**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

 ---------------------------------

BÁO CÁO THỰC TẬP: TRUYỀN THÔNG & MẠNG MÁY TÍNH

**ĐỀ TÀI: THIẾT KẾ MẠCH ĐIỀU KHIỂN THIẾT BỊ**

**BẰNG GIỌNG NÓI**

**Giáo viên hướng dẫn: ĐINH THỊ KIM PHƯỢNG**

**Sinh viên thực hiện: Phan Anh Tuấn**

**Mã sinh viên : 1141150071**

Hà Nội – Năm 2020

[Lời cảm ơn 3](#_Toc33565430)

[Lời mở đầu 4](#_Toc33565431)

[Chương 1. Giới thiệu chung 5](#_Toc33565432)

[1.1 Đặt vấn đề 5](#_Toc33565433)

[1.2 Những khái niệm cơ bản về giọng nói: 6](#_Toc33565434)

[1.3 Cách thức nhận dạng tiếng nói 8](#_Toc33565435)

[1.4 Các khái niệm & thuật ngữ 9](#_Toc33565436)

[Chương 2. Cơ sở lý thuyết 10](#_Toc33565437)

[2.1 Phương pháp nhận diện sử dụng 10](#_Toc33565438)

[2.2 Raspberry là gì? 11](#_Toc33565439)

[2.3 Bộ công cụ CMU Sphinx là gì? 13](#_Toc33565440)

[2.4 Machine Learning - Nhận diện giọng nói 17](#_Toc33565441)

[2.5 Điều khiển theo mức 24](#_Toc33565442)

[Chương 3. Thiết lập nhận diện giọng nói 26](#_Toc33565443)

[3.1 Chuẩn bị 26](#_Toc33565444)

[3.2 Biên dịch phần mềm 27](#_Toc33565445)

[3.3 Sử dụng Pocketsphinx 29](#_Toc33565446)

[3.4 Code python 34](#_Toc33565447)

**Danh mục hình ảnh**

[Hình 1‑1 Ví dụ bản ghi âm giọng nói 6](file:///C:\Users\Su\OneDrive\Máy%20tính\Baocaothuctap.docx#_Toc33565410)

[Hình 2‑1 Cấu hình phần cứng 11](file:///C:\Users\Su\OneDrive\Máy%20tính\Baocaothuctap.docx#_Toc33565411)

[Hình 2‑2 Kiến trúc tổng quát Sphinx 15](file:///C:\Users\Su\OneDrive\Máy%20tính\Baocaothuctap.docx#_Toc33565412)

[Hình 2‑3 Amazon Echo Dot 17](file:///C:\Users\Su\OneDrive\Máy%20tính\Baocaothuctap.docx#_Toc33565413)

[Hình 2‑4 Truyền thông tin mạng Noron 18](file:///C:\Users\Su\OneDrive\Máy%20tính\Baocaothuctap.docx#_Toc33565414)

[Hình 2‑5 Ví dụ tập hợp giá trị điểm ảnh 19](file:///C:\Users\Su\OneDrive\Máy%20tính\Baocaothuctap.docx#_Toc33565415)

[Hình 2‑6 Biểu diễn đoạn âm “Hello” 19](file:///C:\Users\Su\OneDrive\Máy%20tính\Baocaothuctap.docx#_Toc33565416)

[Hình 2‑7 Lấy mẫu 20](file:///C:\Users\Su\OneDrive\Máy%20tính\Baocaothuctap.docx#_Toc33565417)

[Hình 2‑8 Phóng to 1 đoạn nhỏ đoạn âm “Hello” 20](file:///C:\Users\Su\OneDrive\Máy%20tính\Baocaothuctap.docx#_Toc33565418)

[Hình 2‑9 Tín hiệu lấy mẫu có chính xác? 21](file:///C:\Users\Su\OneDrive\Máy%20tính\Baocaothuctap.docx#_Toc33565419)

[Hình 2‑10 Mạng Noron hồi quy 22](file:///C:\Users\Su\OneDrive\Máy%20tính\Baocaothuctap.docx#_Toc33565420)

[Hình 2‑11 Dự đoán của mạng Noron 23](file:///C:\Users\Su\OneDrive\Máy%20tính\Baocaothuctap.docx#_Toc33565421)

[Hình 2‑12 Điều khiển độ rộng xung 25](file:///C:\Users\Su\OneDrive\Máy%20tính\Baocaothuctap.docx#_Toc33565422)

[Hình 3‑1 Dictionary để nạp cách đánh vần cho sphinx 30](file:///C:\Users\Su\OneDrive\Máy%20tính\Baocaothuctap.docx#_Toc33565423)

[Hình 3‑2 file language.model.lm 30](file:///C:\Users\Su\OneDrive\Máy%20tính\Baocaothuctap.docx#_Toc33565424)

[Hình 3‑3 Configuration.py file nạp thiết bị từ file cấu hình 33](#_Toc33565425)

[Hình 3‑4 Điều khiển quạt 34](#_Toc33565426)

[Hình 3‑5 Điều khiển đèn 34](#_Toc33565427)

[Hình 3‑6 Chương trình chính 35](file:///C:\Users\Su\OneDrive\Máy%20tính\Baocaothuctap.docx#_Toc33565428)

[Hình 3‑7 Sau khi chạy chương trình 36](#_Toc33565429)

# Lời cảm ơn

Trong suốt quá trình thực hiện báo cáo tốt nghiệp, chúng tôi đã nhận được rất nhiều sự giúp đỡ tận tình của quý thầy cô hướng dẫn cùng sự động viên từ gia đình, người thân và bạn bè. Nhân dịp hoàn thành báo cáo thực tập tốt nghiệp, chúng tôi xin gửi lời cám ơn sâu sắc và chân thành tới:

Cô Đinh Thị Kim Phượng – Giảng viên Khoa Điện tử Trường Đại Học Công nghiệp Hà Nội, tuy luôn bận rộn với công việc nghiên cứu và giảng dạy riêng của mình, cô vẫn luôn quan tâm và hướng dẫn nhóm hằng tuần để đảm bảo tiến độ báo cáo. Đặc biệt, với chuyên môn và kiến thức trong lĩnh vực nghiên cứu của mình, cô đã chỉ dẫn và hướng chúng tôi theo sát đề tài và phạm vi báo cáo. Qua đó, chúng tôi đã học hỏi được rất nhiều, không chỉ kiến thức chuyên môn mà còn là thái độ nghiên cứu một cách khoa học, nghiêm túc, đúng chuẩn mực, giúp chúng tôi hoàn thành báo cáo một cách tốt nhất.

Các thầy cô giảng dạy tại Khoa Điện tử Trường Đại Học Công nghiệp Hà Nội, chúng tôi đã học được từ các thầy cô những kiến thức chuyên môn quý báu trong khoảng thời gian 4 năm đại học. Có thể những kiến thức đó không trực tiếp được sử dụng trong luận văn này, nhưng phần nào đã giúp chúng tôi có một nền tảng chuyên môn vững chắc để tự tin hơn khi thực hiện luận văn.

Và đặc biệt, lời cảm ơn cuối cùng, chúng tôi xin được dành cho gia đình và những người thân của chúng tôi, những người luôn sát cánh và động viên chúng tôi trong những lúc khó khăn nhất, đó chính là nguồn động lực tinh thần quý giá giúp nhóm hoàn thành luận văn ngày hôm nay.

Một lần nữa chúng tôi cảm ơn và ghi nhận tất cả những sự giúp đỡ kể trên. Với tất cả sự nỗ lực và cố gắng của bản thân trong những tháng thực hiện, chúng tôi đã hoàn thành khóa thực tập tốt nghiệp, và tất nhiên sẽ không tránh khỏi những thiếu sót cần phải hoàn thiện, rất mong nhận được sự góp ý của quý thầy cô và các bạn.

# Lời mở đầu

Sự phát triển của khoa học kỹ thuật ngày càng nhanh góp phần nâng cao năng suất lao động. Đặc biệt sự ra đời và phát triển các công nghệ mới nhằm tạo ra sự tự động hóa, sự tiện lợi trong xã hội cũng như trong công nghiệp. Theo luồng phát triển đó, các dự án nhà thông minh (SmartHome) lần lượt ra đời, góp phần tiện lợi hơn cho cuộc sống hiện đại. Gần đây, những nghiên cứu về nhận dạng giọng nói dần được ứng dụng vào lĩnh vực này. Ở một số nước trên thề giới, việc điều khiển bằng giọng nói đã được nghiên cứu, ứng dụng vào đời sống và sản xuất đã ra đời vài năm trở lại đây. Như ở MỸ đã được ứng dụng để điều khiển robotcam trong y khoa. Riêng ở nước ta lĩnh vực này còn khá mới.

Ngoài ra trong những bộ phim khoa học viễn tưởng, chúng ta rất thường hay thấy cảnh con người tương tác với các thiết bị điện tử bằng giọng nói. Giấc mơ này đã có từ khá lâu và bây giờ nó đang dần trở thành hiện thực. Những chiếc smartphone, tablet giờ đây đều có tính năng điều khiển bằng giọng nói, một số sản phẩm thậm chí còn trả lời lại người dùng như thể hai người đang nói chuyện với nhau. Sự ra đời của công nghệ điều khiển bằng giọng nói đã trở thành một xu hướng mới mẻ trong thị trường ứng dụng di động, đặc biệt là những app trước đây đòi hỏi người dùng phải gõ và nhập liệu nhiều.

