

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI



BÁO CÁO TỔNG KẾT

ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN

NĂM HỌC 2023-2024

XÂY DỰNG PHẦN MỀM (PYTHON) VÀ TRIỂN KHAI TRÊN NỀN TẢNG NHÚNG (RASPBERRY PI) NHẪM GIẢI QUYẾT BÀI TOÁN TƯƠNG TÁC NGƯỜI DÙNG

Sinh viên thực hiện:

Nguyễn Ngọc Giáp Lớp: KTDTVT1-K62 Khoa: Điện- Điện tử

Đặng Văn Cậy Lớp: KTDTVT1-K62 Khoa: Điện- Điện tử

Đỗ Đức Duy Lớp: KTDTVT1-K62 Khoa: Điện- Điện tử

Vũ Anh Tuấn Lớp: KTDTVT1-K62 Khoa: Điện- Điện tử

Nguyễn Duy Hoàn Lớp: KTDTVT1-K62 Khoa: Điện- Điện tử

Người hướng dẫn: ThS. Võ Quang Sơn

Hà Nội 2024

LỜI NÓI ĐẦU

Sự quan tâm đến Trí tuệ Nhân tạo (AI) đã tăng lên đáng kể trong những năm gần đây, và sự phát triển đáng kể của lĩnh vực này bắt đầu từ cuối những năm 2000, đặc biệt là sau khi các công ty công nghệ lớn bắt đầu đầu tư mạnh mẽ vào nghiên cứu và phát triển trong lĩnh vực này. Tuy nhiên, sự chú ý rộng rãi đến AI đã trở nên đặc biệt nổi bật từ khoảng giữa thập kỷ 2010 và tiếp tục tăng cao vào các năm sau đó, khi các ứng dụng của nó trở nên phổ biến trong nhiều lĩnh vực như dịch vụ trực tuyến, y tế, tài chính, ô tô tự lái và nhiều lĩnh vực khác. Và nhóm chúng em quyết định chọn chatbot làm đối tượng nghiên cứu và ứng dụng cụ thể điều khiển nhà thông minh.

Nhà thông minh được xây dựng sử dụng Raspberry Pi 4 cho việc điều khiển và giám sát những thiết bị trong nhà thông qua việc giao tiếp với Chatbot. Nhóm chúng em quyết định làm mô hình ngôi nhà và các thiết bị điện bên trong. Một số thiết bị được điều khiển qua việc trò chuyện với chatbot:

- Điều khiển bật, tắt các thiết bị điện trong nhà.
- Điều khiển tuyến tính các thiết bị.
- Giám sát nhiệt độ, độ ẩm trong nhà.
- Bật tắt từng đèn hoặc tắt cả đèn.
- Đóng mở cửa bằng động cơ servo.

Xin chân thành cảm ơn!

MỤC LỤC

Lời nói đầu.....	2
CHƯƠNG 1: TỔNG QUAN.....	6
1.1 Đặt vấn đề.....	6
1.2 Mục tiêu.....	6
1.3 Nội dung nghiên cứu.....	7
1.4 Giới hạn.....	7
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT.....	8
2.1 Tổng quan về Chatbot.....	8
2.1.1 Khái niệm về Chatbot.....	8
2.1.2 Phân loại chatbot.....	8
2.1.3 Xử lý ngôn ngữ tự nhiên (Natural Language Processing).....	9
2.2 TỔNG QUAN VỀ API.....	9
2.2.1 Khái niệm về API.....	9
2.2.2. Mô hình của một API.....	9
2.2.3. Các tác vụ thực hiện qua API.....	10
2.3 WEBHOOK.....	11
2.3.1 Khái niệm Webhook.....	11
2.3.2 Hoạt động của Webhook.....	11
2.4 TỔNG QUAN VỀ DIALOGFLOW.....	12
2.4.1 Giới thiệu về Dialogflow.....	12
2.4.2 Cấu trúc của Dialogflow.....	13
2.5 RASBERRY PI 4.....	20
2.5.1 Giới thiệu về Board Raspberry Pi.....	20
2.5.2 Phần cứng của Raspberry Pi 4.....	21
2.5.3 Hệ điều hành Raspbian cho kit Raspberry Pi.....	25

