lý ảo tại Việt Nam. Họ tập trung vào việc xây dựng hệ thống trợ lý ảo tiếng Việt, nhằm cung cấp một phương thức tương tác thông minh và tiện ích cho người dùng trong các lĩnh vực như điều khiển thiết bị, tìm kiếm thông tin, giao tiếp và thực hiện các tác vụ hàng ngày. Đây là một số ví dụ điển hình:

- FPT.AI Text to Speech: là một dịch vụ chuyển đổi văn bản thành giọng nói Tiếng Việt với ngữ điệu tự nhiên. Dịch vụ này sử dụng công nghệ tổng hợp giọng nói và công nghệ học sâu (Deep Learning) để tạo ra các giọng đọc phong phú về giới tính (nam/nữ) và ngữ âm (Bắc, Trung, Nam). Dịch vụ này có thể truy cập dưới dạng API và tích hợp dễ dàng trên mọi hệ thống, trên nhiều ứng dụng và thiết bị khác nhau.
- VNPT Smart Voice: là một dịch vụ chuyển đổi giọng nói thành văn bản Tiếng Việt và ngược lại. Dịch vụ này sử dụng công nghệ nhận dạng giọng nói và công nghệ học máy (Machine Learning) để xử lý các tín hiệu âm thanh và chuyển đổi chúng thành các đoạn văn bản hoặc các file âm thanh. Dịch vụ này có thể ứng dụng trong các lĩnh vực như ghi âm cuộc gọi, ghi chú bằng giọng nói, phụ đề video, phiên dịch tự động...
- Viettel AI: là một dịch vụ cung cấp các giải pháp trợ lý ảo thông minh cho doanh nghiệp. Dịch vụ này sử dụng công nghệ hội thoại tự động (Conversational AI) để tạo ra các cuộc hội thoại tự nhiên và linh hoạt với khách hàng qua kênh điện thoại hoặc chatbot. Dịch vụ này có thể ứng dụng trong các lĩnh vực như chăm sóc khách hàng, telesales, marketing, khảo sát...

1.4. PHƯƠNG PHÁP NGHIÊN CỨU

Dựa vào các sản phẩm trợ lý ảo và các công tắc thông minh có sẵn trên thị trường, các dự án mã nguồn mở tương tự, các kiến thức có sẵn từ các môn học và những hiểu biết về phần cứng Raspberry Pi, tôi đã sử dụng phương pháp nghiên cứu tài liệu và phương pháp nghiên cứu thực nghiệm khoa học để xây dựng ý tưởng và triển khai đề tài này.

1.5. BỐ CỤC VÀ NỘI DUNG THỰC HIỆN

Đề tài này được xây dựng dựa trên bố cục như sau:

- Chương 1. Tổng quan: Trình bày tổng quan về cách xây dựng đề tài.
- Chương 2. Cơ sở lý thuyết: Thông tin về các phần cứng quan trọng trong thiết bị.
- Chương 3. Thiết kế hệ thống: Trình bày ý tưởng thiết kế, sơ đồ khối, sơ đồ nguyên lý phần cứng, lưu đồ phần mềm.
- Chương 4. Kết quả: Trình bày thành quả đạt được trong quá trình xây dựng đề tài như hoạt động sản phẩm phần cứng, chức năng của phần mềm và chỉ ra rõ được nhược điểm của sản phẩm.
- Chương 5. Kết luận-Hướng phát triển: Dựa vào những mục tiêu đưa ra ban đầu và kết quả đạt được ở chương 4, từ đó rút ra đánh giá cho toàn bộ quá trình xây dựng đề tài. Nêu ra phương hướng giải quyết nhược điểm và định hướng phát triển cho đề tài.

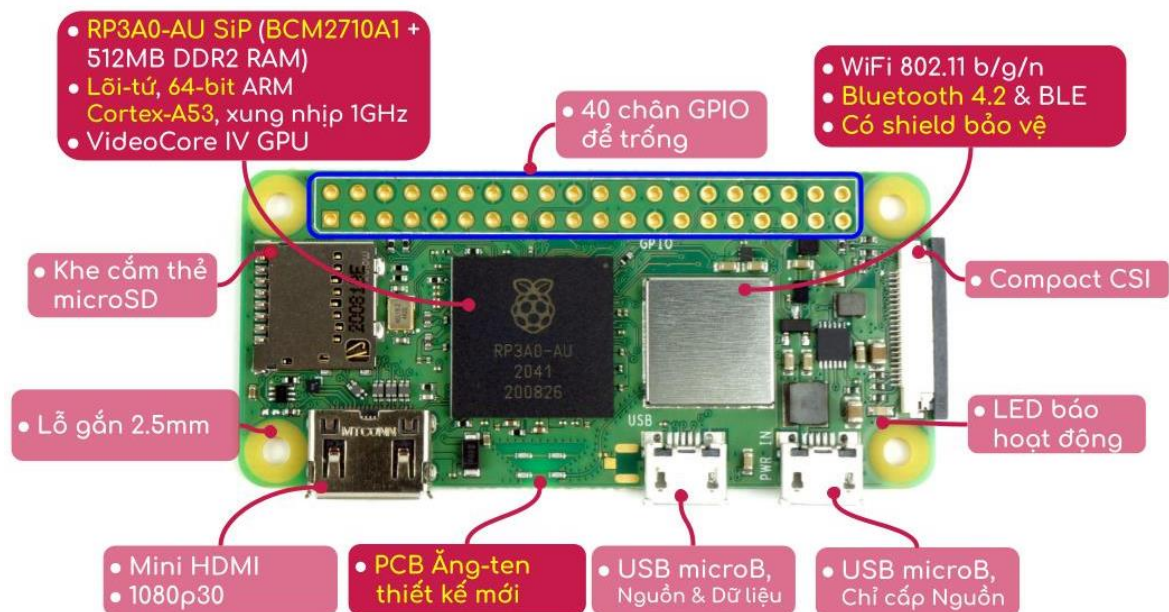
CHƯƠNG 2

CƠ SỞ LÝ THUYẾT

2.1. GIỚI THIỆU PHẦN CỨNG RASPBERRY PI ZERO 2W

Raspberry Pi Zero 2 W, một trong những mẫu máy tính nhỏ gọn và tiết kiệm năng lượng được phát triển bởi Raspberry Pi Foundation. Đây là một phiên bản nâng cấp của Raspberry Pi Zero W với nhiều cải tiến đáng chú ý.

Chiếc máy tính nhúng này được xây dựng trên vi xử lý ARM Cortex-A53 4 nhân, 64bit, tốc độ 1 GHz, với 512MB RAM. Điều này cung cấp cho máy tính khả



Hình 2.1 Thông tin cấu hình cơ bản của Raspberry Pi Zero 2 W

năng xử lý nhanh chóng và đáp ứng tốt với các tác vụ thông thường, từ chạy các ứng dụng đơn giản đến phát trực tuyến video. Đồng thời, với bộ vi xử lý này, Raspberry Pi Zero 2 W có thể sử dụng các phiên bản phần mềm hiện đại hơn và đa dạng hơn.

Bảng 2.1 Bảng so sánh Raspberry Pi Zero 2 W và Raspberry Pi 3 Model B

Tính năng/Thông số	Raspberry Pi 3 Model B	Raspberry Pi Zero 2W
Ngày ra mắt	29 Tháng 2, 2016	28 Tháng 10, 2021
SoC/SiP	Broadcom BCM2837B0 (Cập nhật Tháng 12, 2019)	RP3A0-AU SiP (BCM2710A1 + 512MB DDR2 RAM)
Kiến trúc lõi	Lõi-tứ, 64-bit, Cortex-A53	Lõi-tứ, 64-bit, Cortex-A53
GPU	VideoCoreIV	VideoCoreIV
Xung nhịp CPU	1.2 GHz	1 GHz
Bộ nhớ	microSD	microSD
RAM	1GB DDR2	512MB DDR2 (wire bond)
Cấp nguồn	USB micro-B hoặc GPIO, 5V	USB micro-B hoặc GPIO, 5V
Ethernet	10/100 Mbit/s (100base T)	Không có
USB	4 x USB 2.0	1 x USB OTG
HDMI	1 x HDMI (chuẩn)	1 x HDMI (mini)
WiFi	2.4 GHz (1 kênh) IEEE 802.11 b/g/n	2.4 GHz (1 kênh) IEEE 802.11 b/g/n
Bluetooth	Bluetooth 4.1, BLE	Bluetooth 4.2, BLE
Antenna	Chip Antenna	PCB Antenna được cải thiện
GPIO	40-pin đã được hàn sẵn	40-pin để trống
Hệ điều hành	Raspberry Pi OS	Raspberry Pi OS
Kích thước	85.6mm x 56.5mm x 11mm (HxWxD)	30mm x 65mm x 13mm (HxWxD)
PoE	None	None
Giá (Chưa thuế + Vận chuyển)	USD 35.00	USD 15.00

Ngoài ra, Raspberry Pi Zero 2 W còn được tích hợp sẵn Wi-Fi và Bluetooth, cho phép kết nối mạng và thiết bị ngoại vi không dây dễ dàng. Điều này làm cho Raspberry Pi Zero 2 W trở thành một lựa chọn phù hợp cho các ứng dụng IoT và cách hệ thống mạng không dây.

Raspberry Pi Zero 2 W được thiết kế để có kích thước nhỏ gọn, chỉ với kích thước 65 x 30 mm và trọng lượng chỉ 9 gram. Điều này làm cho nó rất thuận tiện để dùng cho các thiết bị nhỏ gọn và mang tính di động.

Dù với kích thước gọn như thế, chiếc máy tính nhúng này vẫn được trang bị hai cổng Micro-USB, một cho nguồn điện và một cổng cho kết nối ngoại vi. Bên cạnh đó còn có một cổng Micro-HDMI cho đầu ra video và 40 GPIO hỗ trợ nhiều giao thức cho phép máy tính này có thể giao tiếp với khá nhiều loại ngoại vi và các thiết bị khác.