Thấy được khả năng phát triển và nhu cầu tìm hiểu về điều khiển bằng giọng nói của chính bản thân và của những người yêu thích mong muốn được sử dụng công nghệ này, em đã bắt tay vào thực hiện nghiên cứu đề tài: **“Điều khiển thiết bị bằng giọng nói”.**

# Giới thiệu chung

## Đặt vấn đề

#### Vì sao lại là giọng nói?

Có cả trăm, cả nghìn ứng dụng cho phép người dùng tìm kiếm, viết email, ghi chú và đặt lịch hẹn trên smartphone. Thế nhưng, với một số người, việc sử dụng bàn phím nhỏ xíu trên điện thoại là không tiện lợi, thậm chí là rất khó chịu. Tốc độ gõ của bạn có thể rất nhanh và chính xác, nhưng cũng có những người như nổi điên lên khi phải xóa đi viết lại chỉ một chữ duy nhất chỉ vì họ bấm nhầm sang những phím bên cạnh. Giọng nói là giải pháp hợp lý nhất tính đến thời điểm hiện tại. Người dùng không chỉ xài giọng nói của mình như một phương thức nhập liệu mà chính bản thân thiết bị cũng có thể xài giọng nói để đọc ra những thông tin cần thiết. Đó là chưa kể đến lợi ích to lớn mà những phần mềm dựa trên giọng nói có thể mang lại cho những người khiếm thị. Họ có thể tận hưởng những tiến bộ công nghệ tương tự như những gì mà một người bình thường có thể làm, không còn khoảng cách xuất hiện do những khiếm khuyết về giác quan.

Chúng ta đang trong thời đại công nghệ thông tin, các chương trình giao tiếp người dùng ngày càng đòi hỏi sự thân thiện và hiệu năng mạnh mẽ. Nhu cầu giao tiếp với thiết bị máy bằng tiếng nói trở nên cần thiết, đó là phương thức giao tiếp thông minh và tự nhiên nhất. Trong những năm gần đây, công nghệ này đã có mặt trên rất nhiều thiết bị và chủng loại, từ máy tính đến điện thoại di động, và các thiết bị nhúng khác, các thiết bị ngày càng trở nên nhỏ hơn về kích thước. Tuy nhiên các thiết bị càng nhỏ cũng làm giới hạn về chức năng. Tiếng nói có khả năng điều khiển và tương tác phức tạp với hệ thống nhúng. Nhận diện tiếng nói được phân loại như nhận diện các từ đã được nối với với nhau và nhận biết từng từ một cách độc lập. Đối với hệ thống nhúng thì sử dụng nhận diện từng từ độc lập với nhau có hiệu quả hơn cả.

Tất nhiên, hiện nay công nghệ giọng nói vẫn chỉ mới ở giai đoạn đầu chứ chưa thể nào thay thế hoàn toàn bàn phím ảo/vật lý hoặc các nút trên màn hình. Tuy nhiên, chúng ta đang dần tiến đến một kỉ nguyên hiện đại hơn, các ứng dụng giọng nói cũng dần dần được hoàn thiện. Hãy nhìn vào [Google Voice](https://tinhte.vn/tags/google-voice/) Input, Apple [Siri](https://tinhte.vn/tags/siri/) hay mới đây là Microsoft [Cortana](https://tinhte.vn/tags/cortana/) là bạn sẽ dễ dàng thấy được xu hướng này.Càng ngày những phần mềm [nhận dạng giọng nói](https://tinhte.vn/tags/nhan-dang-giong-noi/) càng thông minh hơn, khả năng nhận dạng chính xác hơn, thông tin trả về cũng hữu ích và đa dạng hơn. Người ta còn áp dụng cả những kĩ thuật như data mining (khai thác dữ liệu theo chiều sâu), machine learning (cho phép máy móc tự học hỏi thói quen, hành vi của người dùng) nhằm cải thiện hiệu suất làm việc của công nghệ nhận dạng giọng nói nữa.

## Những khái niệm cơ bản về giọng nói:

- Giọng nói là một hiện tượng phước tạp. Hiếm có người nào biết được làm sao giọng nói được sinh ra và hiểu được nó. Chúng ta thường ngờ ngệch nhận giọng nói được xây dựng với những từ, và với mỗi từ bao gồm những âm. Nhưng đáng tiếc sự thật là rất khác. Giọng nói là một xử lý động lực học mà không có những phần đặc biệt rõ ràng. Nó rất hữu ích trong việc biên tập âm thanh và nhìn vào bản ghi âm của giọng nói và lắng nghe nó.

Hình 1‑1 Ví dụ bản ghi âm giọng nói

Tất cả các mô tả hiện đại của giọng nói là một số mức độ xác suất. Điều đó có nghĩa là không có ranh giới nhất định giữa các đơn vị, hoặc giữa các từ. Việc chuyển đổi giọng nói sang văn bản và một vài ứng dụng khác của giọng nói là không bao giờ đúng 100%. Đó là một ý tưởng khá tuyệt vời dành cho những nhà phát triển phần mềm, người mà thường làm việc với những hệ thống định luận. Và nó tạo ra rất nhiều các vấn đề riêng chỉ có trong Công nghệ xử lý giọng nói.

#### Cấu trúc của lời nói:

Lời nói là một dòng âm thanh liên tục với các trạng thái ổn định và bất ổn hòa lẫn vào nhau. Trong luồng của các trạng thái này, người ta có thể định nghĩa các lớp có âm thanh giống nhau nhiều hơn hoặc ít hơn, hoặc các tiếng. Các từ được hiểu là được dùng để tạo ra các tiếng, nhưng chắc chắn rằng điều này không đúng. Các thuộc tính âm thanh của một dạng sóng âm trả về cho một tiếng có thể dựa trên nhiều nhân tố đa dạng - nội dung các tiếng, người nói, các nói, và nhiều thứ khác. Tổ chức tạo ra các phụ âm định nghĩa các tiếng rất khác với các tiêu chuẩn trình bày của họ. Kế đó, vì sự chuyển tiếp giữa các từ còn cung cấp nhiều thông tin hơn các âm vùng, các nhà phát triển thường nói về “diphones” (“luyến”)– các phần của âm giữa hai tiếng liên tục. Đôi khi, các nhà phát triển nói về các đơn vị phụ âm – khác với các trạng thái phụ của tiếng. Thường có ba hoặc hơn ba phần của một trạng thái tự nhiên có thể dễ dạng nhận thấy. Phần đầu của tiếng phụ thuộc vào tiếng trước nó, phần giữa cố định và phần tiếp theo dựa vào các tiếng theo sau. Đó là lý do vì sao mà thường có 3 trạng thái trong một tiếng được chọn trong mô hình nhận dạng Markov ẩn.

Đôi khi, các tiếng được cho là có nội dung. Có những tiếng 3 âm, hoặc thậm chí là 5. Nhưng chú ý trằng không như đơn âm và hai âm, chúng phù hợp với cùng một vùng sóng âm và như các tiếng. Chúng chỉ khác tên. Đó là tại sao chúng ta thích gọi đối tượng này là senone. Sự phụ thuộc của một senone vào nội dung có thể phức tạp hơn chỉ nội dung bên trái và bên phải. Nó có thể được một hàm khá phúc tạp định nghĩa bằng một cây quyết định, hoặc một vài cách khác. Kế đó, các tiếng tạo nên các cụm từ, giống như các âm tiết. Để minh họa rõ hơn, khi lời nói trở nên nhanh, các tiếng thường bị biến dạng, nhưng các âm tiết thì vẫn giữ nguyên. Hơn nữa, các âm tiết thường bị chi phối bởi sự ngâm nga của người nói. Có các cách khác để tạo nên một cụm từ dựa trên mặt hình thái học trong một ngôn ngữ giàu hình thái hoặc dựa trên ngữ âm. Các cụm từ thường được dùng trong các bộ từ điển mở trong quá trình nhận dạng giọng nói. Cụm từ tạo nên các từ. Các từ thì quan trọng trong nhận dạng lời nói vì chúng có một giới hạn kết hợp giữa các tiếng theo một ý nghĩa đặc biệt. Nếu có 40 tiếng và trung bình một từ có bảy tiếng, có khoảng 40^7 từ. May mắn thay, ngay cả một người có học vấn cũng ít khi dùng hơn 20 ngàn từ trong cuộc sống, điều này làm việc nhận dạng trở nên càng khả thi hơn. Các từ và các tiếng không thuộc ngôn ngữ, thứ chúng ta hay gọi là fillers (tiếng thở, um, uh, tiếng ho), từ nhiều lời nói. Chúng được chia từng khúc dựa vào các khoảng ngắt. Chúng không cần kết hợp các câu có nghĩa.

## Cách thức nhận dạng tiếng nói

Nhận dạng tiếng nói Nhận dạng tiếng nói là một quá trình phức tạp bao gồm nhiều khâu biến đổi. Tín hiệu tiếng nói phát ra là tương tự. Từ quá trình lấy mẫu, lượng tử hóa và mã hóa để thu được tín hiệu số. Các mẫu tín hiệu này được trích chọn đặc trưng. Những đặc trưng này sẽ là đầu vào của quá trình nhận dạng. Hệ thống nhận dạng sẽ đưa ra kết quả nhận dạng. Tín hiệu tiếng nói đầu tiên được tiền xử lý và rút trích đặc trưng. Kết quả thu được sau quá trình này là tập các đặc trưng âm học (acoustic features). Để có thể thực hiện việc so sánh với các tham số đầu vào của hệ thống nhận dạng, trước hết hệ thống phải được huấn luyện và xây dựng các đặc trưng. Trong quá trình huấn luyện, hệ thống dùng các vector đặc trưng được đưa vào để ước lượng, tính toán các tham số cho các mẫu tham khảo. Một mẫu tham khảo chính là bản mẫu dùng để so sánh và nhận dạng, các mẫu tham khảo này mô phỏng cho một từ, một âm tiết, hoặc một âm vị. Trong quá trình nhận dạng, dãy các vector đặc trưng được so sánh với các mẫu tham khảo. Sau đó, hệ thống tính toán độ tương đồng của dãy vector đặc trưng và mẫu tham khảo. Việc tính toán độ tương đồng được được thực hiện bằng cách áp dụng các thuật toán đã được chứng minh hiệu quả như thuật toán Vitertbi (trong Markov ẩn). Mẫu có độ tương đồng cao nhất là kết quả của quá trình nhận dạng. Có các loại hệ thống nhận dạng là nhạn dạng liên tục, nhận dạng từ tách biệt, nhận dang phụ thuộc ngưới nói và độc lập người nói. Một số phương pháp nhận dạng như : phương pháp âm học – ngữ âm học, phương pháp nhận dạng mẫu và phương pháp trí tuệ nhân tạo.