2.6 ĐỘNG CƠ SERVO.....	26
2.6.1 Giới thiệu.....	26
2.6.2 Nguyên lý hoạt động.....	27
2.7 CẢM BIẾN NHIỆT ĐỘ, ĐỘ ẨM DHT11.....	28
2.8 MODULE RELAY 2 CHANNEL 5V	29
CHƯƠNG 3: THIẾT KẾ VÀ XÂY DỰNG MÔ HÌNH HỆ THỐNG.....	31
3.1 GIỚI THIỆU	31
3.2 TÍNH TOÁN VÀ THIẾT KẾ HỆ THỐNG.....	31
3.2.1 Thiết kế sơ đồ khối hệ thống.....	31
3.2.2 Tính toán và thiết kế mạch.....	32
3.3 XÂY DỰNG MÔ HÌNH PHẦN MỀM.....	34
3.3.1 LẬP TRÌNH TRÊN RASPBERRY PI 4.....	34
3.3.2. Giới thiệu phần mềm lập trình Visual Studio Code.....	36
3.3.3 XÂY DỰNG DIALOGFLOW.....	38
3.3.4 Lưu đồ giải thuật.....	48
3.3.5 Viết chương trình hệ thống.....	49
3.3.6 Public server lên internet bằng ngrok.....	50
3.4 XÂY DỰNG MÔ HÌNH PHẦN CỨNG.....	51
3.4.1 Đóng gói bộ điều khiển.....	51
3.4.2 Thi công mô hình.....	52
3.5 KẾT QUẢ VẬN HÀNH HỆ THỐNG.....	53
3.6 HƯỚNG DẪN SỬ DỤNG ,THAO TÁC	55
3.6.1 Hướng dẫn sử dụng.....	55
3.6.2 Quy trình thao tác.....	55
3.7 KẾT QUẢ NHẬN XÉT ĐÁNH GIÁ	56
3.7.1 DIALOGFLOW.....	56

3.7.2 RASPBERRY PI 4.....	57
3.7.3 PHẦN CỨNG.....	57
3.7.4 NHẬN XÉT VÀ ĐÁNH GIÁ.....	58
3.8. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	58
3.8.1 KẾT LUẬN.....	58
3.8.2 HƯỚNG PHÁT TRIỂN.....	59
Tài liệu tham khảo.....	60
Phụ lục.....	61

CHƯƠNG I: TỔNG QUAN

1.1 ĐẶT VẤN ĐỀ

Sự quan tâm đến Trí tuệ Nhân tạo (AI) đã tăng lên đáng kể trong những năm gần đây, và sự phát triển đáng kể của lĩnh vực này bắt đầu từ cuối những năm 2000, đặc biệt là sau khi các công ty công nghệ lớn bắt đầu đầu tư mạnh mẽ vào nghiên cứu và phát triển trong lĩnh vực này. Tuy nhiên, sự chú ý rộng rãi đến AI đã trở nên đặc biệt nổi bật từ khoảng giữa thập kỷ 2010 và tiếp tục tăng cao vào các năm sau đó, khi các ứng dụng của nó trở nên phổ biến trong nhiều lĩnh vực như dịch vụ trực tuyến, y tế, tài chính, ô tô tự lái và nhiều lĩnh vực khác. Và nhóm chúng em quyết định chọn chatbot làm đối tượng nghiên cứu và ứng dụng cụ thể điều khiển nhà thông minh.

Chatbot là một lĩnh vực của trí tuệ nhân tạo, chatbot thực hiện hội thoại thông minh giữa máy tính với người dùng thông qua trò chuyện trực tiếp. Với việc cung cấp thông tin cho từng ngữ cảnh để điều khiển các thiết bị khác nhau qua internet. Đồng thời ngày nay mạng lưới Internet đang ngày càng trở nên phổ biến, điều đó khiến cho việc điều khiển thiết bị từ xa qua mạng Internet trở nên dễ dàng và tiện lợi hơn.

Nhà thông minh được thiết kế xây dựng sử dụng Raspberry Pi 4 cho việc điều khiển và giám sát những thiết bị trong nhà thông qua việc giao tiếp với Chatbot. Ngoài việc điều khiển thiết bị về app android như những đề tài về Smarthome đã có, người dùng có thể giao tiếp với Chatbot về mọi thứ và điều khiển mọi thiết bị trong nhà. So với việc điều khiển bằng thao tác cứng nhắc trên các app android thông thường như các đề tài về IoT đã có trước đây, người dùng có thể trò chuyện với chatbot như một trợ lý ảo giúp người dùng quản lý và điều khiển hoạt động của ngôi nhà. Vì những lý do đó, nhóm chúng em quyết định thực hiện đề tài “*Xây dựng phần mềm (Python) và triển khai trên nền tảng nhúng (Raspberry Pi) nhằm giải quyết bài toán tương tác Người dùng*”.