Với cấu hình mạnh mẽ và kích thước nhỏ gọn như thế, Raspberry Pi Zero 2W rất thích hợp để thay thế Raspberry Pi 3 dùng cho các dự án nghiên cứu và phát triển hệ thống IoT, hệ thống nhúng nhỏ gọn có tính di động cao, thiết bị nhận dạng giọng nói và đặc biệt là thiết bị trợ lý ảo. [4]

2.2. GIỚI THIỆU RESPEAKER 2-MICS RASPBERRY PI HAT

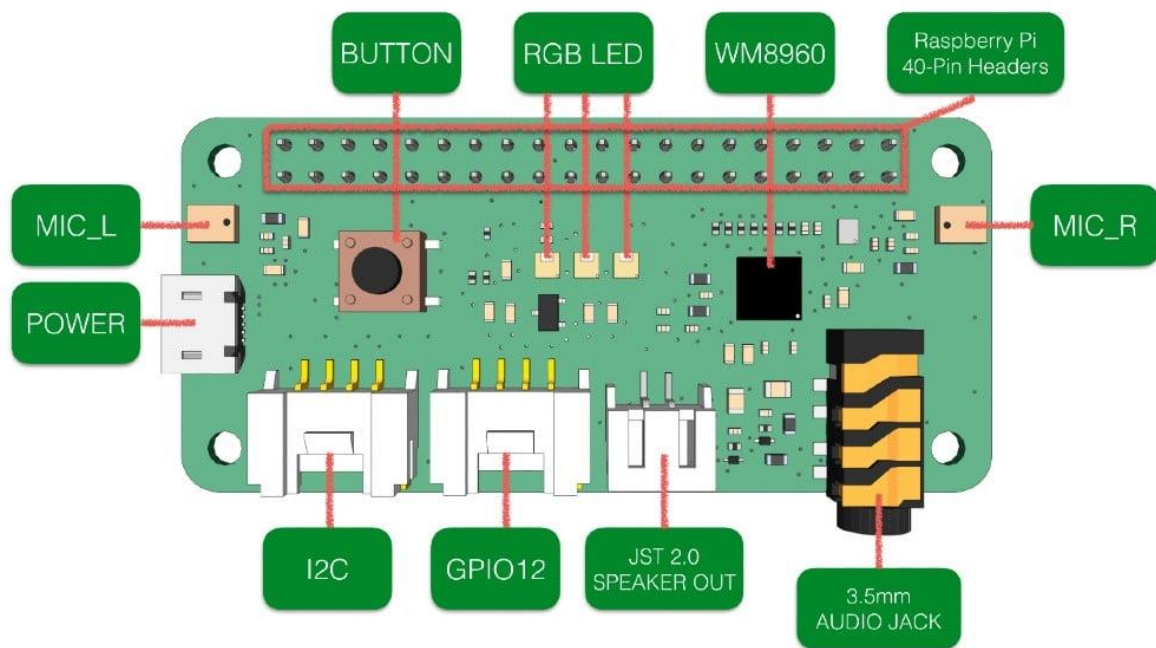
Với mục đích sử dụng giọng nói của người dùng để điều khiển thiết bị, tôi quyết định sử dụng ReSpeaker 2-Microphone Raspberry Pi HAT, một module mở rộng hỗ trợ các tác vụ âm thanh cho hầu hết các dòng máy tính nhúng Raspberry Pi và NVIDIA Jetson. ReSpeaker 2-Mics Pi HAT là một bo mạch mở rộng được thiết kế đặc biệt cho Raspberry Pi, với 2 microphone tích hợp. Đây là một giải pháp lý tưởng cho các ứng dụng liên quan đến trí tuệ nhân tạo và giọng nói. Với bo mạch này, tôi có thể xây dựng các sản phẩm mạnh mẽ và linh hoạt, tích hợp các dịch vụ như Amazon Alexa Voice Service, Google Assistant và nhiều dịch vụ khác.

Bo mạch được phát triển dựa trên chip giải mã âm thanh nổi tiếng WM8960, hỗ trợ âm thanh stereo. Hai microphone được đặt ở hai cạnh của bo mạch, cho phép thu âm chất lượng cao. Ngoài ra, bo mạch còn tích hợp 3 đèn LED RGB APA102, 1

nút nhấn và 2 cổng giao tiếp Grove để mở rộng và kết nối với các thiết bị khác. Điều này tạo ra sự linh hoạt và tùy chỉnh cao cho việc phát triển các ứng dụng đa dạng.

ReSpeaker 2-Mics Pi HAT cũng hỗ trợ xuất tín hiệu âm thanh thông qua cổng 3.5mm hoặc JST 2.0, cho phép kết nối trực tiếp với loa 1W 8Ω. Điều này giúp đơn giản hóa quá trình phát âm thanh và cung cấp âm thanh chất lượng cao cho sản phẩm.

Tổng quan phần cứng:



Hình 2.2 Tổng quan phần cứng ReSpeaker 2-Mics Pi HAT

- BUTTON: Nút nhấn có thể lập trình, kết nối tới GPIO17
- MIC_L and MIC_R: 2 Microphones
- RGB LED: 3 APA102 RGB LEDs, kết nối tới SPI interface
- WM8960: chip giải mã âm thanh nổi công suất thấp
- Raspberry Pi 40-Pin Headers: hỗ trợ Raspberry Pi Zero, Raspberry Pi 1 B+, Raspberry Pi 2 B, Raspberry Pi 3 B, Raspberry Pi 3 B+ và Raspberry Pi 4 B
- POWER: Cổng Micro USB để cấp nguồn cho ReSpeaker 2-Mics Pi HAT, xin hãy cấp nguồn cho board nếu muốn sử dụng loa

- I2C: Kết nối Grove I2C, kết nối tới I2C-1
- GPIO12: Cổng kỹ thuật số của Grove, kết nối tới GPIO12 & GPIO13
- JST 2.0 SPEAKER OUT: để kết nối tới loa, thông qua kết nối JST 2.0
- 3.5mm AUDIO JACK: để kết nối loa hoặc tai nghe qua cổng 3.5mm

Với những tính năng và khả năng tùy chỉnh đa dạng, ReSpeaker 2-Mics Pi HAT là một lựa chọn tuyệt vời cho việc phát triển các ứng dụng AI và giọng nói trên Raspberry Pi, mang đến khả năng điều khiển bằng giọng nói và trải nghiệm tương tác thông minh cho người dùng. [5]

2.3. GIỚI THIỆU CÔNG NGHỆ XỬ LÝ NGÔN NGỮ TỰ NHIÊN

Công nghệ xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP) đã trở thành một lĩnh vực quan trọng trong ngành trí tuệ nhân tạo và khoa học dữ liệu. NLP tập trung vào khả năng máy tính hiểu và tương tác với ngôn ngữ con người theo cách tự nhiên nhất có thể. Điều này cho phép máy tính xử lý, phân tích, và tạo ra ngôn ngữ tự động, mở ra nhiều cơ hội trong việc truyền tải thông tin, tương tác người-máy, và ứng dụng trong nhiều lĩnh vực khác nhau.

Một trong những thách thức lớn trong NLP là xử lý ngôn ngữ tự nhiên đa ngôn ngữ. Tiếng Việt có những đặc trưng ngữ pháp và từ vựng riêng biệt, gây khó khăn cho các phương pháp tiếng Anh truyền thống. Tuy nhiên, có một số thư viện NLP hỗ trợ tiếng Việt đã được phát triển, giúp các nhà phát triển và nghiên cứu dễ dàng làm việc với dữ liệu tiếng Việt. [6]

Một trong những thư viện NLP phổ biến nhất cho tiếng Việt là Pyvi, một thư viện mã nguồn mở được xây dựng trên Python. Pyvi cung cấp các công cụ phân tích từ vựng như tách từ (word segmentation), gán nhãn từ loại (part-of-speech tagging), rút gọn từ (lemmatization), và nhận dạng tên riêng (named entity recognition). Thư viện này giúp xử lý cú pháp tiếng Việt một cách hiệu quả và chính xác.

2.4. CÔNG NGHỆ CHUYỂN ĐỔI VĂN BẢN VÀ GIỌNG NÓI

Công nghệ nhận dạng giọng nói và chuyển đổi văn bản thành giọng nói đã đạt được sự tiến bộ đáng kể trong thời gian gần đây, đóng vai trò quan trọng trong việc tương tác giữa con người và máy tính. Lý thuyết này tập trung vào khả năng máy tính hiểu và tái tạo giọng nói con người theo cách tự nhiên nhất có thể, mở ra nhiều cơ hội trong việc tạo ra các ứng dụng như chatbot, hệ thống hỗ trợ giọng nói, và dịch tự động.

Nhận dạng giọng nói là quá trình chuyển đổi âm thanh thành văn bản. Đầu tiên, tín hiệu âm thanh được thu thập thông qua các thiết bị ghi âm hoặc microphone. Sau đó, thông qua các thuật toán và mô hình học máy, âm thanh được phân tích và rút trích các đặc trưng như tần số, biên độ, và thời gian. Các đặc trưng này được sử dụng để so khớp với các mô hình ngôn ngữ và từ điển để xác định văn bản tương ứng.

Trong khi đó, chuyển đổi văn bản thành giọng nói là quá trình ngược lại, trong đó văn bản được chuyển đổi thành âm thanh. Đầu tiên, văn bản được xử lý và phân tích để tạo ra các đơn vị ngôn ngữ như từ, âm tiết và âm vị. Sau đó, các mô hình giọng nói sử dụng các thông tin về phương ngôn ngữ, giọng điệu, và dấu câu để tạo ra âm thanh tương ứng.

Có nhiều công cụ và thư viện hỗ trợ công nghệ nhận dạng giọng nói và chuyển đổi văn bản thành giọng nói. Một trong những công cụ phổ biến là Google Cloud Speech-to-Text và Text-to-Speech API, cung cấp các dịch vụ chuyển đổi giữa giọng nói và văn bản dễ sử dụng và mạnh mẽ. Các thư viện mã nguồn mở như Mozilla DeepSpeech và Festival cũng cung cấp các công cụ nhận dạng giọng nói và chuyển đổi văn bản thành giọng nói cho cộng đồng phát triển phần mềm.

Công nghệ nhận dạng giọng nói và chuyển đổi văn bản thành giọng nói đã mở ra một tương lai đầy tiềm năng cho việc tạo ra các ứng dụng tương tác giọng nói và giúp giảm bớt rào cản trong việc truyền tải thông tin giữa con người và máy tính. [7]

2.5. LINUX KERNEL VÀ MODULE TRÌNH ĐIỀU KHIỂN

Linux Kernel là một thành phần cốt lõi của hệ điều hành Linux, chịu trách nhiệm điều khiển và quản lý các tài nguyên phần cứng của máy tính. Nó là một hệ điều hành mã nguồn mở, cho phép nhà phát triển trên toàn cầu tham gia đóng góp và cải tiến. Linux Kernel cung cấp một loạt các chức năng quan trọng, bao gồm quản lý bộ nhớ, quản lý quy trình, quản lý tệp tin, giao tiếp mạng và nhiều hơn nữa.

Một trong những thành phần quan trọng của Linux Kernel là module trình điều khiển (driver). Module trình điều khiển là một phần mở rộng của Kernel, có chức năng tương tác và điều khiển phần cứng cụ thể. Module trình điều khiển cung cấp các phương thức và chức năng để hệ điều hành tương tác với phần cứng, như card đồ họa, card âm thanh, card mạng và nhiều loại phần cứng khác.

Khi một phần cứng được kết nối với máy tính, Linux Kernel sẽ tìm và tải module trình điều khiển tương ứng để cho phép phần cứng hoạt động. Module trình điều khiển này sẽ tương tác với phần cứng thông qua các giao thức và gửi lệnh từ Kernel để điều khiển hoạt động của phần cứng. Điều này cho phép hệ điều hành và các ứng dụng có thể sử dụng và tương tác với phần cứng một cách hiệu quả.