## Các khái niệm & thuật ngữ

**Acoustic model:** mô hình âm học – là bản lưu các âm kèm với kí tự text.

VD: chữ A -> đọc là ah, được lưu theo định dạng từ điển lexicon.

**Cepstra / Cepstral/ Cepstrum:** Là một mô hình vec-tơ đặc trưng, được dùng riêng cho kỹ thuật nhận dạng giọng nói. Hệ thống sẽ chuyển giọng của chúng ta thành mô hình này để tiện cho việc phân tích, có thể hiểu như hệ thống sẽ chuyển đổi lời nói của chúng ta thành một dãy các biến số. Khi một biến số thay đổi, nghĩa là chúng ta nói một từ khác.

**Corpus (Corpora):** tập lục – là một tập hợp các file giọng nói được thu sẵn để giúp máy học tốt hơn. Corpus này có thể là 1 tập tin dài có tiếng nói được thu âm sẵn (vd: các file của đài tiếng nói việt nam VOV, dài nhiều giờ đồng hồ). Máy sẽ cố gắng nhận dạng các tập tin này, và lưu lại các kết quả có thích phù hợp cao, và biến đó thành các âm chuẩn làm mẫu. Lấy ví dụ: trong cơ sở dữ liệu ta lưu từ “một” với tốc độ đọc vừa phải làm chuẩn, khi tìm thấy 1 từ “một” có tốc độ đọc nhanh hơn một tí, với cao độ và âm sắc khác đi một tí, máy sẽ cho điểm tương thích cho từ một này, đạt đến mức nhất định, hệ thống sẽ chấp nhận và lưu lại. Sau này khi ta đọc từ “một” nhanh hơn một tí, máy sẽ lập tức nhận ra ngay.

**HMM: Hidden Markov Models** - mô hình markov ẩn – là một mô hình dùng để phân tích các âm theo đặc trưng và giúp hệ thống xác định sự khác biệt giữa các âm thanh.

**Language model:** mô hình ngôn ngữ - đây như là 1 bộ từ điển sự kết hợp giữa các từ có thể có, để giúp hệ thống tiên đoán chính xác hơn. VD: từ “trái” sẽ có các từ đi sau như “cam”, “mận” … Nhờ có mô hình ngôn ngữ mà hệ thống nhận dạng chính xác hơn Linguist: Cơ sở dữ liệu của Sphinx, lưu trữ các âm thanh chuẩn làm mẫu để so sánh, phân tích và nhận dạng. Gồm 2 phần là: accoustic model và language model.

**Lattice:** lưới – là một đồ thị có hướng diễn tả các biến trong quá trình nhận dạng. Thường là chọn ra các tổ hợp không có trong thực tế nhất: Trong trường hợp đó đo, lưới (lattice) đóng vai trò trung gian để trình bày kết quả.

**N-best list:** danh sách các biến tốt nhất giống như lattice – lưới, nhưng các tổ hợp kết quả thưa hơn lattice.

Speech database: Cơ sở dữ liệu lời nói – là một tập hợp các đoạn ghi âm điển hình lấy từ các cơ sở dữ liệu nhiệm vụ. Nếu chúng ta phát triển hệ thống đối thoại, nó có thể đối thoại với các đoạn ghi âm của người dùng. Đối với hệ thống chính tả, nó có thể đọc các đoạn ghi âm. CSDL lời nói thường được dùng để huấn luyện, đồng điệu và kiểm tra hệ thống giải mã.

**Text databases:** các CSDL văn bản là các mẫu văn bản thu gom để dùng cho việc huấn luyện mô hình ngôn ngữ và nhiều ứng dụng khác. Thông thường, CSDL văn bản được thu gom trong một mẫu văn bản. Vấn đề là chuyển các tài liệu hiện tại (PDF, trang web, bản scan) thành các mẫu văn bản nói.

**Word confusion networks:** mạng các từ nghi vấn là các lưới lattice mà trật tự chặt chẽ của các nốt được lấy trong các cạnh của lưới lattice.

# Cơ sở lý thuyết

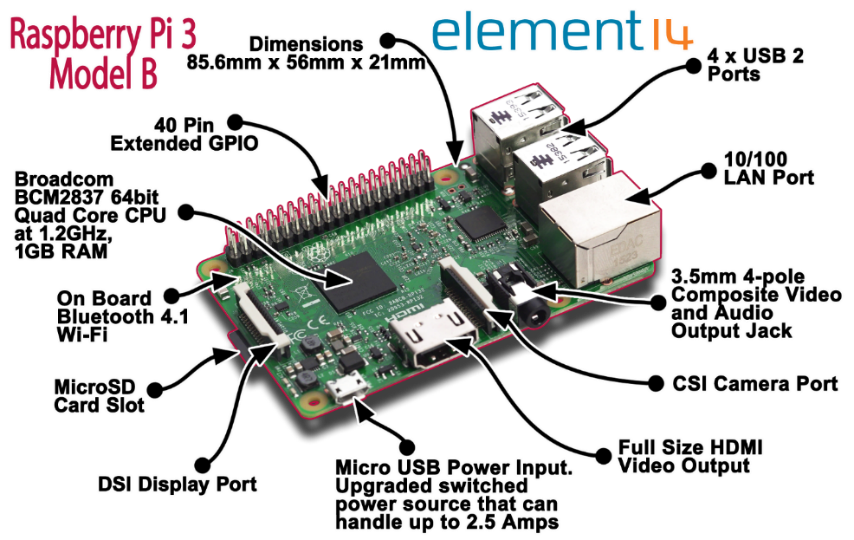
## Phương pháp nhận diện sử dụng

Thông thường, nhận dạng tiếng nói là một loại mẫu nhận dạng dựa trên huấn luyện và nhận dạng. Phương pháp sử dụng là HMM (Hidden Markov Model) và thư viện sử dụng là bộ nhận dạng HTK. Tín hiệu tiếng nói người dùng được lấy bởi micro USB đã được kết nối với hệ thống. Để xử lý tín hiệu nhận dạng bài báo cáo sử dụng bộ thư viện nhận dạng tiếng nói CMU Sphinx nhận dạng các câu lệnh điều khiển thiết bị. Chương trình điều khiển được viết bằng ngôn ngữ Python và được biên dịch trên máy tính nhúng Raspberry Pi để điều khiển các thiết bị đèn, quạt, tivi trực tiếp qua hàng chân GPIO (General Purpose Input/ Output) trên Raspberry Pi.

## Raspberry là gì?

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

### Cấu hình

Phiên bản Raspberry Pi đầu tiên được phát hành tháng 2 năm 2012, và tới nay đã có nhiều phiên bản khác nhau, với sự nâng cấp của phần cứng, cũng như hướng tới những mục tiêu khác nhau. Phiên bản theo thứ tự ra mắt là : Pi A → Pi A+ → Pi 1 B → Pi 1B+ → Pi 2B → Pi Zero → Pi 3B

Hình 2‑1 Cấu hình phần cứng

Cấu hình của PI3

Đây là cấu hình của một chiếc Pi3 Model B (cấu hình phổ biến, được nhiều người sử dụng hiện tại)

* Broadcom BCM2837 chipset running at 1.2 GHz
* 64-bit quad-core ARM Cortex-A53
* 802.11 b/g/n Wireless LANBluetooth 4.1 (Classic & Low Energy)
* Dual core Videocore IV® Multimedia co-processor
* 1 GB LPDDR2 memory
* Supports all the latest ARM GNU/Linux distributions and Windows 10 IoT
* MicroUSB connector for 2.5 A power supply
* 1 x 10/100 Ethernet port
* 1 x HDMI video/audio connector
* 1 x RCA video/audio connector
* 4 x USB 2.0 ports
* 40 GPIO pins
* Chip antenna
* DSI display connector
* MicroSD card slot
* Dimensions: 85 x 56 x 17 mm

### Raspberry Pi có thể dùng để làm gì?

Người ta có thể dùng Pi làm các thứ như:

* Đầu coi phim HD giống như Android Box, hỗ trợ KODI đầy đủ.
* Máy chơi game cầm tay, console, game thùng. Chơi như máy điện tử băng ngày xưa, giả lập được nhiều hệ máy.
* Cắm máy tải Torrent 24/24.
* Dùng làm VPN cá nhân.
* Biến ổ cứng bình thường thành ổ cứng mạng (NAS).
* Làm camera an ninh, quan sát từ xa.
* Hiển thị thời tiết, hiển thị thông tin mạng nội bộ...
* Máy nghe nhạc, máy đọc sách.
* Làm thành một cái máy Terminal di động có màn hình, bàn phím, pin dự phòng để sử dụng mọi lúc mọi nơi, dò pass Wi-Fi...
* Làm thiết bị điều khiển Smart Home, điều khiển mọi thiết bị điện tử trong nhà.
* Điều khiển robot, máy in không dây từ xa, Airplay...
* ...

#### Sau khi mua về thì phải cài hệ điều hành, có nhiều hệ điều hành cho Pi

**Raspbian:** OS chính thức, giao diện giống như Windows/Mac/Linux.

Ngoài ra còn có Ubuntu, Windows 10 IoT, RiscOS...

Raspbian là OS chính thức dành cho Pi. Cách cài đặt rất đơn giản, chỉ cần tải Raspbian về, giải nén, copy vào thẻ nhớ microSD rồi gắn thẻ vào máy Pi cho nó chạy cài đặt là xong.