1.2 MỤC TIÊU

Tạo ra được chatbot giao tiếp với người dùng và thực hiện nhiệm vụ khi có yêu cầu từ người dùng thông qua mạng internet. Đề tài được nghiên cứu, khảo sát, xây dựng với mục đích áp dụng những kiến thức đã học và tìm hiểu được để thiết kế, tạo ra một hệ thống Chatbot để điều khiển thiết bị một cách hoàn hảo.

Một số chức năng của hệ thống điều khiển bằng chatbot:

- Điều khiển ON/OFF các thiết bị dân dụng.

- Điều khiển tuyến tính các thiết bị.
- Giám sát nhiệt độ và độ ẩm trong gia đình hoặc bất cứ nơi nào có trang bị hệ thống.
- Có thể ON/OFF tất cả các thiết bị cùng 1 lúc hoặc từng thiết bị.

1.3 NỘI DUNG NGHIÊN CỨU

NỘI DUNG 1: Nhóm nghiên cứu tìm hiểu về Chatbot và cách tạo ra Chatbot.

NỘI DUNG 2: Nhóm tìm hiểu về các khái niệm về API, webhook và dialogflow.com (tiền thân là API.AI).

NỘI DUNG 3: Nhóm tìm hiểu các khái niệm Intent, Entity, Fulfillment,...và thiết kế ra một Chatbot từ Dialogflow .

NỘI DUNG 4: Nhóm tìm hiểu định dạng JSON và cách trích xuất dữ liệu để nhận biết yêu cầu từ người dùng.

NỘI DUNG 5: Nhóm tìm hiểu về kit Raspberry pi 4 và ngôn ngữ lập trình python.

NỘI DUNG 6: Lập trình giao tiếp giữa Raspberry pi 4 với chatbot bằng webhook thông qua mạng internet.

NỘI DUNG 7: Thiết kế, xây dựng và lập trình khối điều khiển công suất, cảm biến trong nhà.

NỘI DUNG 8: Thiết kế, xây dựng mô hình nhà thông minh.

NỘI DUNG 9: Chạy thử nghiệm và điều chỉnh hệ thống.

NỘI DUNG 10: Đánh giá kết quả thực hiện.

1.4 GIỚI HẠN

Tạo ra Chatbot cơ bản có thể trò chuyện với người dùng và thực hiện điều khiển các thiết bị trong nhà khi có yêu cầu từ người dùng.

Thiết kế mô hình ngôi nhà thông minh:

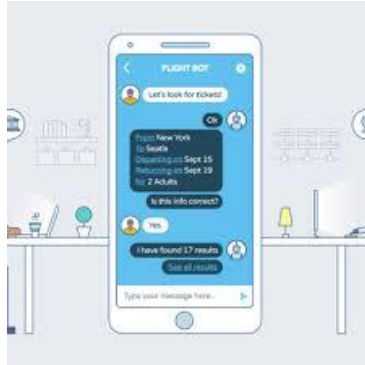
- Giám sát ngôi nhà bằng cảm biến và giao tiếp với người dùng thông qua Chatbot.
- Dùng Chatbot để điều khiển đèn, quạt, đóng mở cửa.... để mô phỏng.
- Hệ thống hoạt động thông qua mạng internet, Chatbot giao tiếp đơn giản.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1 TỔNG QUAN VỀ CHATBOT

2.1.1 Khái niệm về Chatbot

Chatbot là một chương trình máy tính tương tác với người dùng bằng ngôn ngữ tự nhiên với một giao diện đơn giản, âm thanh hoặc dưới dạng tin nhắn.



Hình 2.1: Khái niệm Chatbot

Về cơ bản, thì Chatbot là một hình thức thô sơ của phần mềm trí tuệ nhân tạo. Nó hoạt động độc lập, nó có thể tự động trả lời những câu hỏi hoặc xử lý tình huống càng thật càng tốt. Phạm vi và sự phức tạp của Chatbot được xác định bởi thuật toán của người tạo nên chúng.