Việc phát triển module trình điều khiển đòi hỏi kiến thức sâu về phần cứng và quy trình tương tác giữa hệ điều hành và phần cứng. Người phát triển module trình điều khiển phải hiểu rõ cấu trúc và nguyên lý hoạt động của phần cứng, cũng như các giao thức giao tiếp và quy chuẩn liên quan. Họ phải viết code và tối ưu hóa module trình điều khiển để đảm bảo tính ổn định, hiệu suất và tương thích với hệ thống.

Module trình điều khiển đóng vai trò quan trọng trong việc mở rộng khả năng tương thích phần cứng của Linux Kernel. Chúng cho phép hệ điều hành Linux chạy trên nhiều loại phần cứng khác nhau và hỗ trợ các chức năng đa dạng. Sự phát triển và bảo trì module trình điều khiển liên tục được thực hiện để cải thiện khả năng tương thích và hiệu suất của Linux trên nhiều nền tảng phần cứng. [8]

CHƯƠNG 3

THIẾT KẾ HỆ THỐNG

3.1. YÊU CẦU HỆ THỐNG

Để có thể điều khiển bật tắt các thiết bị khác bằng giọng nói, phần cứng này cần thực hiện được các yêu cầu cơ bản sau:

- Có thể ghi âm giọng nói người dùng.
- Nhận diện được giọng nói và chuyển thành văn bản.
- Có thể phân tích văn bản, lọc ra được từ khóa yêu cầu và thực thi.
- Điều khiển bật tắt các thiết bị khác bằng rơ-le.

Bên cạnh đó, tôi mở rộng thêm một số yêu cầu với mục đích nâng cao hiệu suất hoạt động và sự tiện lợi cho người dùng:

- Phần cứng có thể được đánh thức bằng một từ khóa đặt biệt.
- Phần cứng phải sử dụng trực tiếp nguồn điện 220VAC của gia đình để hoạt động và trực tiếp cấp nguồn cho các thiết bị được điều khiển.
- Có âm thanh phản hồi hoạt động và đèn báo hiệu giúp người dùng dễ sử dụng.

3.2. ĐẶC TẢ HỆ THỐNG

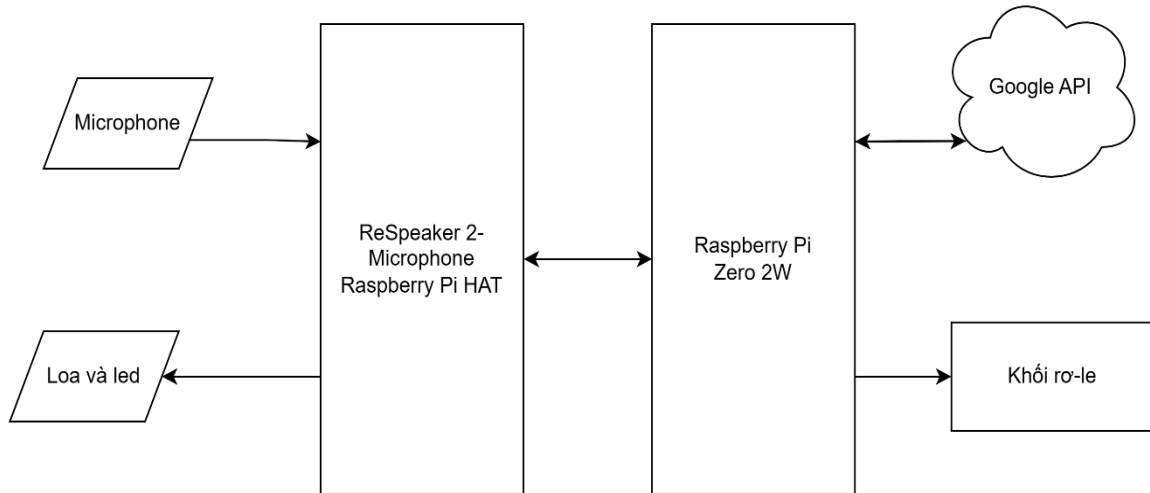
3.2.1. Chức năng của thiết bị

Thiết bị này được thiết kế như một bộ công tắc thông minh được điều khiển bằng giọng nói. Bộ công tắc thông minh này sẽ điều khiển bật tắt các thiết bị khác bằng cách quyết định có cấp nguồn 220VAC cho thiết bị đó hay không và điện áp nguồn đó được lấy trực tiếp từ nguồn cấp cho bộ công tắc thông minh này. Mỗi khi thiết bị được người dùng đánh thức bằng từ khóa, sẽ có âm thanh phản hồi lại người

dùng và lắng nghe yêu cầu. Sau khi hoàn tất yêu cầu, thiết bị sẽ phân tích, thực hiện và phản hồi lại yêu cầu đó.

3.2.2. Mô hình tổng thể

Phần cứng điều khiển thiết bị bằng giọng nói này có mô hình hoạt động tổng thể như sau:



Hình 3.1 Sơ đồ tổng thể hoạt động của phần cứng

Trong đó:

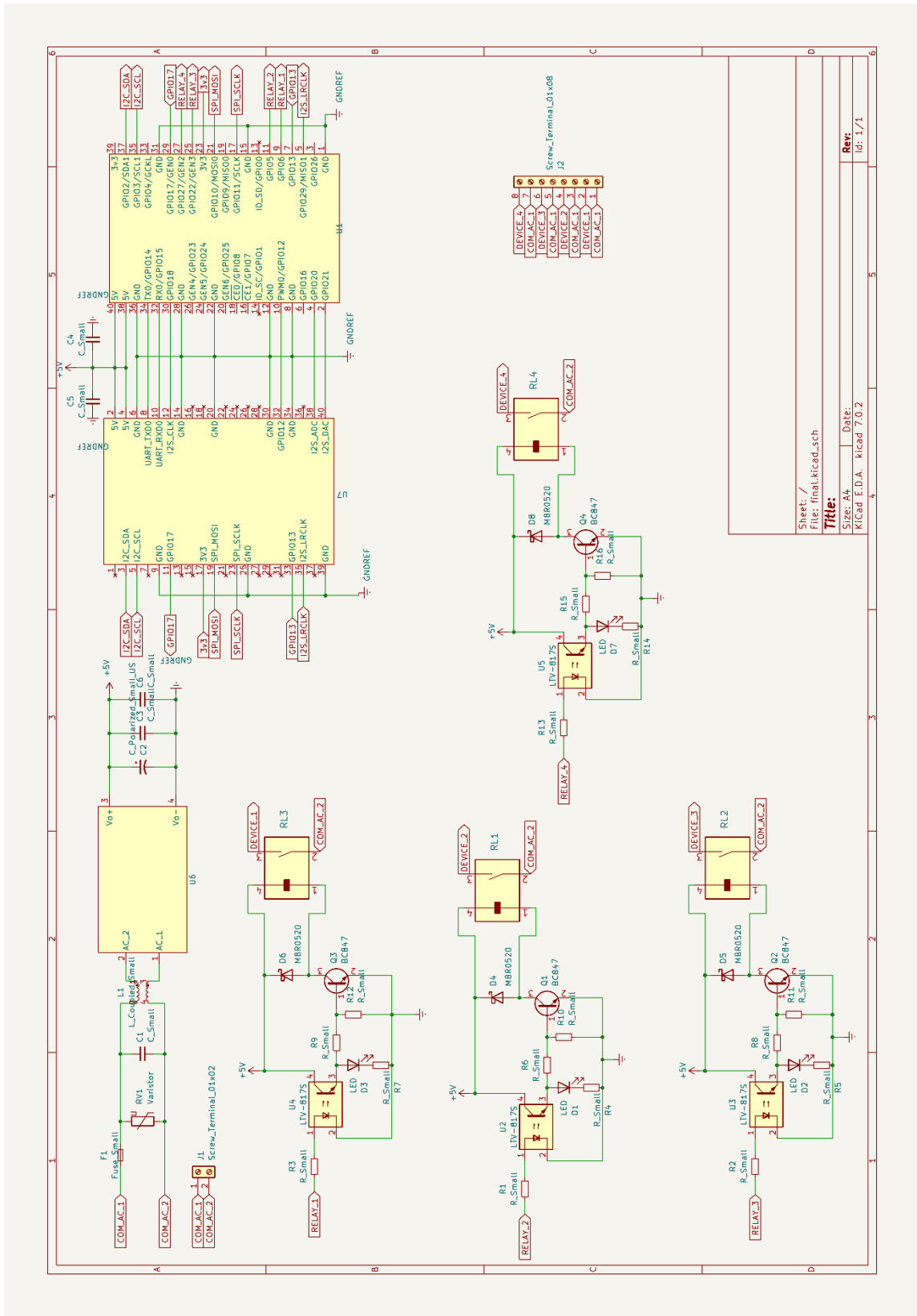
- Raspberry Pi Zero 2W: là bộ não trung tâm của thiết bị, có nhiệm vụ giao tiếp với “ReSpeaker 2-Microphone Raspberry Pi HAT” để xử lý các tác vụ âm thanh và điều khiển led phản hồi, giao tiếp với Google API để chuyển đổi dữ liệu 2 chiều giữa âm thanh và văn bản, điều khiển các rơ-le.
- ReSpeaker 2-Microphone Raspberry Pi HAT: là một module mở rộng cho Raspberry Pi, hỗ trợ cho Raspberry Pi việc ghi âm giọng nói của người dùng, phản hồi lại yêu cầu của người dùng bằng loa và đèn led.
- Khởi rơ-le: Chịu sự điều khiển của Raspberry Pi và kiểm soát nguồn cấp của các thiết bị được điều khiển.

- Google API: Cung cấp API hỗ trợ chuyển đổi giọng nói thành văn bản và ngược lại.

3.3. THIẾT KẾ PHẦN CỨNG

Để xây dựng một thiết bị điều khiển thiết bị gia đình bằng giọng nói, cần sử dụng một phần cứng thông minh và đa chức năng. Dựa vào các yêu cầu đã đưa ra ở 3.1, phần cứng này đã được tôi thiết kế bao gồm 3 khối quan trọng là: khối nguồn, khối rơ-le điều khiển và khối kết nối module.