Qua liệt kê phía trên các ta có thể thấy Pi được hỗ trợ rất nhiều hệ điều hành cho nhiều mục đích khác nhau, từ giải trí, học tập, nguyên cứu, làm các ứng dụng IOT,v.v... chính sự đa dạng đó làm cho Pi3 có sức hút rất lớn về mặt ứng dụng cho cuộc sống, đặc biệt cho những ai yêu thích và làm việc về công nghệ.

## Bộ công cụ CMU Sphinx là gì?

CMU là viết tắt của Carnegie Mellon University, cái nôi của CMUSphinx. Trên thực tế, Sphinx không phải tài sản riêng của CMU, mà là thành quả của tập thể cộng đồng đến từ nhiều quốc gia trên thế giới.

CMUSphinx được phát triển chủ yếu trên 2 ngôn ngữ: C# và Java. Và dựa trên tiền đề là mô hình Markov ẩn để lưu trữ và xử lý thông tin.

Công cụ Sphinx là một nền tảng mã nguồn mở, đang được rất nhiều chuyên gia sử dụng làm công cụ nhận dạng tiếng nói. Sphinx là một công cụ nhận dạng tiếng nói rất mạnh mẽ, có tính môđun hóa cao. Mỗi thành phần biểu diễn một môđun có thể dễ dàng được thay thế, cho phép các nhà nghiên cứu thực nghiệm một môđun khác mà không cần phải thay đổi các thành phần còn lại của hệ thống. Sphinx đã được sử dụng trong nhiều hệ thống nhận dạng như: cairo, jvoicexml,… các chương trình điều khiển như SpeechLion, VoiceKey… Sự kết hợp giữa phần cứng và các thành phần đã tạo ra một hệ thống nhỏ gọn có thể điều khiển bật/tắt một số thiết bị điện gia dụng như: đèn, quạt và tivi. Đặc biệt là truyền tín hiệu điều khiển thiết bị không dây nhằm tạo sự linh động, dễ dàng lắp đặt.

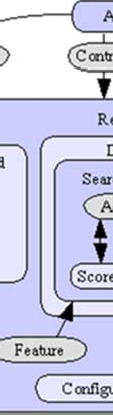
### Nhận dạng giọng nói bằng thư viện CMUSphinx

Sphinx là một bộ thư viện nhận dạng tiếng nói mạnh mẽ và được sử dụng rất nhiều trong các ứng dụng trong cuộc sống. Đó là nhờ các đặc điểm:

* Hỗ trợ nhận dạng tiếng nói ở chế độ trực tiếp hoặc chia lô, có khả năng nhận dạng tiếng nói rời rạc và liên tục.
* Là một hệ thống nhận dạng đồ sộ nhưng có khả năng tháo lắp rất linh động**.** Hỗ trợ sẵn đầy đủ các tính năng đáp ứng nhu cầu nhận dạng như xây dựng các bộ lọc, các hàm cửa sổ, các phép biến đổi…
* Hỗ trợ nhiều mô hình ngôn ngữ dạng ASCII và các phiên bản nhị phân của unigram, bigram, trigram, J va Speech API Grammar Format (JSGF) và ARPAformat FST grammars.
* Xây dựng sẵn các thuật toán tìm kiếm tối ưu (breath first, word pruning), dễ dàng tinh chỉnh cho phù hợp với nhu cầu nhận dạng.

Kiến trúc bộ thư viện nhận dạng gồm các thành phần sau:

Hình 2‑2 Kiến trúc tổng quát Sphinx



**Bộ ngoại vi (FrontEnd):** Xử lý tín hiệu từ bên ngoài, thực hiện qua một số bộ lọc và xử lý dữ liệu cho ra kết quả là một tập các vector đặc trưng.

**Bộ ngôn ngữ (Linguist):** bằng các công cụ và phương pháp ngôn ngữ, đọc vào các **t**ập tin cấu tr c của một ngôn ngữ rồi mô hình hóa chúng vào đồ thị. Ở bộ này cấu tạo khá phức tạp vì nó quy định hầu như toàn bộ phạm vi ngôn ngữ mà chúng ta cần nhận dạng, nó gồm các thành phần nhỏ sau**: Mô hình ngôn ngữ:** Đọc vào tập tin cấu trúc ngôn ngữ ở cấp độ là các từ. Thành phần này có vai trò quan trọng xác định những thứ hệ thống cần nhận dạng. Cấu trúc ngôn ngữ sẽ được mô hình hóa ở đây theo hai mô hình: graph-driven grammar và Stochastic N-Gram; **Mô hình graph-driven grammar:** Biễu diễn một đồ thị từ có hướng. Mỗi nút biểu diễn một từ đơn và mỗi cung là xác suất dịch chuyển sang một từ; **Mô hình stochastic N-Gram:** Mô hình này cung cấp các xác suất cho các từ được c o dựa vào việc quan sát N-1 từ đứng trước.

**SimpleWordListGrammar:** Định nghĩa một từ dựa trên danh sách các từ. Một tham số tùy chọn chỉ ra ngữ pháp có lặp hay không. Nếu không lặp, ngữ pháp sẽ dùng được dùng cho một nhận dạng từ tách biệt. Nếu lặp, nó sẽ được dùng để hỗ trợ liên kết nhận dạng từ tầm thường.

**JSGFGrammar:** Hỗ trợ JavaTM Speech API Grammar Format (JSGF), định nghĩa một biểu diễn theo BNF, độc lập nền tảng, Unicode của các ngữ pháp.

**LMGrammar:** Định nghĩa một ngữ pháp dựa trên một mô hình ngôn ngữ thống kê. LMGrammar phát sinh một nút ngữ pháp mỗi từ và làm việc tốt với các unigram và bigram, xấp xỉ 1000 từ. FSTGrammar: Hỗ trợ một bộ chuyển đổi trạng thái giới hạn (finite-state tranducer) trong định dạng ngữ pháp ARPA FST. SimpleNGramModel Cung cấp hỗ trợ cho các mô hình ASCII N-Gram trong định dạng ARPA. SimpleNGramModel không cố làm tối ưu việc sử dụng bộ nhớ, do đó nó làm việc tốt với các mô hình ngôn ngữ nhỏ. LargeTrigramModel: Cung cấp hỗ trợ các mô hình N- Gram đúng được phát sinh bởi CMU-Cambridge Statictical Language Modeling Toolkit. LargeTrigramM del tối ưu việc lưu trữ bộ nhớ, cho phép nó làm việc với các tập tin rất lớn, trên 100MB. **B**ộ từ điển: Thành phần này cung cấp cách phát âm cho các từ ta đã xây dựng trong mô hình ngôn ngữ. Mô hình âm học: Cung cấp một ánh xạ giữa một đơn vị tiếng nói và một HMM có thể được đánh giá dựa vào các đặc trưng được cung cấp bởi bộ ngoại vi. Các ánh xạ có thể đưa thông tin vị trí của từ và ngữ cảnh từ thành phần mô hình ngôn ngữ. Định nghĩa ngữ **c**ảnh này được xây dựng từ cấu trúc ngữ pháp của mô hình ngôn ngữ.

**Đồ thị tìm kiếm (Search Graph)**: Là kết quả mà bộ ngôn ngữ phát sinh được cuối cùng để đưa v o sử dụng trong bộ giải mã. Đồ thị tìm kiếm này là một đồ thị có hướng, trong đó mỗi nút được gọi là một trạng thái tìm kiếm (SearchState), biểu diễn một trong hai trạng thái: phát hoặc không phát (emitting state hay non-emitting state). Và các đường cung biểu diễn các trạng thái biến đổi có thể, trên các cung này có các giá trị xác suất được tính toán từ mô hình âm học: biểu diễn khả năng chuyển từ trạng thái này đến trạng thái kia. Bộ giải mã (Decoder): Sử dụng các đặc trưng (Features) từ bộ ngoại vi kết hợp với đồ thị tìm kiếm được phát sinh từ bộ ngôn ngữ để tiến hành giải mã và áp dụng các thuật toán suy ra kết quả nhận dạng. Nhiệm vụ của thành phần quản lý tìm kiếm là nhận dạng các tập các vector đặc trưng để tìm ra ánh xạ tương ứng của nó trong đồ thị tìm kiếm. Để đáp ứng tìm ra kết quả chính xác t ong đồ thị tìm kiếm khi xử lý kết quả, Sphinx cung cấp các tiện ích có khả năng phát sinh lưới và các đánh giá độ tin cậy từ kết quả. Và thêm đặc điểm nữa khác các hệ thống khác là không gian tìm kiếm trong Sphinx có thể được tinh chỉnh thay đổi trong quá tình tìm kiếm để tăng hiệu suất tìm kiếm. Ngoài ra để nâng cao hiệu suất của kết quả nhận dạng, Sphinx còn bổ sung thêm các công cụ hỗ trợ cho việc đánh giá kết quả nhận được, đó là thành phần đánh giá (Scorer) và thành phần cắt tỉa (Pruner). Nói về thành phần Scorer thì nó là một module dùng để ước lượng xác suất của trạng thái khi cung cấp các giá trị mật độ trạng thái xuất hiện. Khi thành phần quản lý tìm kiếm yêu cầu đánh giá điểm số cho một trạng thái, nó sẽ gọi đến thành phần Scorer, nó sẽ phân tích các thông tin đặc trưng của trạng thái đó rồi áp dụng các phép toán để tính điểm số.