Chatbot là sự kết hợp của các kịch bản có sẵn và tự học trong quá trình tương tác. Với các câu hỏi được đặt ra, chatbot sẽ dự đoán và phản hồi chính xác nhất có thể. Nếu tình huống đó chưa xảy ra (không có trong dữ liệu), Chatbot sẽ bỏ qua nhưng sẽ đồng thời “Bắt chước “ để áp dụng cho các cuộc trò chuyện thường xuyên (lặp đi lặp lại nhiều lần) về sau.

2.1.2 Phân loại chatbot

Chatbot thường được chia thành 2 loại :

- Audiotory (Âm thanh):

- + Siri (Apple).
- + Google Assistant (Google).
- + Cortana (Microsoft).
- + Jarvis của Tony - Stark.

- Textual (tin nhắn):

- + Chat GPT (do OpenAI phát triển dựa trên mô hình Transformer của Google).

- + Thời trang - tư vấn quần áo (H&M).
- + Thực phẩm - order pizza (Dominos Pizza).
- + Làm đẹp - stylish cá nhân (Sephora).
- + Giao thông - thông tin tàu điện vùng Kanto (qmau.me).

2.1.3 Xử lý ngôn ngữ tự nhiên (Natural Language Processing)

Để Chatbot có thể hiểu được người dùng nói gì thì ta phải dùng đến NLP (Natural Language Processing). Xử lý ngôn ngữ tự nhiên (NLP) là một nhánh của Trí tuệ nhân tạo, tập trung vào việc nghiên cứu sự tương tác giữa máy tính và ngôn ngữ tự nhiên của con người. Mục đích của lĩnh vực này hướng tới là giúp máy tính hiểu và thực hiện những nhiệm vụ liên quan đến ngôn ngữ của con người như: tương tác giữa người và máy, cải thiện hiệu quả giao tiếp giữa con người với con người, hoặc đơn giản là nâng cao hiệu quả xử lý văn bản và giọng nói. NLP được Dialogflow sử dụng để tạo ra những Chatbot thông minh.

2.2 TỔNG QUAN VỀ API

2.2.1 Khái niệm về API



Hình 2.2: Khái niệm về API

API (Application Programming Interface) là một giao diện lập trình ứng dụng mà một hệ thống máy tính hay ứng dụng cung cấp để cho phép các yêu cầu.

Dịch vụ có thể được tạo ra từ các chương trình máy tính khác, hoặc cho phép dữ liệu có thể được trao đổi qua lại giữa chúng dựa trên các thỏa thuận.

2.2.2. Mô hình của một API

API bao gồm 3 phần:

1. User: người đưa ra yêu cầu.
2. Client: máy tính gửi yêu cầu tới server.

3. Server: máy tính phản hồi yêu cầu .

Máy chủ được xây dựng với chức năng thu thập và lưu trữ dữ liệu. Khi máy chủ đang hoạt động, các lập trình viên tạo ra dữ liệu, bao gồm các Endpoint (điểm cuối) nơi có thể tìm thấy dữ liệu cụ thể. Sau đó, người dùng bên ngoài có thể truy vấn (hoặc tìm kiếm) dữ liệu trên máy chủ hoặc xây dựng một chương trình chạy tìm kiếm trên cơ sở dữ liệu và biến thông tin đó thành định dạng có thể sử dụng khác.

Nhiều công ty sử dụng các API mở từ các công ty lớn hơn như Google và Facebook để truy cập dữ liệu không có sẵn. Trong trường hợp này các API làm giảm đáng kể các rào cản phát triển đối với các công ty nhỏ hơn.

2.2.3. Các tác vụ thực hiện qua API

API là cách để hai máy tính (phần mềm) giao tiếp với nhau. Có bốn loại hành động mà API có thể thực hiện.

- GET: yêu cầu dữ liệu từ server.
- POST: gửi các thay đổi từ máy khách đến máy chủ khi thêm thông tin vào máy chủ, như tạo một mục mới
- PUT: sửa đổi hoặc thêm vào thông tin hiện có.
- DELETE: xóa thông tin hiện có.

Khi ta kết hợp các End point với các hành động này, thì có thể tìm kiếm, cập nhật mọi thông tin có sẵn trên API. Chúng ta cần phải kiểm tra tài liệu API để tìm hiểu cách mã hóa các hành động này, bởi vì chúng hoàn toàn khác nhau.