3.3.1. Sơ đồ nguyên lý tổng thể

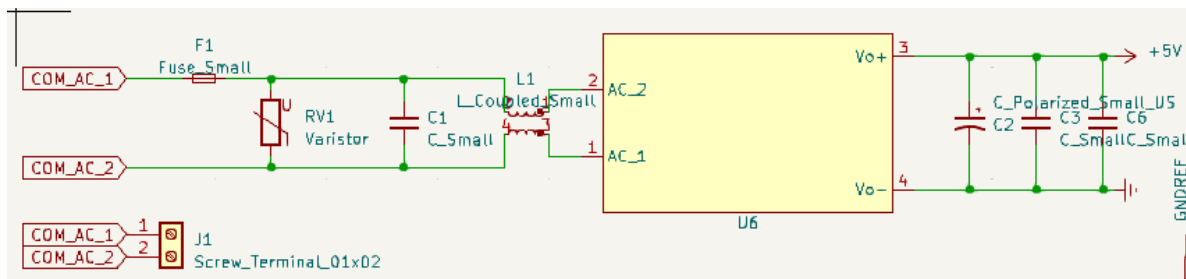


Hình 3.2 Sơ đồ nguyên lý của phân cứng

3.3.2. Khối nguồn

Khối nguồn đóng vai trò cung cấp nguồn điện phù hợp cho toàn bộ thành phần có trong thiết bị và cấp nguồn cho cả các thiết bị được điều khiển. Nó đảm bảo rằng các thành phần điện tử trong thiết bị luôn được cung cấp đủ năng lượng để hoạt động ổn định. Khối nguồn được thiết kế sử dụng bộ chuyển đổi AC-DC HLK-20M05 chuyển điện áp 220VAC thành 5VDC với công suất tối đa là 20W, đảm bảo công suất cho Raspberry Pi Zero 2W, các rơ-le và loa hoạt động ổn định.

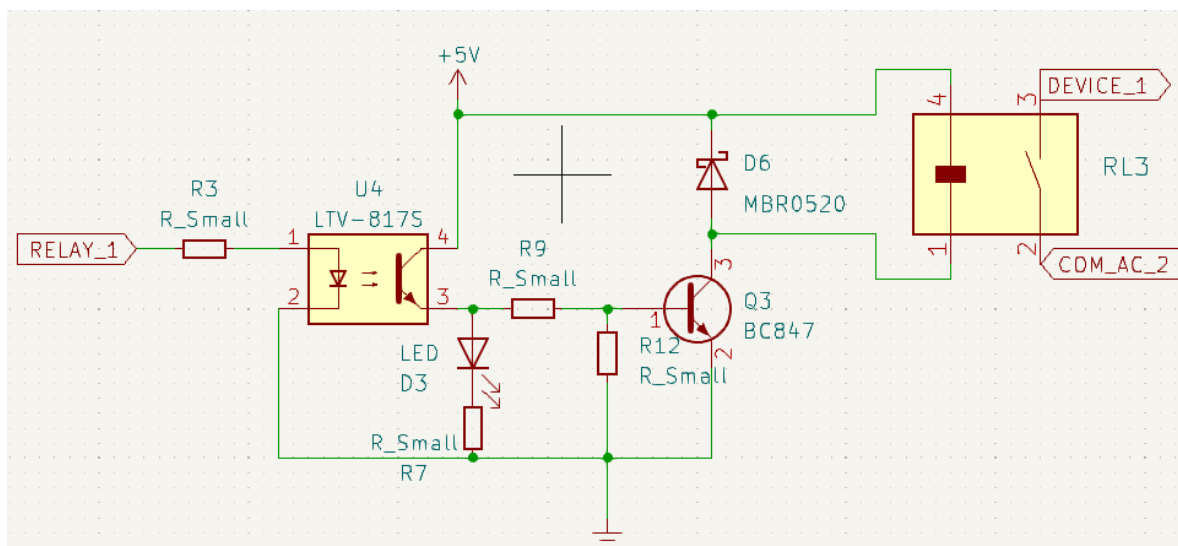
Sơ đồ nguyên lý khối nguồn:



Hình 3.3 Sơ đồ nguyên lý khối nguồn

3.3.3. Khối rơ-le điều khiển

Khối rơ-le điều khiển có nhiệm vụ bật tắt các thiết bị bằng cách quyết định có cấp điện áp từ dây pha đến thiết bị điện trong nhà hay không. Khối này được xây dựng từ một mạch phân cực cơ bản để điều khiển cuộn dây của rơ-le, có đèn led đỏ báo trạng thái của rơ-le, có diode bảo vệ transistor và có cả opto quang để bảo vệ chân tín hiệu của Raspberry Pi. Ở thiết bị này tôi sử dụng rơ-le G5NB-1A-E-5VDC, có điện áp điều khiển tối đa là 250VAC, dòng điện tối đa là 5A, kích thước nhỏ gọn, phù hợp cho việc điều khiển các thiết bị công suất vừa và nhỏ trong gia đình.



Hình 3.4 Sơ đồ nguyên lý khối rơ-le điều khiển

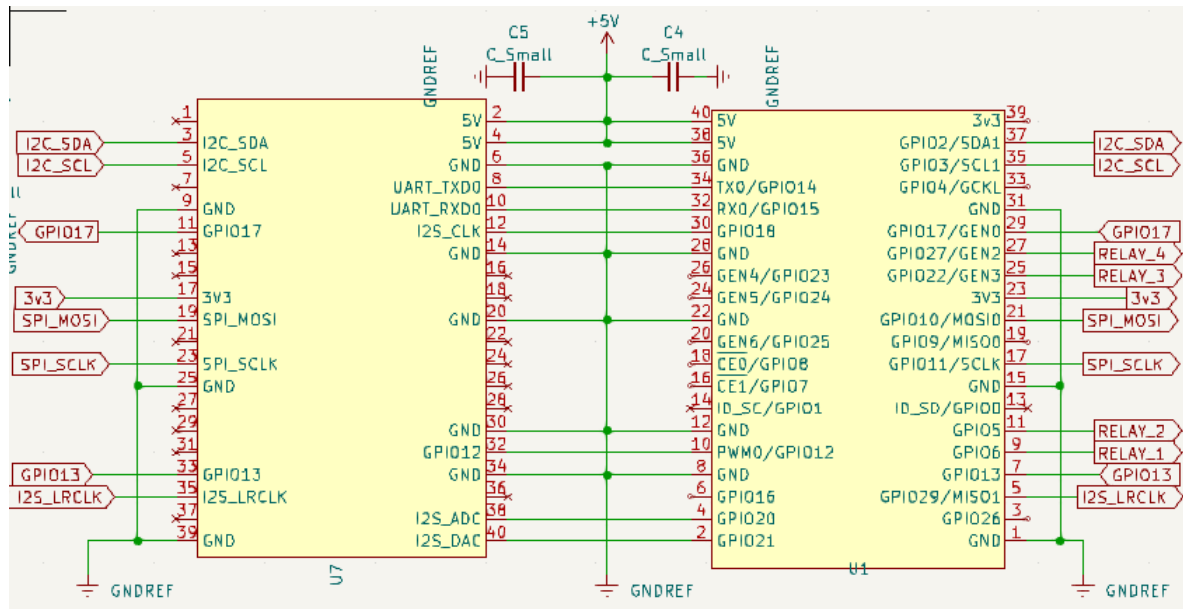
3.3.4. Khối kết nối module

Khối kết nối module bao gồm một hàng rào cái 20x2 để cắm có nhiệm vụ vừa kết nối giữa Raspberry Pi với ReSpeaker 2-Microphone Raspberry Pi HAT vừa cấp nguồn cho cả hai, ngoài ra còn kết nối gpio của Raspberry Pi với các khối rơ-le. Sơ đồ nối chân giữa Raspberry Pi và rơ-le:

Bảng 3.1 Sơ đồ kết nối Raspberry Pi và rơ-le

Rơ-le	GPIO
Relay 1	6
Relay 2	5
Relay 3	22
Relay 4	27

Sơ đồ nguyên lý khối kết nối module:



Hình 3.5 Sơ đồ nguyên lý khối kết nối module

3.4. XÂY DỰNG PHẦN MỀM

3.4.1. Xây dựng trình điều khiển cho các rơ-le

Để có thể điều khiển rơ-le bằng các gpio của Raspberry Pi, tôi quyết định sử dụng API do Linux kernel cung cấp để truy cập và sử dụng các gpio. Trình điều khiển này cung cấp khả năng điều khiển điện áp của các gpio qua hệ thống file ảo thuộc đường dẫn “/proc/” bằng cách ghi dữ liệu “on” hoặc “off” vào tệp proc tương ứng. Trình điều khiển được viết bằng ngôn ngữ C với nội dung cơ bản như sau:

- Khai báo các thư viện cần thiết:

```
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kdev_t.h>
#include <linux/fs.h>
#include <linux/cdev.h>
#include <linux/device.h>
#include <linux/delay.h>
#include <linux/proc_fs.h>
#include <linux/uaccess.h> //copy_to/from_user()
#include <linux/gpio.h> //GPIO
```

Hình 3.6 Khai báo các thư viện cần thiết cho trình điều khiển

- Khởi tạo mảng số nguyên chứa giá trị gpio của các rơ-le, khởi tạo các đối tượng định nghĩa cho các tệp proc:

```
static int RELAY[] = {-1, 6, 5, 22, 27};

static struct proc_dir_entry *driver_proc_relay_1 = NULL;
static struct proc_dir_entry *driver_proc_relay_2 = NULL;
static struct proc_dir_entry *driver_proc_relay_3 = NULL;
static struct proc_dir_entry *driver_proc_relay_4 = NULL;
static struct proc_dir_entry *driver_proc_all_relay = NULL;
```

Hình 3.7 Khởi tạo các đối tượng cần thiết

- Tạo các hàm sẽ được gọi đến khi bật hoặc tắt trình điều khiển, các hàm sẽ được gọi khi thực hiện một trong các thao tác “mở, đóng, đọc, ghi” trên các tệp proc:

```
static int __init etx_driver_init(void);
static void __exit etx_driver_exit(void);

/***** Driver functions *****/

static int etx_open(struct inode *inode, struct file *file);
static int etx_release(struct inode *inode, struct file *file);

/* RELAY_1*/
static ssize_t etx_read_1(struct file *filp,
                        char __user *buf, size_t len, loff_t * off);
static ssize_t etx_write_1(struct file *filp,
                        const char *buf, size_t len, loff_t * off);
/* RELAY_2*/
```

Hình 3.8 Hàm được gọi khi bật tắt driver hoặc tương tác với tệp proc

- Khởi tạo các đối tượng proc_ops định nghĩa các hàm sẽ được gọi đến khi tương tác với một tệp proc nhất định, các hàm này sẽ là một trong các hàm vừa được định nghĩa ở trên:

```

/* RELAY_1 */
static struct proc_ops fops_1 =
{
    .proc_read      = etx_read_1,
    .proc_write     = etx_write_1,
    .proc_open      = etx_open,
    .proc_release   = etx_release,
};

/* RELAY_2 */
static struct proc_ops fops_2 =
{
    .proc_read      = etx_read_2,
    .proc_write     = etx_write_2,
    .proc_open      = etx_open,
    .proc_release   = etx_release,
};

/* RELAY_3 */

```

Hình 3.9 Các đối tượng `proc_ops` sẽ được dùng khi tạo tệp `proc`

- Định nghĩa hai hàm sẽ được gọi khi thực hiện mở hoặc đóng các tệp `proc`, hai hàm sẽ thông báo tệp `proc` đã được mở hoặc đóng trên nhật ký kernel:

```

/*
** This function will be called when we open the Proc file
*/
static int etx_open(struct inode *inode, struct file *file)
{
    pr_info("Proc file Opened...!!!\n");
    return 0;
}

/*
** This function will be called when we close the Proc file
*/
static int etx_release(struct inode *inode, struct file *file)
{
    pr_info("Proc file Closed...!!!\n");
    return 0;
}