## Machine Learning - Nhận diện giọng nói

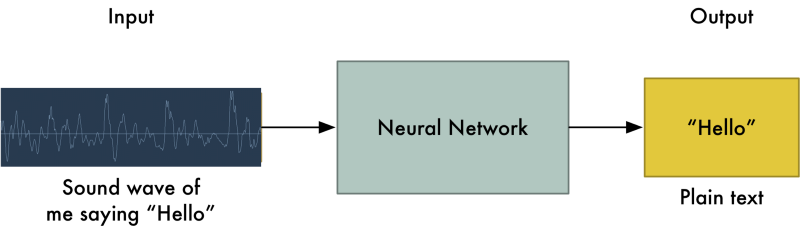
Nhận diện giọng nói đang xâm nhập vào cuộc sống hiện đại. Nó được cài đặt trong những chiếc điện thoại, điều khiển trò chơi hay những chiếc đồng hồ thông minh. Chỉ với khoảng $50, bạn có thể có Amazon Echo Dot - một chiếc hộp thần kỳ cho phép bạn đặt pizza, nhận thông tin dự báo thời tiết hoặc thậm chí mua những vật dụng - chỉ bằng cách đưa ra mệnh lệnh:

Hình 2‑3 Amazon Echo Dot

Echo Dot trở nên phổ biến trong kỳ nghỉ đến nỗi Amazon bị cháy hàng.

Nhưng nhận diện giọng nói đã được biết đến hàng thập kỷ, tại sao chỉ đến bây giờ, công nghệ mới thực sự bùng nổ? Sự ra đời của [Deep Learning](https://viblo.asia/p/machine-learning-that-thu-vi-2-tu-dong-tao-game-mario-WAyK81o9ZxX#viet-ra-mot-cau-chuyen-6) đã giúp nhận diện giọng nói chính xác, thậm chí ở ngoài môi trường phòng lab.

Andrew Ng đã dự đoán từ lâu rằng ngay khi độ chính xác của nhận diện giọng nói đạt ngưỡng 99%, nó sẽ trở thành phương thức giao tiếp chủ yếu với máy tính. Và nhờ có **Deep Learning**, cuối cùng chúng ta đã có thể chạm tới ngưỡng này.

Chúng ta có thể đoán rằng: chúng ta chỉ cần truyền đoạn ghi âm vào mạng nơron và đào tạo nó để tạo ra "bản dịch”

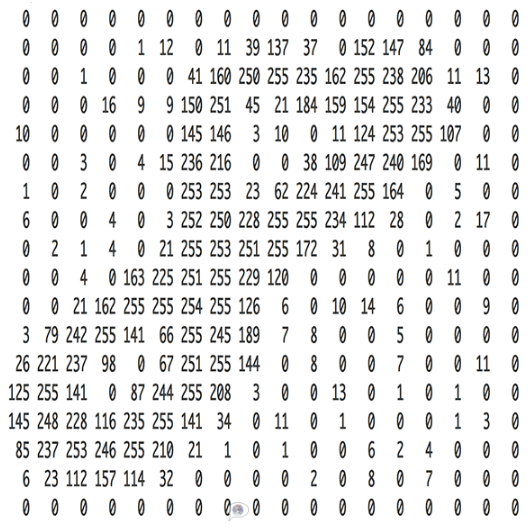
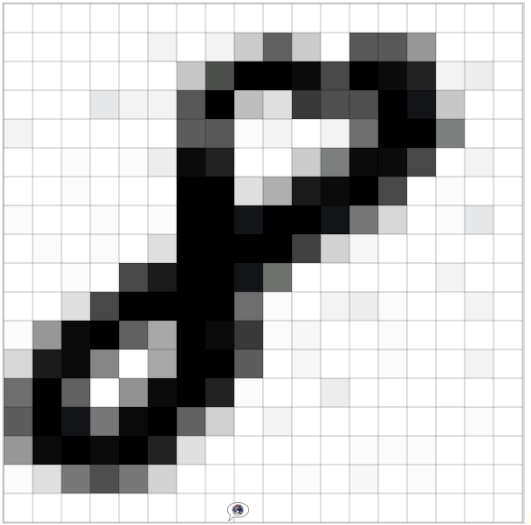
Hình 2‑4 Truyền thông tin mạng Noron

Đó cũng là điều mà nhận diện giọng nói với deep learning hướng tới, nhưng chúng ta chưa đạt đến trình độ đó (có thể cần phải thêm vài năm nữa).

Vấn đề lớn nhất chính là tốc độ nói biến thiên. Một người có thể nói "Hello" rất nhanh và người khác nói "heeeellllllloooooo!" cực chậm, tạo ra âm thanh dài hơn với nhiều dữ liệu hơn. Cả 2 âm đều nên được nhận dạng chính xác là từ một - "hello!". Tự động chỉnh file âm thanh với nhiều biến thể độ dài khác nhau của từng từ để tạo ra văn bản đồng nhất lại khá khó.

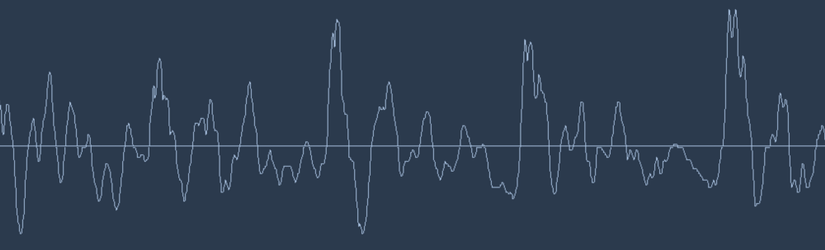
Để xử lý vấn đề này, chúng ta sẽ sử dụng một số kỹ thuật đặc biệt và thêm một vài bước vào mạng deep learning.

### Chuyển âm thanh thành số

Bước đầu tiên trong nhận diện giọng nói khá rõ ràng - chung ta cần truyền sóng âm vào máy tính.

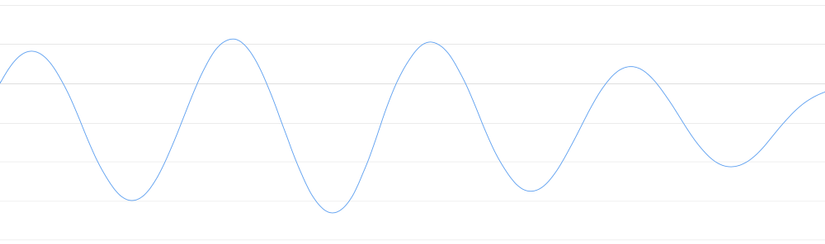
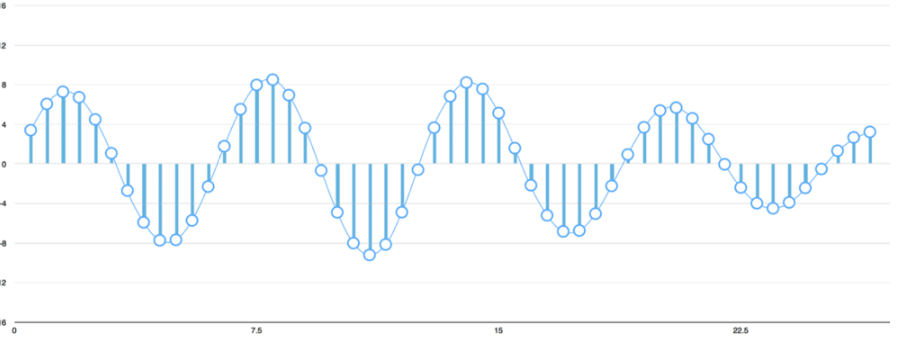
Hình 2‑5 Ví dụ tập hợp giá trị điểm ảnh

Chúng ta coi hình ảnh là tập hợp giá trị, với mỗi giá trị đại diện cho độ sáng của điểm ảnh, để truyền vào mạng nơron.

Nhưng âm thanh được truyền qua sóng âm. Làm thế nào chúng ta chuyển sóng âm thành số? Hãy sử dụng đoạn âm nói "Hello" dưới đây:

Hình 2‑6 Biểu diễn đoạn âm “Hello”

Sóng âm có **một chiều** dữ liệu. Ở mỗi thời điểm, chúng có một giá trị cao độ. Hãy phóng to một đoạn nhỏ sóng âm để nhìn rõ hơn:

Để chuyển sóng âm thành số, chúng ta chỉ cần ghi lại độ cao của sóng ở từng khoảng:

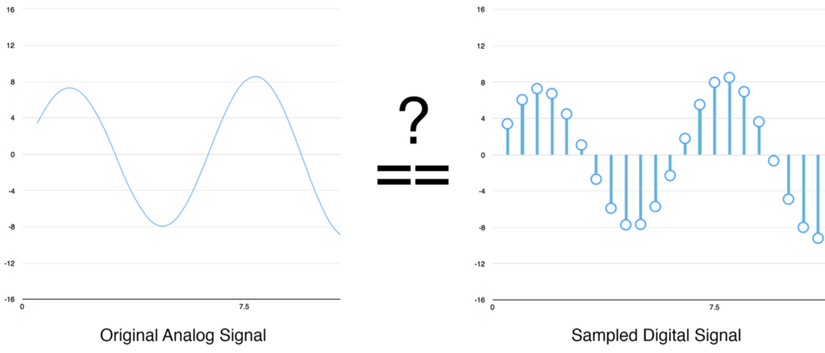
Hình 2‑7 Lấy mẫu

Hình 2‑8 Phóng to 1 đoạn nhỏ đoạn âm “Hello”

Phương pháp này gọi là **sampling** - lấy mẫu. Chúng ta đọc mẫu mỗi 1/1000s và ghi lại con số đại diện chiều cao cùa sóng âm. Đây chính là file .wav khi không bị nén. Những âm thanh chất lượng tốt được ghi ở tần số 44.1khz (44,100 lần đọc mỗi giây). Nhưng với nhận diện giọng nói, tốc độ lấy mẫu ở 16khz (16,000 mẫu mỗi giây) là quá đủ.

#### Lấy mẫu liệu có được chính xác?

 lấy mẫu chỉ tạo ra đồ thị xấp xỉ so với sóng âm, bởi vì nó chỉ đọc dữ liệu theo từng khoảng. Liệu chúng ta có bị mất dữ liệu giữa mỗi lần đọc?