Thực hiện một yêu cầu trên một máy chủ:

- **HTTP**: giao thức truyền siêu văn bản. Đây là cách bạn truy cập web bằng cách nhập URL vào thanh tìm kiếm trong trình duyệt.
- **Định dạng văn bản**: XML, JSON. Đây là ngôn ngữ để truy cập dữ liệu qua API. Khi bạn nhận được dữ liệu của mình, bạn sẽ cần phải biết đọc mã XML hoặc JSON để hiểu những gì máy chủ đã cung cấp cho bạn.

Ý tưởng và các bước trong quy trình API:

- Hầu hết các API đều yêu cầu Key API. Khi bạn tìm thấy API bạn muốn sử dụng, hãy xem tài liệu về các yêu cầu truy cập. Hầu hết các API sẽ yêu cầu bạn hoàn tất xác minh danh tính, chẳng hạn như đăng nhập bằng tài khoản Google của bạn.

Bạn sẽ nhận được một chuỗi ký tự và số duy nhất để sử dụng khi truy cập API, thay vì chỉ thêm email và password của bạn.

- Các công cụ sẵn có (và thường miễn phí) giúp bạn điều hướng các yêu cầu của mình để truy cập các API hiện có bằng Key API mà bạn nhận được. Bạn vẫn sẽ cần phải biết một số cú pháp từ tài liệu, nhưng có rất ít kiến thức mã hóa cần thiết.
- Cách tốt nhất tiếp theo để lấy dữ liệu từ API là tạo URL từ tài liệu API.
- Yêu cầu API trông không khác nhiều so với URL trình duyệt thông thường, nhưng dữ liệu được trả về sẽ ở dạng dễ đọc cho máy tính.

2.3 WEBHOOK

2.3.1 Khái niệm Webhook

- Webhook là một HTTP callback: là 1 công cụ để truy vấn và lưu trữ dữ liệu của một Event xác định. Khi một trong những sự kiện được kích hoạt, nó sẽ gửi một HTTP POST đến URL đã được cấu hình webhook.

2.3.2 Hoạt động của Webhook

- Một ứng dụng web đang sử dụng Webhook sẽ POST một thông báo tới một URL khi một sự kiện xảy ra. Các sự kiện này do người dùng cấu hình và URL này được người dùng đăng ký với trang web. Hay nói cách khác, bằng cách cho phép người dùng chỉ định một URL cho các sự kiện khác nhau, ứng dụng sẽ POST dữ liệu tới các URL đó khi các sự kiện xảy ra. Sau khi nhận được thông báo thành công, webhook sẽ trả về status code 200 OK về cho trang web.

Những đặc điểm của webhook:

- **Push:** nhận dữ liệu theo thời gian thực

Bất cứ khi nào một event đã được cấu hình trước xảy ra thì ngay lập tức dữ liệu sẽ gửi đến URL đã đăng ký ngay lập tức với thời gian thực.

- **Pipes:** nhận dữ liệu và truyền đi

Một Pipe xảy ra khi webhook không chỉ nhận dữ liệu thời gian thực, mà còn tiếp tục thực hiện khi có một sự kiện mới, kích hoạt các hành động không liên quan đến sự kiện lúc đầu.

- **Plugins:** xử lý dữ liệu và trả lại thứ gì đó

Đây là nơi toàn bộ web sẽ trở thành một nền tảng để lập trình. Bạn có thể sử dụng webhook này để cho phép người khác mở rộng ứng dụng của bạn. Ý tưởng chung là một ứng dụng web gửi dữ liệu qua webhook cũng sẽ sử dụng phản hồi từ webhooks để sửa đổi dữ liệu của riêng nó. Theo cách này, webhook thay đổi ứng dụng của bạn nếu bạn muốn cho phép những người khác thực sự mở rộng và nâng cao khả năng của ứng dụng của bạn.

2.4 TỔNG QUAN VỀ DIALOGFLOW

2.4.1 Giới thiệu về Dialogflow



Hình 2.4.1: Hình ảnh logo Dialogflow

Dialogflow.com (tiền thân là API AI) là một API do Google cung cấp nhằm giúp các lập trình viên có thể thao tác dễ dàng hơn khi lập trình các sản phẩm có giao tiếp giữa user (người dùng) với sản phẩm thông qua các đoạn hội thoại bằng văn bản (text) hoặc giọng nói (voice).

Dialogflow là nền tảng về công nghệ tương tác giữa con người và máy tính dựa trên các cuộc trò chuyện bằng ngôn ngữ tự nhiên và có thể được sử dụng để phát hiện các từ khóa và ý định trong câu của người dùng. Vai trò của nó là giúp xây dựng chatbot sử dụng Machine Learning.