```

Hình 3.10 Hai hàm thông báo đóng và mở tệp `proc` lên nhật ký của kernel

- Định nghĩa hàm sẽ được gọi khi thực hiện đọc tệp proc, hàm này sẽ thực hiện kiểm tra mức điện áp hiện tại của gpio tương ứng của rơ-le, sao chép giá trị cho người dùng đọc và thông báo trên nhật ký của kernel:

```
static ssize_t etx_read_1(struct file *filp,
                          char __user *buf, size_t len, loff_t *off)
{
    uint8_t gpio_state = 0;

    gpio_state = gpio_get_value(RELAY[1]);

    len = 1;
    if( copy_to_user(buf, &gpio_state, len) > 0) {
        pr_err("ERROR: Not all the Relay1's bytes have been copied to user\n");
    }

    pr_info("Read function : RELAY_1 = %d \n", gpio_state);

    return 0;
}
```

Hình 3.11 Nội dung của hàm kiểm tra trạng thái rơ-le số 1

- Định nghĩa hàm sẽ được gọi khi ghi dữ liệu vào tệp proc, hàm này sẽ kiểm tra giá trị người dùng ghi vào tệp proc, sau đó thực hiện điều chỉnh mức điện áp của gpio tương ứng khi dữ liệu đầu vào thuộc 1 trong các giá trị sau: “1, 0, on, off”, nếu không đúng sẽ báo lỗi trên nhật ký của kernel:

```

static ssize_t etx_write_1(struct file *filp,
                          const char __user *buf, size_t len, loff_t *off)
{
    uint8_t rec_buf[10] = {0};

    if( copy_from_user(rec_buf, buf, len) > 0) {
        pr_err("ERROR: Not all the Relay1's bytes have been copied from user\n");
    }

    pr_info("Write Function : RELAY_1 Set = %c\n", rec_buf[0]);

    if (rec_buf[0]=='1' || strcmp(rec_buf,"on",2)==0) {
        gpio_set_value(RELAY[1], 1);
        pr_info("Relay 1: ON");
    }
    else if (rec_buf[0]=='0' || strcmp(rec_buf,"off",3)==0) {
        gpio_set_value(RELAY[1], 0);
        pr_info("Relay 1: OFF");
    }
    else {
        pr_err("Unknown command : Please provide one of the following valid values: 1 | 0 | on | off.\n");
    }

    return len;
}

```

Hình 3.12 Nội dung của hàm điều khiển rơ-le

- Định nghĩa hàm `etx_driver_init`, hàm này sẽ được gọi khi bật trình điều khiển. Hàm này có nhiệm vụ khởi tạo các tệp proc dựa trên các đối tượng `proc_ops` đã định nghĩa trước đó; kiểm tra tính khả dụng, yêu cầu, đặt chế độ hoạt động là output, trạng thái mặc định là tắt và cho phép debug các gpio; nếu có thất bại thì sẽ đóng trình điều khiển. Sau khi tất cả đã hoàn tất thì thực hiện thông báo trên nhật ký của kernel:

```

static int __init etx_driver_init(void)
{
    //Create proc file
    driver_proc_relay_1 = proc_create("relay_1",0666,NULL, &fops_1);
    if (driver_proc_relay_1 == NULL){
        pr_err("Cannot create the Proc file");
        goto r_proc_1;
    }
    else{
        pr_info("Success to create proc file: /proc/relay_1");
    }

    driver_proc_relay_2 = proc_create("relay_2",0666,NULL, &fops_2);
    if (driver_proc_relay_2 == NULL){
        pr_err("Cannot create the Proc file");
        goto r_proc_2;
    }
    else{
        pr_info("Success to create proc file: /proc/relay_2");
    }
}

```

Hình 3.13 Thực hiện tạo các tệp proc

```

//Checking the GPIO is valid or not
if(gpio_is_valid(RELAY[1]) == false){
    pr_err("GPIO %d is not valid\n", RELAY[1]);
    goto r_proc_1;
}

if(gpio_is_valid(RELAY[2]) == false){
    pr_err("GPIO %d is not valid\n", RELAY[2]);
    goto r_proc_2;
}

```

Hình 3.14 Thực hiện kiểm tra tính khả dụng của các gpio

```
//Requesting the GPIO
if(gpio_request(RELAY[1],"RELAY_1") < 0){
    pr_err("ERROR: GPIO %d request\n", RELAY[1]);
    goto r_gpio_1;
}

if(gpio_request(RELAY[2],"RELAY_2") < 0){
    pr_err("ERROR: GPIO %d request\n", RELAY[2]);
    goto r_gpio_2;
}
```

Hình 3.15 Yêu cầu sử dụng các gpio sau khi đã kiểm tra

```
//configure the GPIO as output
gpio_direction_output(RELAY[1], 0);
gpio_direction_output(RELAY[2], 0);
gpio_direction_output(RELAY[3], 0);
gpio_direction_output(RELAY[4], 0);
```

Hình 3.16 Đặt chế độ hoạt động và trạng thái ban đầu cho các gpio

```
//Enable debug
gpio_export(RELAY[1], false);
gpio_export(RELAY[2], false);
gpio_export(RELAY[3], false);
gpio_export(RELAY[4], false);

pr_info(" Driver Setup...Done!!!\n");
return 0;
```

Hình 3.17 Cho phép debug các gpio và báo hoàn thành trình điều khiển

```

r_gpio_4:
    gpio_free(RELAY[4]);
r_gpio_3:
    gpio_free(RELAY[3]);
r_gpio_2:
    gpio_free(RELAY[2]);
r_gpio_1:
    gpio_free(RELAY[1]);

r_proc_all:
    proc_remove(driver_proc_all_relay);
r_proc_4:
    proc_remove(driver_proc_relay_4);
r_proc_3:
    proc_remove(driver_proc_relay_3);
r_proc_2:
    proc_remove(driver_proc_relay_2);
r_proc_1:
    proc_remove(driver_proc_relay_1);

    return -1;
}

```

Hình 3.18 Hủy tất cả hành động khi gặp lỗi trong quá trình bật driver

- Định nghĩa hàm `etx_driver_exit`, hàm này sẽ được gọi khi tắt trình điều khiển. Nhiệm vụ của nó là tắt chế độ kiểm tra lỗi, giải phóng tất cả gpio điều khiển rơ-le và thông báo lên nhật kí của kernel:

```

static void __exit etx_driver_exit(void)
{
    gpio_unexport(RELAY[1]);
    gpio_unexport(RELAY[2]);
    gpio_unexport(RELAY[3]);
    gpio_unexport(RELAY[4]);

    gpio_free(RELAY[1]);
    gpio_free(RELAY[2]);
    gpio_free(RELAY[3]);
    gpio_free(RELAY[4]);

    proc_remove(driver_proc_relay_1);
    proc_remove(driver_proc_relay_2);
    proc_remove(driver_proc_relay_3);
    proc_remove(driver_proc_relay_4);
    proc_remove(driver_proc_all_relay);

    pr_info("Driver Remove...Done!!\n");
}

```

Hình 3.19 Hàm thực hiện đóng trình điều khiển

Sau khi hoàn thành nội dung của trình điều khiển. Tôi cần tạo một Makefile để xây dựng trình điều khiển ở trên thành một module cho kernel. Trong Makefile này chứa tên của module cần được xây dựng, đường dẫn trỏ đến thư viện của linux kernel, lệnh để xây dựng module và lệnh để xóa sạch các tệp đầu ra đã xây dựng trước đó:

```

obj-m += driver.o
KDIR = /lib/modules/$(shell uname -r)/build

all:
    make -C $(KDIR) M=$(shell pwd) modules
clean:
    make -C $(KDIR) M=$(shell pwd) clean

```

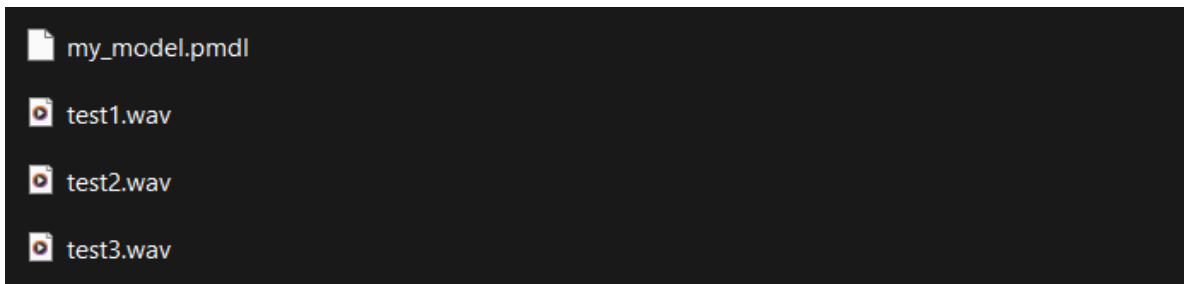
Hình 3.20 Nội dung Makefile để build driver

3.4.2. Tạo một từ khóa đặc biệt để gọi thiết bị hoạt động

Để thiết bị có cơ chế đánh thức tương tự các thiết bị trợ lý ảo khác, tôi đã sử dụng Snowboy để tạo một model chứa từ khóa đặc biệt: “quản gia”. Snowboy [9] là một dự án mã nguồn mở do đội KITT.AI phát triển với mục đích tạo ra các từ khóa

đặc biệt (hotword) hỗ trợ đánh thức được nhiều phần mềm trên nhiều nền tảng khác nhau. Mặc dù dự án này đã được ngưng phát triển từ cuối năm 2020 nhưng hiện nay vẫn được sử dụng rộng rãi cho các dự án DIY mã nguồn mở, đặc biệt là các dự án xây dựng trợ lý ảo.

Sau khi dự án Snowboy ngưng phát triển, đã có một dự án khác [10] được xây dựng dựa trên phiên bản cuối cùng của Snowboy với mục đích duy trì khả năng tạo model mới thay thế cho trang web đã đóng cửa. Dựa trên các hướng dẫn do dự án này đưa ra, tôi đã ghi âm 3 tệp âm thanh chứa từ khóa “quản gia” và sử dụng lệnh: “python generate_pmdl.py -r1=test1.wav -r2=test2.wav -r3=test3.wav -lang=other -n=my_model.pmdl” tạo ra một model hotword mới chứa từ khóa “quản gia”. Tôi sẽ sử dụng model này cho chương trình python chính trên Raspberry Pi.