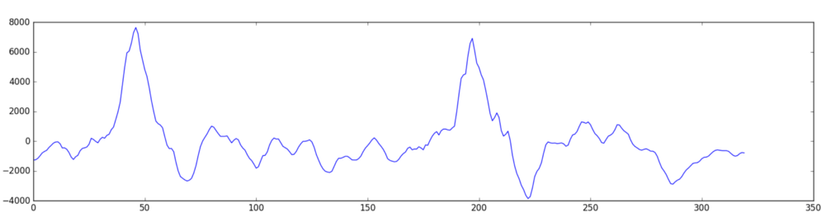
Nhờ có [lý thuyết Nyquist](https://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem), chúng ta có thể sử dụng toán học để tái tạo chính xác sóng âm gốc từ những mẫu tách biệt - miễn là chúng ta lấy mẫu với tần số **gấp đôi** tần số âm chúng ta muốn ghi lại. Không phải cứ lấy mẫu với tần số càng cao thì chất lượng âm thanh càng tốt .

Hình 2‑9 Tín hiệu lấy mẫu có chính xác?

### Tiền xử lý dữ liệu mẫu âm thanh

Chúng ta đã có dãy số với mỗi số đại diện cho cao độ âm tại 1/16,000s.

Ta có thể truyền những số này vào mạng nơron, nhưng cố gắng nhận diện cấu trúc âm thanh trực tiếp bằng những mẫu này rất khó. Thay vào đó, chúng ta giải quyết vấn đề dễ hơn bằng cách **tiền xử lý** dữ liệu. Hãy bắt đầu nhóm mẫu âm thanh trong khoảng 20ms. Ghi lại những con số này trong đồ thị giúp chúng ta có ước lượng xấp xỉ về âm thanh gốc trong chu kỳ 20ms:



Bản ghi âm này chỉ khoảng 1/50s. Nhưng thậm chí một đoạn ghi âm rất ngắn là một mớ hỗn độn cao độ âm khác nhau. Có những âm thấp, âm trung và thậm chí cả âm cao. Nhưng cùng với nhau, những âm này tạo lên giọng nói.

Để giúp mạng nơron xử lý dữ liệu dễ hơn, ta tách sóng âm phức tạp này thành từng phần: phần chứa âm thấp, âm cao hơn, cao hơn nữa... Sau đó, ta tính tổng năng lượng ở những những dải tần số (từ thấp đến cao) và kết nối lại tạo ra **fingerprint -** nhận dạng duy nhất cho từng đoạn trích âm thanh. Chúng ta làm điều đó nhờ vào việc sử dụng [**Fourier transform**](https://en.wikipedia.org/wiki/Fourier_transform) trong toán học. Nó chia nhỏ những sóng âm phức tạp thành sóng âm đơn tạo ra nó, và ta có thể tính tổng năng lượng ở mỗi đơn âm.

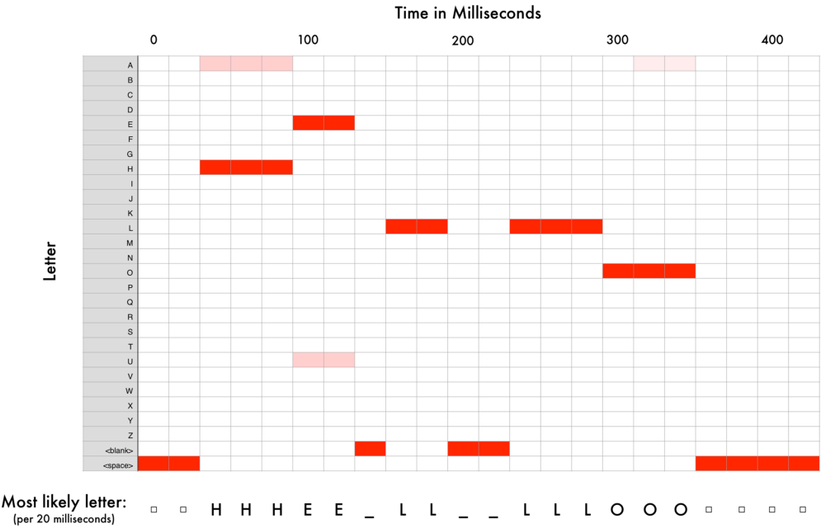
*Sau khi sử dụng lý thuyết Nyquist ở trên, sóng âm đã trở thành một dải liên tục. Và sử dụng Fourier transform, chúng ta lại tách dải liên tục đó ra thành các notes riêng biệt (được tính toán bởi thuật toán Fourier) để tìm ra tổng năng lượng ở từng note.*

### https://viblo.asia/uploads/1db545d0-0ae6-41c7-8936-412e76834739.pngNhận diện ký tự từ đoạn âm ngắn

Hình 2‑10 Mạng Noron hồi quy

Bây giờ chúng ta truyền từng dải âm 20ms vào mạng nơron đa lớp. Với mỗi mảng cắt âm thanh, chúng ta cố gắng tìm ra ký tự đại diện cho âm thanh phát ra.

Chúng ta sử dụng [Recurrent Neural Network - RNN](https://viblo.asia/p/machine-learning-that-thu-vi-2-tu-dong-tao-game-mario-WAyK81o9ZxX#recurrent-neural-network---mang-noron-hoi-quy-4) - mạng nơron hồi quy: kết quả tiên đoán quá khứ có ảnh hưởng tới kết quả tiên đoán trong tương lai. Đó là bởi vì các ký tự có sự liên quan đến nhau. Ví dụ chúng ta đã tìm ra "HEL", thì rất có khả năng chúng ta sẽ nói tiếp "LO". Vì thế, những dự đoán trong quá khứ sẽ giúp dự đoán tương lai được tốt hơn.

Sau khi chạy toàn bộ âm thanh thông qua mạng nơron, chúng ta kết nối mỗi dải âm với một ký tự có khả năng được nói cao nhất. Và đây là bản đồ kết nối của từ "HELLO"

Hình 2‑11 Dự đoán của mạng Noron

Mạng nơron trên dự đoán từ được nói là “HHHEE\_LL\_LLLOOO”, nhưng nó cũng nghĩ có khả năng từ đó là “HHHUU\_LL\_LLLOOO”, hoặc thậm chí là “AAAUU\_LL\_LLLOOO”

Chúng ta có thêm một vài bước để làm sạch kết quả. Đầu tiên, chúng ta bỏ đi những ký tự bị lặp, rồi bỏ đi khoảng trống:

HHHEE\_LL\_LLLOOO => HE\_L\_LO => HELLO

HHHUU\_LL\_LLLOOO => HU\_L\_LO => HULLO

AAAUU\_LL\_LLLOOO => AU\_L\_LO => AULLO

Như vậy, ta có 3 khả năng phân âm là "Hello", "Hullo" và "Aullo". Nếu bạn nói chúng thật to, cả 3 đều nghe giống với "Hello". Bởi vì dự đoán từng ký tự một, mạng nơron tìm ra cách đọc các âm chứ không phải cách viết. Ví dụ: nếu bạn nói "He would not go", máy có thể dịch là "He wud net go".

Thủ thuật ở đây là kết hợp những dự đoán phiên âm này với khả năng xuất hiện trong các văn bản (sách, bài bảo...). Bạn sẽ loại bỏ đi những phiên âm ít có khả năng ngoài thực tế và giữ phiên âm thực tế nhất. Và trong 3 từ "Hello", Hullo" và "Aullo". Rõ ràng, "Hello" có tần xuất cao hơn rất rất nhiều, và đây chính là bản phiên âm chúng ta lựa chọn.

#### Tự xây dựng hệ thống nhận diện giọng nói?

Một trong những điều tuyệt vời nhất về học máy là tính đơn giản. Bạn có 1 tập dữ liệu, truyền vào thuật toán học máy, và đột nhiên, bạn có hệ thống trí thông minh nhân tạo hàng đầu thế giới... Có vẻ đúng? Điều đó đúng trong rất nhiều trường hợp, nhưng không cho ngôn ngữ. Nhận diện ngôn ngữ là một vấn đề khó. Bạn có thẻ vượt qua *thử thách không giới hạn:* chất lượng micro kém, môi trường ghi âm ồn, tiếng dội lại, giọng điệu khác nhau... Tất cả những vấn đề này hiện hữu trong quá trình đào tạo, khiến mạng nơron không có độ chính xác cao.

Đây là một ví dụ khác: Khi bạn nói trong một phòng ồn, bạn tự động tăng cao độ giọng nói trong vô thức, con người chúng ta vẫn hiểu dễ dàng. Nhưng với mạng nơron, nó cần được đào tạo để xử lý các trường hợp đặc biệt này. Và để xử lý tiếng ồn, bạn cần dữ liệu đào tạo khi mà mọi người đang gào thét vào mặt người nói.

Để xây dựng hệ thống nhận diện âm thanh đạt tới trình độ của Siri, Google Now! hay Alexa, bạn cần rất nhiều dữ liệu. Bạn khó có được tập dữ liệu này trừ phi bạn thuê hàng trăm người tạo dữ liệu cho bạn. Và vì người sử dụng chịu đựng rất kém với hệ thống nhận diện không tốt, bạn không thể lờ đi những trường hợp đặt biệt. Không một ai muốn hệ thống nhận diện chỉ đạt 80% độ chính xác.