Với DialogFlow hỗ trợ các tích hợp với các nền tảng, dịch vụ khác nhau như Facebook Messenger, các nền tảng Google, Skype, v.v... Chatbot của bạn sẽ làm việc với tất cả các nền tảng này nếu như bạn thiết lập tích hợp đúng cách.

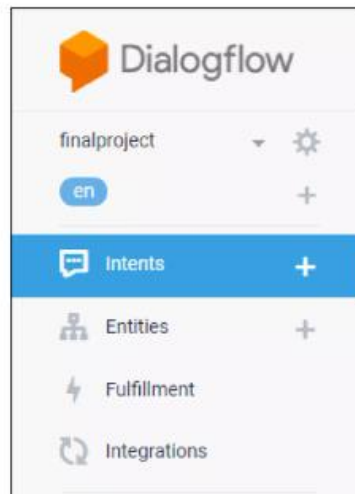
Dialogflow sử dụng trí tuệ nhân tạo (AI) giúp phân tích ngôn ngữ tự nhiên để hiểu được những gì người dùng đưa vào.

Hiện Dialogflow có 2 phiên bản:

- Standard: Hoàn toàn miễn phí để sử dụng
- Enterprise: Cần trả một ít tiền

2.4.2 Cấu trúc của Dialogflow

Dialogflow bao gồm 5 phần chính là Agent, Intent, Entities, Fulfillment và Integrations.



Hình 2.4.2.1: Các thành phần của Dialogflow

a. Agent

Agent là khái niệm được dùng để đại diện cho một module NLU (Natural Language Understanding).

Agent giúp bạn có thể phân tích những gì mà User (người dùng) đưa vào (text hoặc voice) để chuyển thành những dữ liệu mà bạn có thể xử lý được bằng lập trình.

Bạn sẽ sử dụng một Agent để quản lý các đoạn hội thoại thông qua Intents, Entities.

b. Intent

Intent là đại diện cho một ánh xạ giữa những gì User đưa vào và hành động sẽ được thực hiện bởi phần mềm. Một Intent là một tập những gì User (người dùng) nói mà chúng đồng nghĩa với nhau.

Hình 2.4.2.2: Các thành phần trong Intent

Giao diện intent có các thành phần như sau:

- Contexts
- Events
- Training phrases
- Responses
- Action and parameters
- Fullfillment

Contexts:

Contexts thể hiện ngữ cảnh hiện tại của yêu cầu người dùng. Điều này giúp cho Chatbot hiểu những gì mà User (người dùng) đang nói gì bằng việc phân biệt các cụm từ mơ hồ hoặc có ý nghĩa khác nhau. Trong một Intent của Dialogflow có input context và output context như hình dưới đây:

Hình 2.4.2.3: Hình ảnh mục context

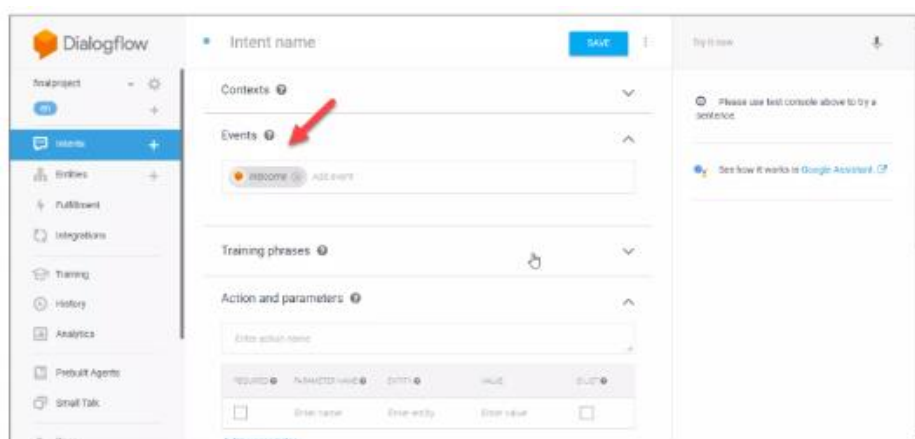
Input context có chức năng giúp Dialogflow xác định Intent này chỉ được thực hiện khi đang có các context tương ứng.

Output context được tạo ra khi một Intent được gọi và dùng để xác định đâu là intent tiếp theo.