Hình 3.21 Các file ghi âm cần thiết và model hotword đầu ra

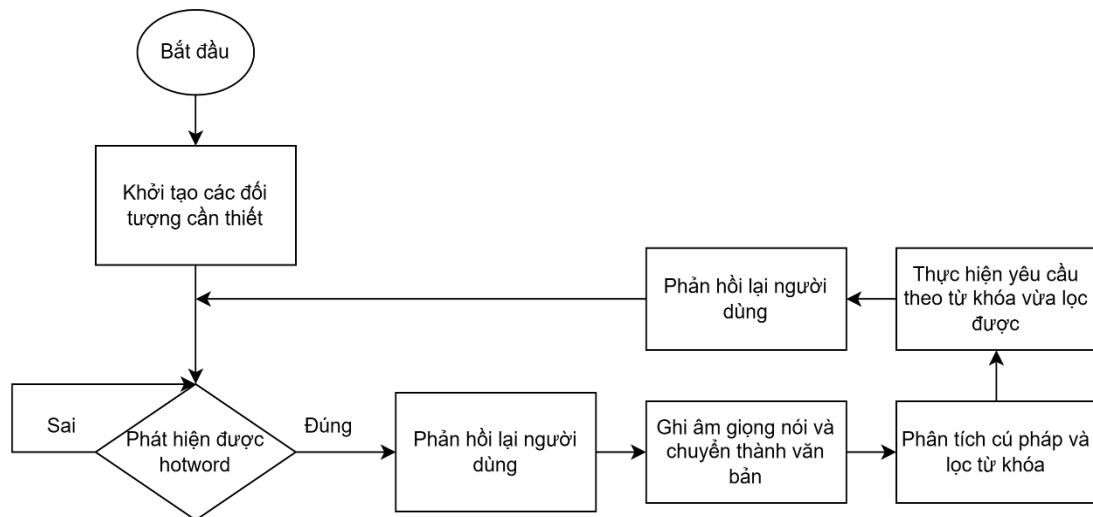
3.4.3. Xây dựng chương trình chính và hoàn thiện tính năng

Mục tiêu của chương trình này là có thể phát hiện từ khóa đặc biệt, nhận dạng được yêu cầu của người dùng và chuyển thành văn bản, phân tích cú pháp và điều khiển rơ-le. Chính vì thế, ý tưởng hoạt động cơ bản của chương trình này sẽ theo các bước của chu trình đơn sau:

- Bước 1: Phát hiện từ khóa “quản gia”.
- Bước 2: Ghi âm giọng nói của người dùng và chuyển thành văn bản.
- Bước 3: Phân tích cú pháp của văn bản và lọc các từ khóa đặc biệt.
- Bước 4: Thực hiện hành động theo các từ khóa lọc được

- Bước 5: Quay trở lại bước 1, tiếp tục phát hiện từ khóa.

Dựa vào chu trình vừa đặt ra, tôi đã xây dựng một lưu đồ chi tiết về hoạt động của chương trình:



Hình 3.22 Lưu đồ hoạt động cơ bản của chương trình chính

Sau khi hoàn thành lưu đồ, tôi sẽ bắt đầu triển khai chương trình chính bằng ngôn ngữ Python. Đầu tiên, tôi cần khai báo các module cần thiết cho chương trình:

```

from snowboy import snowboydecoder
import os
from pyvi import ViTokenizer, ViPosTagger, ViUtils
import speech_recognition as sr
from gtts import gTTS
from rgb_led.pixels import Pixels

```

Hình 3.23 Các module quan trọng cho chương trình chính

Trong đó:

- snowboydecoder trong thư viện snowboy có nhiệm vụ tạo đối tượng phát hiện từ khóa đặt biệt dựa trên model hotword đã tạo ở mục trước.
- Thư viện os để thực thi lệnh phát âm thanh từ file mp3 bằng công cụ mpg123 và thực thi lệnh điều khiển các rơ-le.

- ViTokenizer, ViPosTagger và ViUtils là các phương thức do thư viện pyvi cung cấp để phân tích cú pháp câu lệnh của người dùng.
- speech_recognition là thư viện hỗ trợ chuyển đổi giọng nói thành văn bản sử dụng API của Google.
- gtts là thư viện hỗ trợ chuyển đổi văn bản thành giọng nói và cũng sử dụng API của Google.
- Pixel là một lớp do ReSpeaker định nghĩa trong một dự án [11] để hỗ trợ sử dụng đèn led trên RESPEAKER 2-MICS RASPBERRY PI HAT.

Sau khi khai báo các module, tôi sẽ khởi tạo các đối tượng cần sử dụng cho chương trình:

```
led = Pixels()
led.wakeup()
led.think()
r = sr.Recognizer()
mic = sr.Microphone()
detector = snowboydecoder.HotwordDetector("my_model.pmdl", sensitivity=0.5)

my_control_dictionary = {
    "on": ["mở", "bật"],
    "off": ["đóng", "tắt"],
    "relay_1": ["1", "đèn", "nhất", "một"],
    "relay_2": ["2", "hai", "quạt"],
    "relay_3": ["3", "ba"],
    "relay_4": ["4", "bốn", "tư"],
    "relay_all": ["tất_cả", "hết", "toàn_bộ"],
}
```

Hình 3.24 Các đối tượng cần thiết cho chương trình

Ở hình trên, tôi khởi tạo một đối tượng led để điều khiển đèn rgb trên RESPEAKER 2-MICS RASPBERRY PI HAT và nháy đèn led thể hiện chương trình đang được khởi chạy. Tiếp theo tôi khởi tạo các đối tượng nhận dạng giọng nói “r”, micro thu âm “mic” và nhận dạng từ khóa “detector” dựa trên model “my_model.pmdl” với độ nhạy ở mức trung bình 0,5. Cuối cùng tôi khai báo một từ điển “my_control_dictionary” với mục đích nhận dạng các từ đồng nghĩa và chuẩn hóa về một từ duy nhất.

Khi đã hoàn thành khai báo mọi thứ, tôi bắt đầu định nghĩa một hàm callback. Hàm này chính là đầu não của chương trình, nó sẽ được gọi khi detector phát hiện

được từ khóa và thực hiện điều khiển thiết bị khác dựa vào lời yêu cầu của người dùng. Nội dung của hàm callback như sau:

- Nháy đèn led báo hiệu và phát âm thanh chào người dùng khi được đánh thức bằng hotword:

```
def callback():  
    print("Hotword detected")  
    led.wakeup()  
    led.speak()  
    os.system("sudo mpg123 welcome.mp3")
```

Hình 3.25 Đoạn mã phản hồi lại cho người dùng khi phát hiện hotword

- Thực hiện điều chỉnh tiếng ồn xung quanh và ghi âm yêu cầu của người dùng, đổi trạng thái led để báo trạng thái hoạt động cho người dùng:

```
with mic as source:  
    r.adjust_for_ambient_noise(source)  
    print("Say something!")  
    led.think()  
    audio = r.listen(source)  
    print("Stop listen!")
```

Hình 3.26 Đoạn mã điều chỉnh tiếng ồn và ghi âm giọng nói người dùng

- Thử chuyển đổi âm thanh vừa ghi âm được thành văn bản và chuyển tất cả kí tự in hoa thành kí tự thường:

```
try:  
    print("Process audio!!")  
    text = r.recognize_google(audio, Language='vi-VN').lower()  
    print("You said:", text)
```

Hình 3.27 Đoạn mã chuyển đổi âm thanh thành văn bản có kí tự thường

- Bắt đầu phân tích cú pháp của câu lệnh, phân tích từ loại, lọc ra các từ có từ loại phù hợp, chuẩn hóa các từ đó thành các từ cho lệnh điều khiển dựa vào từ điển my_control_dictionary đã tạo lúc trước:

```

data = ViTokenizer.tokenize(text)
tokens, tags = ViPosTagger.postagging(data)
key_word = [tokens[i] for i in range(len(tokens)) if tags[i] in ['N', 'V', 'M', 'P']]
special_key_word = []
for word in key_word:
    for key, value in my_control_dictionary.items():
        if word in value:
            special_key_word.append(key)

```

Hình 3.28 Đoạn mã phân tích câu lệnh của người dùng

- Thử xây dựng câu lệnh điều khiển thiết bị dựa trên các từ vừa được lọc trước đó, thực thi câu lệnh và phản hồi lại cho người dùng, nếu trong quá trình thực hiện lệnh có vấn đề hoặc ban đầu không chuyển được âm thanh thành giọng nói thì sẽ báo lỗi không nhận dạng được câu lệnh. Cuối cùng là tắt led khi thực hiện xong:

```

        special_key_word.append(key)
    try:
        command = 'echo "' + special_key_word[0] + '" >> /proc/' + special_key_word[1]
        print(os.system(command))
        response = "Vâng tôi đã " + key_word[0]
        print(response)
        gTTS(text=response, Lang='vi').save("response.mp3")
        os.system("sudo mpg123 response.mp3")
    except:
        os.system("sudo mpg123 not_detected.mp3")
except sr.UnknownValueError:
    os.system("sudo mpg123 not_detected.mp3")
led.off()

```

Hình 3.29 Đoạn mã cuối cùng của hàm callback

- Cuối cùng, khi đã chuẩn bị xong mọi thứ, chương trình sẽ tắt led để báo hiệu đã khởi động xong và bắt đầu vào trạng thái chờ đánh thức. Bằng cách gọi phương thức start của đối tượng detector với hàm sẽ được gọi là hàm callback và thời gian tạm nghỉ là 0.03 sau khi thực hiện xong hàm callback, chương trình sẽ bắt đầu đi vào hoạt động:

```

print("Program setup done!!")
led.off()
def interrupt_callback():
    return False

detector.start(detected_callback=callback,
               interrupt_check=interrupt_callback,
               sleep_time=0.03)
detector.stop()

```

Hình 3.30 Hoàn thành chương trình và bắt đầu hoạt động

Sau khi hoàn tất các chương trình cần thiết cho thiết bị, cần phải cấu hình cho các chương trình này tự khởi động khi được cấp nguồn để người dùng có thể dễ dàng sử dụng. Đầu tiên, để trình điều khiển các rô-le có thể tự hoạt động, cần tạo một tác vụ crontab để bật module trình điều khiển chạy cùng với kernel, tác vụ này cần chạy với quyền root và thực thi mỗi lần thiết bị được khởi động. Để làm được điều này, tôi sử dụng lệnh: “sudo crontab -e” và thêm dòng “@reboot /sbin/insmod /home/thang/Project_1/driver/driver.ko” vào cuối tệp:

```

GNU nano 5.4 /tmp/crontab.zosfEG/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
@reboot /sbin/insmod /home/thang/Project_1/driver/driver.ko