## Điều khiển theo mức

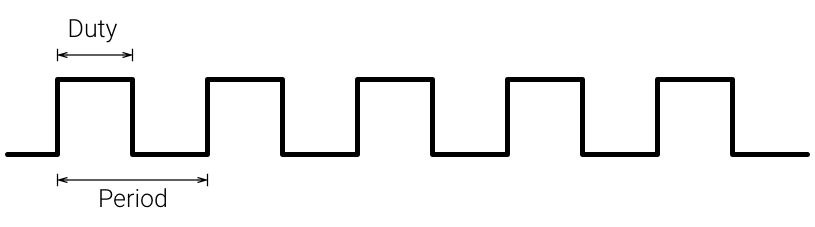
#### Điều chế độ rộng xung (PWM)

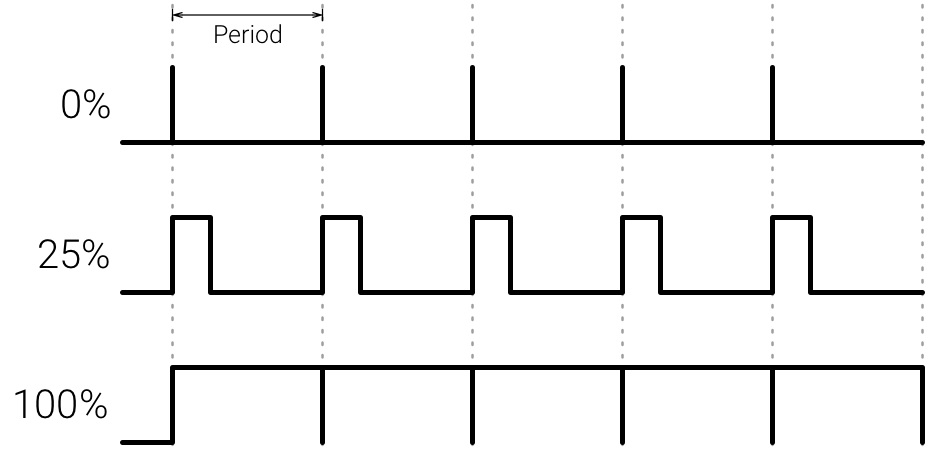
***Xung PWM là gì?***

Không giống như các chân analog Arduino, các chân GPIO của Raspberry Pi đều là các chân kỹ thuật số, chỉ gửi và nhận điện áp cao / thấp. Điều đó có nghĩa là trạng thái nguồn là 1 hoặc 0 và không có gì ở giữa! Ngoài việc bật và tắt đèn LED, nếu bạn muốn giá trị ở giữa giữa như 0,6 (tức là làm mờ ánh sáng!) Ta sẽ cần một số thủ thuậtnhư sử dụng đầu ra PWM tương tự, hoặc mô phỏng nó bằng phần mềm.

Điều chế độ rộng xung PWM ([Pulse Width Modulation](https://en.wikipedia.org/wiki/Pulse-width_modulation)) là một phương pháp thường được sử dụng để để thay đổi điện áp đầu ra trên chân của thiết bị. Giá trị điện áp sẽ được tính theo giá trị trung bình của tín hiệu PWM. Nhờ tính chất điện áp thay đổi nên ta có thể dễ dàng ứng dụng để điều khiển tốc độ dộng cơ điện một chiều, độ sáng của đền LED… Ngoài ra xung PWM còn được sử dụng để điều chỉnh góc quay của động cơ servo.

Tín hiệu PWM gồm có 2 thành phần chính:

* **Tần số**: Số lần lặp lại xung trong 1 giấy (Hz)
* **Độ rộng xung**: Phần trăm ở mức cao trong 1 chu kì xung. (%)

Điều chỉnh độ rộng xung sẽ thay đổi thời gian ở trạng thái cao của tín hiệu trong 1 chu kì.

Hình 2‑12 Điều khiển độ rộng xung

# Thiết lập nhận diện giọng nói

## Chuẩn bị

CMU Sphinx là một gói nhận diện giọng nói tuyệt vời. Vấn đề ở đây là khi thiết lập PocketSphinx có vô số thư viện mà trang web chính yêu cầu tải xuống. Và sau nhiều lần thử nghiệm và sai sót, thì chúng ta chỉ cần 3 thư viện chính:

* SphinxBase là gói cơ sở mà tất cả các Sphinx program khác sử dụng.
* PocketSphinx là trình nhận dạng nhẹ, tuy hơi không chính xác nhưng nó có thể giải mã các cụm từ nhanh hơn.
* PocketSphinx-python.

Trước khi có thể sử dụng Raspberry Pi để nhận dạng giọng nói, ta cần đảm bảo rằng ta có thể ghi lại âm thanh từ micrô USB.

Đầu tiên cắm mic USB và bật nguồn Pi. Khi mọi thứ đã tải xong, ta có thể kiểm tra xem âm thanh USB đã được phát hiện chưa và sẵn sàng sử dụng bằng cách chạy lệnh ***aplay -L***. Nếu thẻ âm thanh USB của chúng ta xuất hiện trong danh sách này thì ta có thể chuyển sang làm mặc định cho hệ thống.

Để thực hiện việc này, hãy sử dụng trình soạn thảo văn bản yêu thích của bạn để chỉnh sửa tệp ***alsa-base.conf*** như sau:

***sudo nano /etc/modprobe.d/alsa-base.conf***

**[contextly\_auto\_sidebar id=”y7PiiZ4b5liqHyGSd7Cxk34ocDcVrFuL”]**

Ta cần thay đổi tùy chọn dòng ***snd-usb-audio index = -2***thành tùy chọn ***snd-usb-audio index = 0***và thêm một dòng bên dưới nó với tùy chọn ***snd\_bcm2835 index = 1*.** Sau khi hoàn tất, lưu tệp và sudo khởi động lại Pi để sử dụng cấu hình mới.

Để kiểm tra các thay đổi, hãy sử dụng lệnh ***arecord -vv --duration = 7 -fdat ~ / test.wav*** để ghi lại đoạn âm thanh ngắn 7 giây từ micrô. Hãy thử phát lại bằng ***aplay ~ / test.wav*** và ta sẽ nghe thấy những gì ta đã ghi lại trước đó qua tai nghe USB.

Nếu nghe bản thu âm thì nó đã sẵn sàng để chuyển sang thiết lập phần mềm.

## Biên dịch phần mềm

Sẽ có một vài gói để cài đặt và một số phần mềm để biên dịch trước khi Pi của chúng ta sẵn sàng lưu trữ phần mềm điều khiển giọng nói.

Đầu tiên, hãy lấy các gói cần thiết cho SphinxBase bằng cách thực hiện lệnh:

**sudo apt-get install libasound2-dev autoconf libtool bison \  
swig python-dev python-pyaudio**

Chúng ta cũng sẽ cần cài đặt một số thư viện Python để sử dụng với ứng dụng demo của chúng ta. Để thực hiện việc này, sử dụng lệnh Python pip với các lệnh sau:

**curl -O https://bootstrap.pypa.io/get-pip.py**

**sudo python get-pip.py**

**sudo pip install gevent grequests**

#### Cài bộ công cụ Sphinx

Bây giờ chúng ta có thể sử dụng thư viện Sphinx.

Để có được SphinxBase, hãy thực hiện các lệnh sau:

**git clone git://github.com/cmusphinx/sphinxbase.git  
cd sphinxbase  
git checkout 3b34d87  
./autogen.sh  
make**

(Thời gian đợi khá lâu)

Chúng ta đã sẵn sàng để chuyển sang PocketSphinx. Để có được PocketSphinx, hãy thực hiện các lệnh sau:

**git clone git://github.com/cmusphinx/pocketsphinx.git  
cd pocketsphinx  
git checkout 4e4e607  
./autogen.sh  
make**

Để cập nhật hệ thống với các thư viện mới, hãy chạy:

**sudo ldconfig**

#### Kiểm tra nhận diện gióng nói

Bây giờ chúng ta chạy thử nghiệm PocketSphinx bằng cách sử dụng **Pocketphinx\_continupt -inmic yes.**

Ta sẽ thấy một cái gì đó như sau, cho biết hệ thống đã sẵn sàng để bạn bắt đầu nói:

***Listening...  
Input overrun, read calls are too rare (non-fatal)***

Chúng ta có thể yên tâm bỏ qua cảnh báo. Hãy tiếp tục và nói!

Khi kết thúc,chúng ta sẽ thấy một số thông tin kỹ thuật cùng với PocketSphinx, dự đoán tốt nhất về những gì đã nói, và sau đó một lời nhắc READY khác cho ta biết nó đã sẵn sàng để nhập thêm.

***INFO: ngram\_search.c(874): bestpath 0.10 CPU 0.071 xRT  
INFO: ngram\_search.c(877): bestpath 0.11 wall 0.078 xRT  
what  
READY....***

Tại thời điểm này, nhận dạng giọng nói đang hoạt động !!!

## Sử dụng Pocketsphinx

Có một số chế độ mà ta có thể định cấu hình cho PocketSphinx. Ví dụ: có thể được yêu cầu nghe một từ khóa cụ thể (nó sẽ cố gắng bỏ qua mọi thứ nó nghe được ngoại trừ từ khóa) hoặc có thể được yêu cầu sử dụng một ngữ pháp mà ta chỉ định (nó sẽ cố gắng phù hợp với mọi thứ nó nghe được giới hạn của ngữ pháp).

Bọn em đang thử sử dụng chế độ ngữ pháp trong ví dụ của mình, với một ngữ pháp mà CLASS thiết kế để cho phép nắm bắt tất cả các lệnh sẽ sử dụng. Tệp ngữ pháp được chỉ định trong JSGF hoặc Jspeech Grammar Format – chúng có cú pháp mạnh mẽ nhưng đơn giản để chỉ định bài phát biểu mà nó dự kiến ​​sẽ nghe theo các quy tắc đơn giản nhất.

Ngoài tệp ngữ pháp, ta sẽ cần thêm ba tệp để sử dụng PocketSphinx trong ứng dụng này:

1. Tệp từ điển sẽ xác định các từ theo cách phát âm.
2. Tệp mô hình ngôn ngữ chứa số liệu thống kê về các từ và thứ tự của chúng.
3. Một mô hình âm thanh được sử dụng để xác định âm thanh tương quan với âm thanh trong từ như thế nào.