```

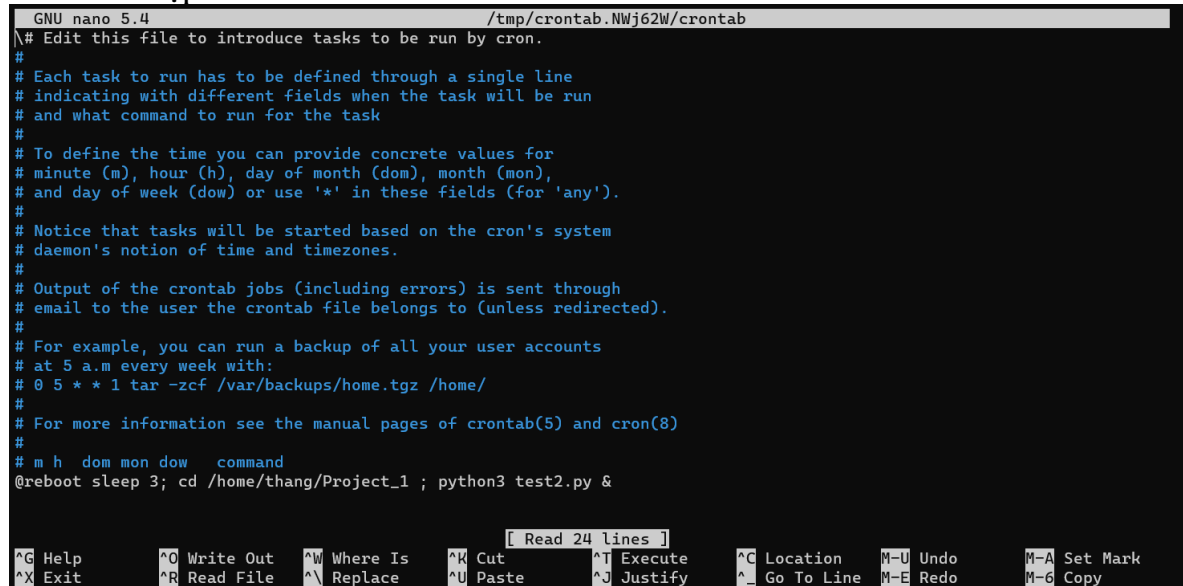
Hình 3.31 Cấu hình bật trình điều khiển rô-le mỗi khi thiết bị khởi động

Với chương trình chính, tệp python này hoạt động mỗi khi thiết bị khởi động hoàn tất, tôi sẽ tiếp tục tạo một tác vụ crontab chạy không cần quyền

root và thực thi mỗi lần sau 3 giây thiết bị được khởi động hoàn thành. Tôi đã dùng lệnh “crontab -e” và thêm dòng:

“@reboot sleep 3; cd /home/thang/Project_1 ; python3 test2.py &”

Vào cuối tệp:



```
GNU nano 5.4 /tmp/crontab.NWj62W/crontab
\# Edit this file to introduce tasks to be run by cron.
\#
\# Each task to run has to be defined through a single line
\# indicating with different fields when the task will be run
\# and what command to run for the task
\#
\# To define the time you can provide concrete values for
\# minute (m), hour (h), day of month (dom), month (mon),
\# and day of week (dow) or use '*' in these fields (for 'any').
\#
\# Notice that tasks will be started based on the cron's system
\# daemon's notion of time and timezones.
\#
\# Output of the crontab jobs (including errors) is sent through
\# email to the user the crontab file belongs to (unless redirected).
\#
\# For example, you can run a backup of all your user accounts
\# at 5 a.m every week with:
\# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
\#
\# For more information see the manual pages of crontab(5) and cron(8)
\#
\# m h dom mon dow   command
@reboot sleep 3; cd /home/thang/Project_1 ; python3 test2.py &
```

Hình 3.32 Cấu hình khởi động chương trình mỗi khi thiết bị khởi động

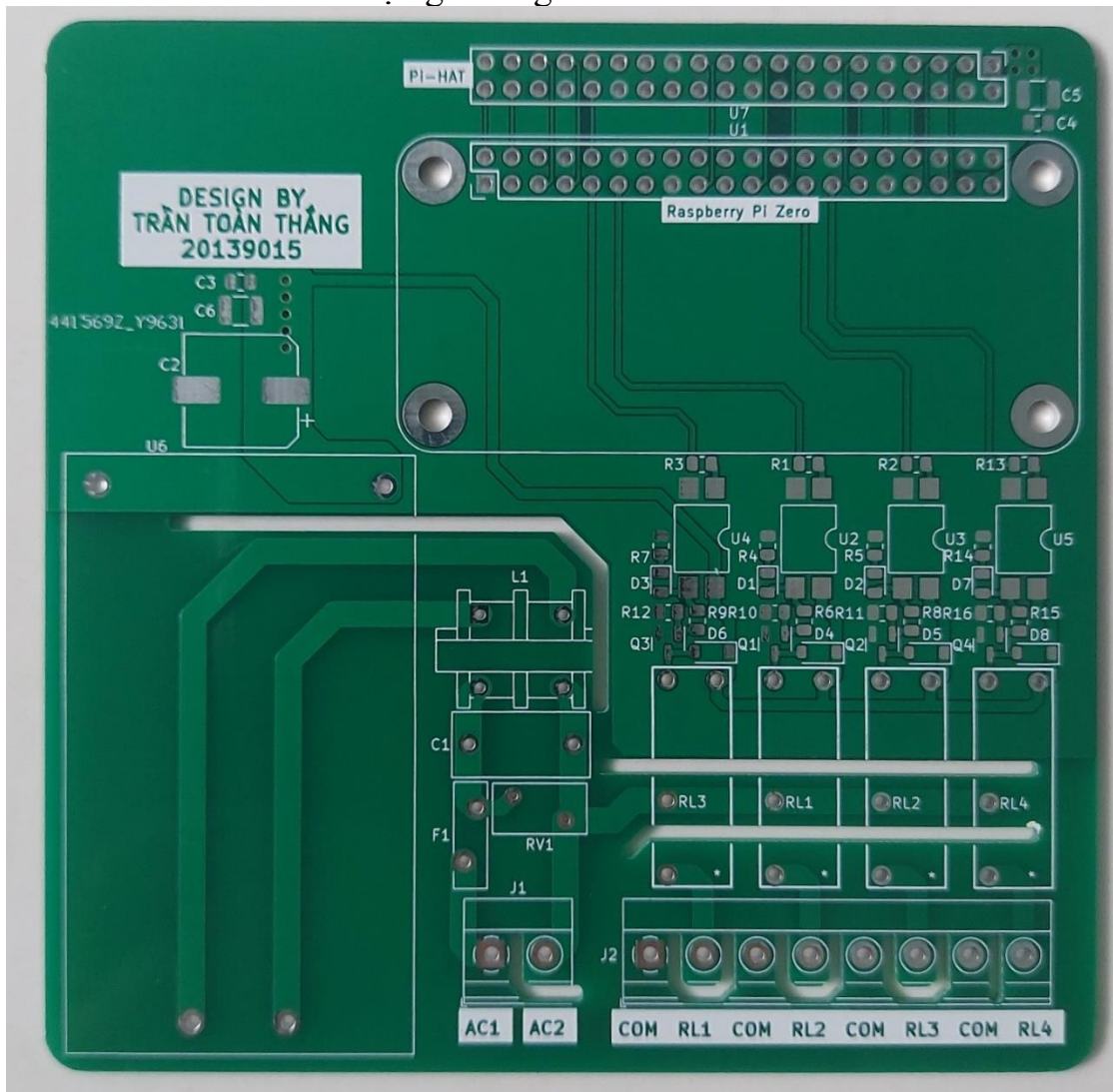
CHƯƠNG 4

KẾT QUẢ

4.1. THI CÔNG THIẾT BỊ

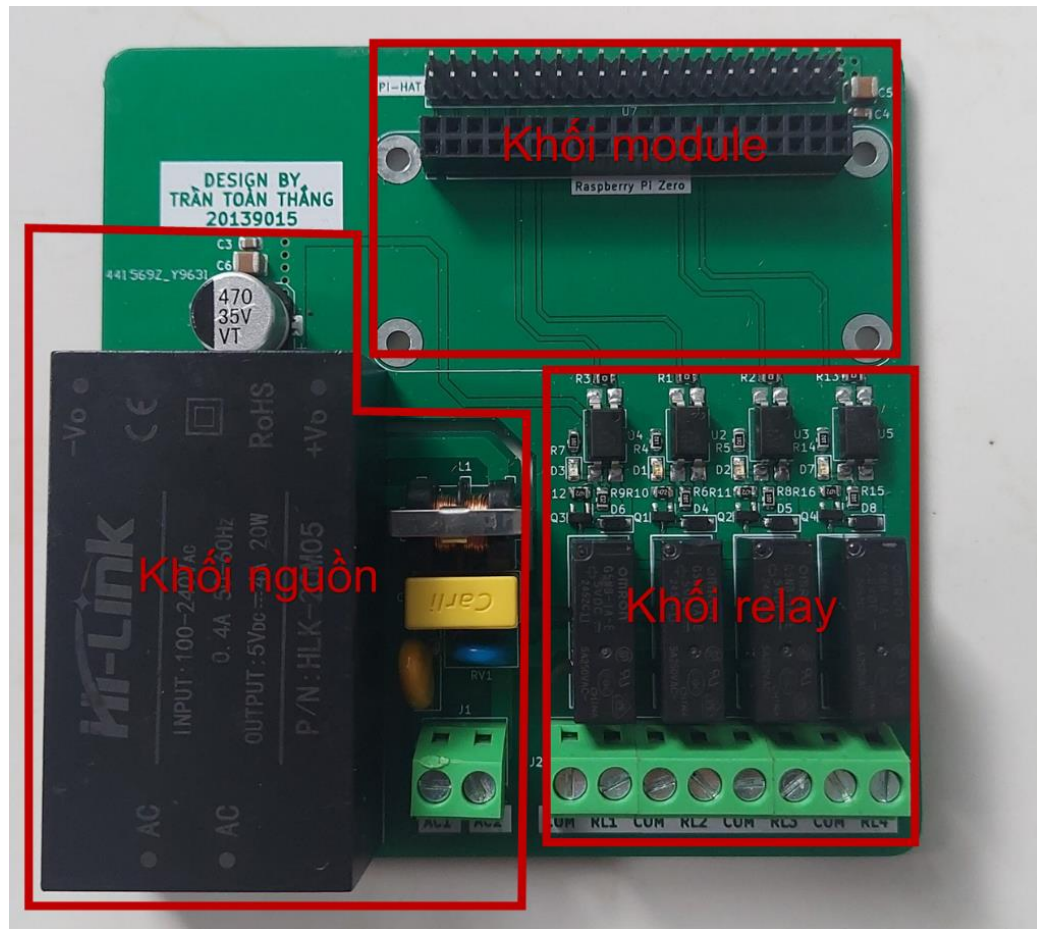
Sau quá trình thiết kế, tính toán lựa chọn linh kiện, vẽ PCB và thi công mạch và tiến hành kết nối các khối, tôi đã hoàn thiện phần cứng điều khiển thiết bị bằng giọng nói, sau đây là một số hình ảnh thực tế phần cứng:

- Ảnh PCB sau khi được gia công hoàn tất:



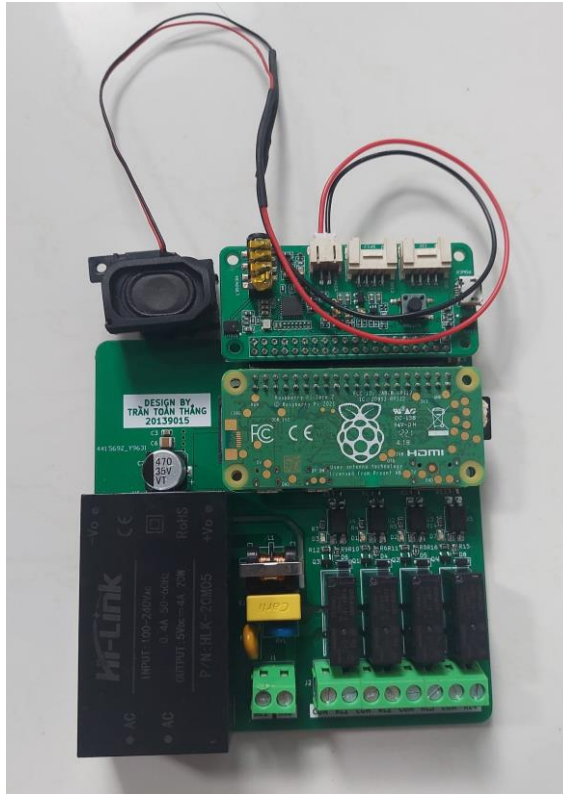
Hình 4.1 Ảnh PCB thực tế sau khi được gia công

- Ảnh phần cứng sau khi hoàn thiện các linh kiện:



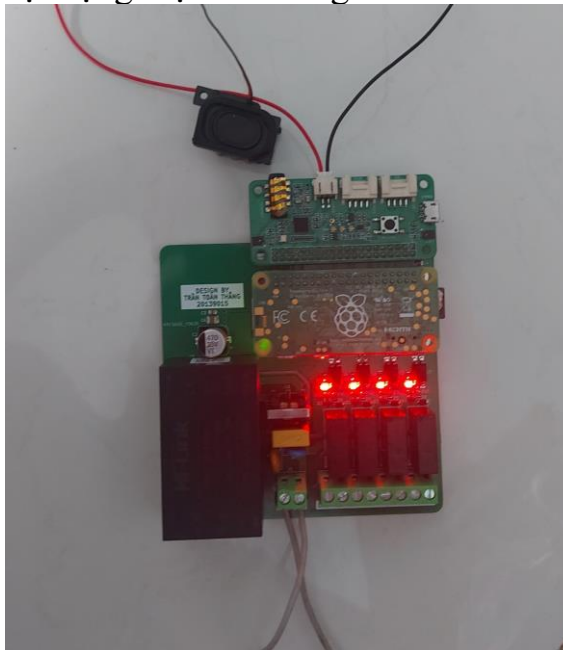
Hình 4.2 Khoanh vùng các khối trong phần cứng

- Ảnh hoàn thiện thiết bị khi lắp các module cuối cùng:



Hình 4.3 Ảnh thiết bị khi hoàn thiện

- Ảnh thiết bị hoạt động thực tế với nguồn 220VAC:



Hình 4.4 Ảnh thiết bị hoạt động thực tế

4.2. KẾT QUẢ HOẠT ĐỘNG CỦA THIẾT BỊ

Sau khi hoàn thiện, người dùng chỉ việc kết nối dây nguồn của các thiết bị gia dụng cần điều khiển vào các cổng domino, cấp nguồn, chờ thiết bị khởi động và sử dụng. Cơ chế hoạt động của thiết bị sẽ như sau:

- Khi vừa cấp nguồn, thiết bị sẽ khởi động một thời gian, sau đó sẽ chạy các phần mềm, trong lúc phần mềm khởi động sẽ có đèn led báo hiệu, khi led tắt thì phần mềm đã khởi động hoàn tất.
- Lúc này người dùng chỉ việc dùng từ khóa “quản gia” để đánh thức thiết bị, khi thiết bị phát hiện từ khóa sẽ phản hồi lại người dùng bằng câu: “Xin chào. Có tôi đây, bạn đang cần gì?” và nháy đèn led.
- Sau khi hoàn tất phản hồi lại cho người dùng, thiết bị sẽ đổi trạng thái led và bắt đầu ghi âm câu lệnh. Khi hoàn tất ghi âm, thiết bị sẽ tiến hành phân tích câu lệnh và thực thi.
- Nếu câu lệnh được thực thi thành công, thiết bị sẽ phản hồi lại yêu cầu vừa rồi của người dùng. Nếu không thể phát hiện giọng nói hoặc phân tích câu lệnh thất bại, thiết bị sẽ báo lỗi: “Xin lỗi....”.
- Kết thúc quá trình điều khiển, thiết bị sẽ quay lại chờ được đánh thức bằng từ khóa “quản gia”.

Trong quá trình chạy thực tế, thiết bị đã hoạt động và có thể thực thi các yêu cầu điều khiển của người dùng. Tuy nhiên, thiết bị này vẫn có một số nhược điểm:

- Thư viện phát hiện từ khóa của Snowboy đã ngưng phát triển từ 2020 cho nên đôi khi phát hiện nhầm từ khóa do ồn, nhiều hoặc các cụm từ có thanh âm gần giống.
- Thư viện pyvi cũng đã tạm ngưng phát triển từ năm 2021, cho nên có một số cụm từ không thể phân tích đúng ngữ cảnh, làm cho thiết bị không thể hiểu được yêu cầu của người dùng mặc dù câu lệnh đó có ý nghĩa tương đồng với câu lệnh có thể điều khiển được.

CHƯƠNG 5

KẾT LUẬN-HƯỚNG PHÁT TRIỂN

5.1. KẾT LUẬN

Sau quá trình tìm hiểu và thực hiện đề tài này, tôi đã tích lũy được một số kinh nghiệm trong việc thiết kế phần cứng, xây dựng trình điều khiển, tự động hóa tác vụ trong hệ điều hành. Nhờ sự giúp đỡ của giáo viên và các bài chia sẻ trên Internet, tôi đã hoàn thành thiết bị một cách suôn sẻ. Mặc dù vẫn còn một vài hạn chế nhưng thiết bị đã đáp ứng hầu hết các yêu cầu đã đề ra ban đầu:

- ✓ Đã có thể ghi âm giọng nói người dùng.
- ✓ Đã nhận diện được giọng nói và chuyển thành văn bản.
- ✓ Đã có thể phân tích văn bản, lọc ra được từ khóa yêu cầu và thực thi, nhưng các câu lệnh vẫn bị giới hạn.
- ✓ Đã điều khiển bật tắt các thiết bị khác bằng rơ-le, tuy nhiên chỉ có thể sử dụng để điều khiển các thiết bị có công suất thấp.
- ✓ Phần cứng đã có thể được đánh thức bằng một từ khóa đặt biệt là “quản gia” nhưng vẫn còn bị nhiễu bởi tiếng ồn.
- ✓ Phần cứng đã sử dụng trực tiếp nguồn điện 220VAC của gia đình để hoạt động và trực tiếp cấp nguồn cho các thiết bị được điều khiển.
- ✓ Đã có âm thanh phản hồi hoạt động và đèn báo hiệu giúp người dùng dễ sử dụng.

5.2. HƯỚNG PHÁT TRIỂN

Sản phẩm này là một ý tưởng đề định hướng cho việc phát triển các thiết bị trợ lý ảo Tiếng Việt cho gia đình hoặc cho cá nhân. Sản phẩm hiện tại đã có thể sử dụng nhưng về vấn đề chi phí và tính đa dụng vẫn chưa thể cạnh tranh

với các hệ thống có tính năng tương tự trên thị trường. Sản phẩm cần được phát triển với tốc độ phản hồi nhanh hơn và chính xác hơn, đáp ứng được nhiều tính năng, điều khiển được nhiều thiết bị để cắt giảm chi phí phải bỏ ra cho việc điều khiển số lượng lớn. Vì thế, tôi đã đưa ra một số định hướng để cải thiện sản phẩm như sau:

- Chuyển đổi sản phẩm thành một hệ thống với bộ xử lý trung tâm và bộ điều khiển hoạt động độc lập để tối ưu chi phí cho việc quản lý nhiều thiết bị.
- Thay đổi giải pháp xử lý giọng nói, thay đổi phương pháp phân tích cú pháp, có thể sử dụng trực tiếp các API trợ lý ảo có sẵn để làm nền tảng cho thiết bị.
- Có thể chuyển đổi thiết bị thành một trợ lý ảo mini dành cho một cá nhân, đặc biệt là những người có vấn đề với thị giác.

TÀI LIỆU THAM KHẢO

- [1] TRUNG TÂM ỨNG DỤNG CÔNG NGHỆ THÔNG TIN, “Công nghiệp 4.0 - Xu hướng thế giới và chính sách phát triển ở Việt Nam,” Trường Đại học Kinh tế Quốc dân, [Trực tuyến]. Available: <https://cait.neu.edu.vn/vi/giai-phap/cong-nghiep-4-0-xu-huong-the-gioi-va-chinh-sach-phat-trien-o-viet-nam>.
- [2] D. Martin, “Cortana vs. Siri vs. Google Assistant vs. Alexa,” 9 3 2021. [Trực tuyến]. Available: https://www.digitaltrends.com/computing/cortana-vs-siri-vs-google-now/#google_vignette.
- [3] “Eight pros and cons of virtual assistants,” LV, [Trực tuyến]. Available: <https://www.lv.com/home-insurance/8-key-financial-pros-and-cons-of-virtual-assistants>.
- [4] “Raspberry Pi Zero 2 W,” Cytron, [Trực tuyến]. Available: <https://www.cytrontech.vn/c-raspberry-pi-main-board/c-raspberry-pi-zero/p-raspberry-pi-zero-2-w-wch>.
- [5] “ReSpeaker 2-Microphone Raspberry Pi HAT,” Cytron, [Trực tuyến]. Available: <https://www.cytrontech.vn/p-respeaker-2-microphone-raspberry-pi-hat>.
- [6] “KHÁM PHÁ VNLP CÔNG NGHỆ XỬ LÝ NGÔN NGỮ TỰ NHIÊN THUẦN VIỆT,” VNLP, 27 11 2019. [Trực tuyến]. Available: <https://emandai.net/blog/kham-pha-vnlp-cong-nghe-xu-li-ngon-ngu-tu-nhien-thuan-viet-hang-dau-hien-nay/>.
- [7] “Speech Recognition for Learning,” National Center for Technology Innovation, [Trực tuyến]. Available: <https://www.readingrockets.org/article/speech-recognition-learning>.
- [8] Baeldung, “What’s the Difference Between Kernel Drivers and Kernel Modules?,” [Trực tuyến]. Available: <https://www.baeldung.com/linux/kernel-drivers-modules-difference>.
- [9] KITT.AI, [Trực tuyến]. Available: <https://github.com/Kitt-AI/snowboy>.

- [10] seasalt-ai. [Trực tuyến]. Available: <https://github.com/seasalt-ai/snowboy>.
- [11] ReSpeaker, ReSpeaker , [Trực tuyến]. Available: https://github.com/respeaker/mic_hat.