### Tạo từ điển

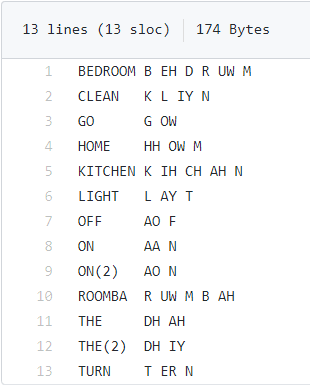
Ví dụ chúng ta để các câu sau đây trong kho văn bản:

* + - * + *turn on the kitchen light*
        + *turn off the kitchen light*
        + *turn on the bedroom light*
        + *turn off the bedroom light*
        + *…*

Chúng ta có thể nhập chúng vào trình soạn thảo văn bản và lưu tệp dưới dạng .txt

Sau khi có tập tin văn bản, hãy sử dụng **lmtool**.

Trang công cụ Imtool của Sphinx:

* <http://www.speech.cs.cmu.edu/tools/lmtool-new.html>

Hình 3‑1 Dictionary để nạp cách đánh vần cho sphinx

### Tạo tập tin ngữ pháp

Hình 3‑2 file language.model.lm

Như đã đề cập trước đó, mọi thứ mà PocketSphinx nghe thấy, nó sẽ cố gắng và phù hợp với các từ ngữ pháp. Kiểm tra định dạng JSGF được mô tả trong ghi chú W3C. Nó bắt đầu bằng một tuyên bố định dạng theo sau là một tuyên bố về tên ngữ pháp. Chúng ta chỉ đơn giản gọi các lệnh.

Chúng ta sử dụng ba quy tắc chính: hành động, đối tượng và lệnh. Đối với mỗi quy tắc, bạn sẽ xác định các mã thông báo, đó là những gì bạn muốn người dùng nói.

Ví dụ: hai mã thông báo cho quy tắc hành động của chúng tôi là TURN ON và TURN OFF.

**<action> = TURN ON  |  
TURN OFF ;**

Tương tự quy tắc \_object\_ chúng tôi xác định là:

**<object> =  KITCHEN LIGHT|  
BEDROOM LIGHT|  
FAN    ;**

Cuối cùng, để chứng minh rằng chúng ta có thể lồng các quy tắc hoặc tạo chúng bằng các mã thông báo rõ ràng, chúng ta cần xác định một lệnh là:

**public <command> = <action> THE <object>  |  
...         ;**

Lưu ý từ khóa puplic phía trước <lệnh>. Điều này cho phép chúng ta sử dụng quy tắc <lệnh> bằng cách nhập nó vào các tệp ngữ pháp khác trong tương lai.

### Khởi tạo bộ giải mã

Chúng ta sử dụng Python làm ngôn ngữ lập trình vì nó dễ đọc, mạnh mẽ và nhờ vào tầm nhìn xa của các nhà phát triển PocketSphinx, nó cũng rất dễ sử dụng với PocketSphinx.

Công việc chính khi nhận dạng giọng nói với PocketSphinx là bộ giải mã. Để sử dụng bộ giải mã, trước tiên chúng ta phải đặt cấu hình cho bộ giải mã sử dụng.

**from pocketsphinx import \***

**hmm = 'cmusphinx-5prealpha-en-us-ptm-2.0/'  
dic = 'dictionary.dic'  
grammar = 'grammar.jsgf'**

**config = Decoder.default\_config()  
config.set\_string('-hmm', hmm)  
config.set\_string('-dict', dic)  
config.set\_string('-jsgf', grammar)**

Khi điều này được thực hiện, việc khởi tạo bộ giải mã cũng đơn giản như bộ giải mã = ​​Decoder (config).

Ví dụ, khi sử dụng thư viện pyAudio để lấy lời nói của người dùng từ micrô để xử lý bởi PocketSphinx. Các chi tiết cụ thể của thư viện này ít quan trọng hơn cho mục đích của chúng ta (điều tra nhận dạng giọng nói) và do đó chúng tôi sẽ chỉ đơn giản chấp nhận rằng pyAudio hoạt động như quảng cáo. Các chi tiết cụ thể để có được bộ giải mã văn bản đầu ra của bộ giải mã là hơi phức tạp, tuy nhiên quy trình cơ bản có thể được chắt lọc theo các bước sau.

*\# Bắt đầu 'utterance'* **decoder.start\_utt()***\# Xử lý một soundbite* **decoder.process\_raw(soundBite, False, False)***\# Kết thúc câu nói khi người dùng nói xong*

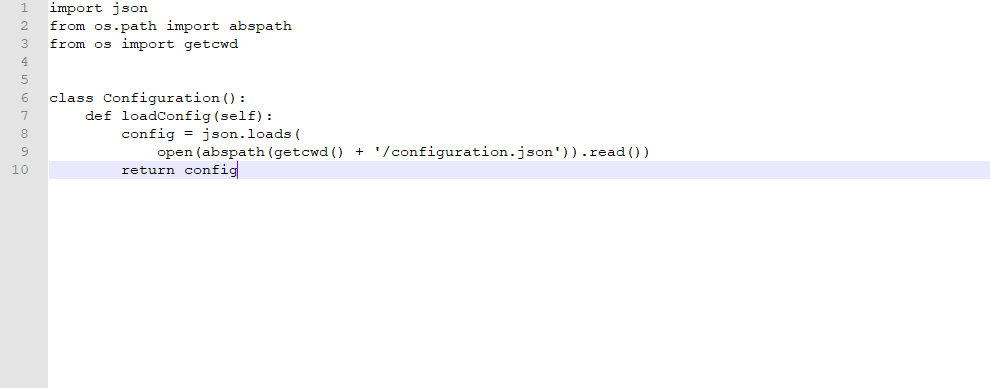
**decoder.end\_utt()***\# Lấy lại giả thuyết (cho những gì đã nói)* **hypothesis = decoder.hyp()***\# Lấy văn bản của giả thuyết* **bestGuess = hypothesis.hypstr***\# In ra những gì đã nói* **print 'I just heard you say:"{}"'.format(bestGuess)**

Có rất nhiều thông số cấu hình khác nhau mà ta có thể thử nghiệm và thử các chế độ nhận dáng khác. Chẳng hạn, điều tra tùy chọn cấu hình **-allphone\_ci PocketSphinx** và tác động của nó đến độ chính xác giải mã.

Hoặc thử phát hiện từ khóa để kích hoạt đèn.

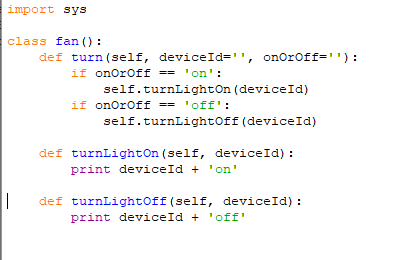
Hoặc thử mô hình ngôn ngữ thống kê, giống như mô hình ngôn ngữ được tạo khi sử dụng lmtool trước đó, thay vì tệp ngữ pháp.

Một điều mà bạn sẽ nhanh chóng nhận thấy là PocketSphinx là một hệ thống nghiên cứu được phát triển tích cực và điều này đôi khi sẽ có nghĩa là bạn cần viết lại ứng dụng của mình để khớp với các API và tên hàm mới.

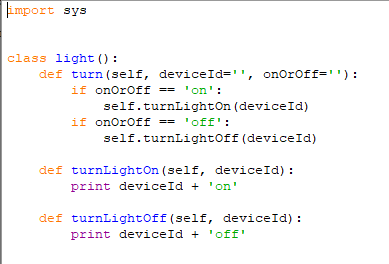


Hình ‑ Configuration.py file nạp thiết bị từ file cấu hình

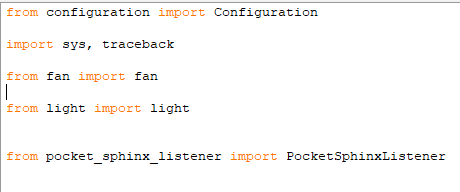
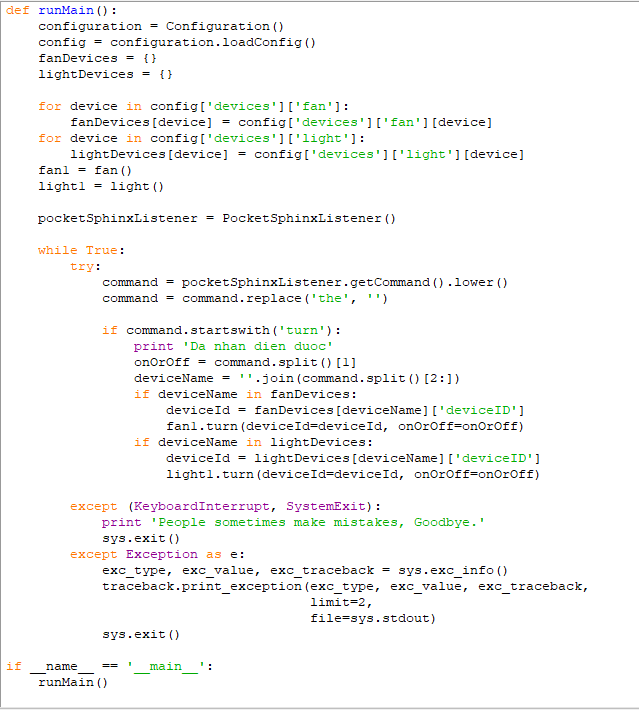
## Code python



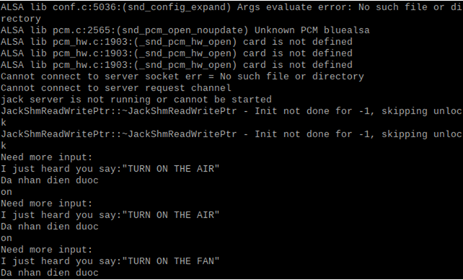
Hình ‑ Điều khiển quạt



Hình ‑ Điều khiển đèn

******

Hình 3‑6 Chương trình chính



Hình ‑ Sau khi chạy chương trình