Mặc định một context của Dialogflow sẽ hết hạn sau 5 lần requests hoặc sau mỗi 10 phút kể từ khi nó được tạo ra.

Event:

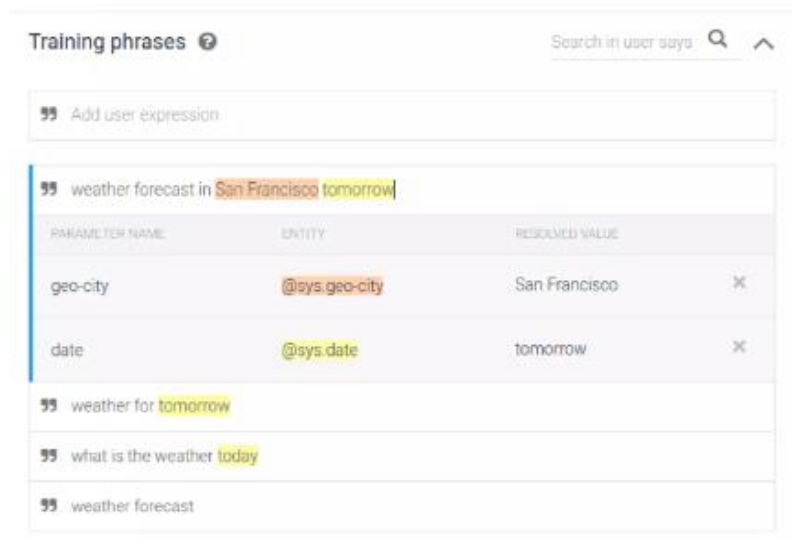
Events là một tính năng cho phép bạn gọi các Intents theo tên sự kiện thay vì truy vấn từ User (người dùng). Dưới đây là các tên event được Dialogflow hỗ trợ khi người dùng click vào các nền tảng nhắn tin.



Hình 2.4.2.4: Hình ảnh mục Events

Training Phase:

- Là những gì mà người dùng có thể nói để thực hiện một yêu cầu nhất định.
- Có 2 loại user say trong Dialogflow:
 - Là những mẫu(Template) văn bản mà nó sẽ tham chiếu đến các Entities.
 - Là những ví dụ(Example) mà User có thể nói.

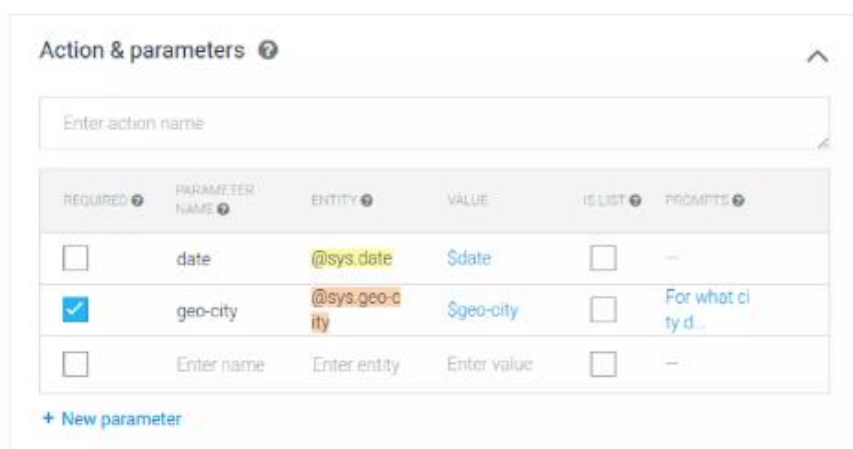


Hình 2.4.2.5: Hình ảnh mục Training Phrases

Mặc dù Dialogflow hỗ trợ cả hai loại hình trên, nhưng chúng ta chỉ nên sử dụng loại số 2 (example) vì nó giúp cho Dialogflow học cũng như xử lý nhanh và tối ưu nhanh hơn. Ngoài ra, khi chúng ta tạo intents, nếu nhập được càng nhiều các ví dụ ở user say thì giúp cho Chatbot càng “thông minh” hơn.

Action and parameters:

Action (hành động) chính là giá trị mà Dialogflow hỗ trợ để giúp lập trình viên phân biệt được trong trường hợp này đâu là các văn bản liên quan đến Intent đề cập và đâu là các văn bản khác. Đi kèm với Action là các Parameter (tham số).



Hình 2.4.2.6: Hình ảnh mục Action & Parameters

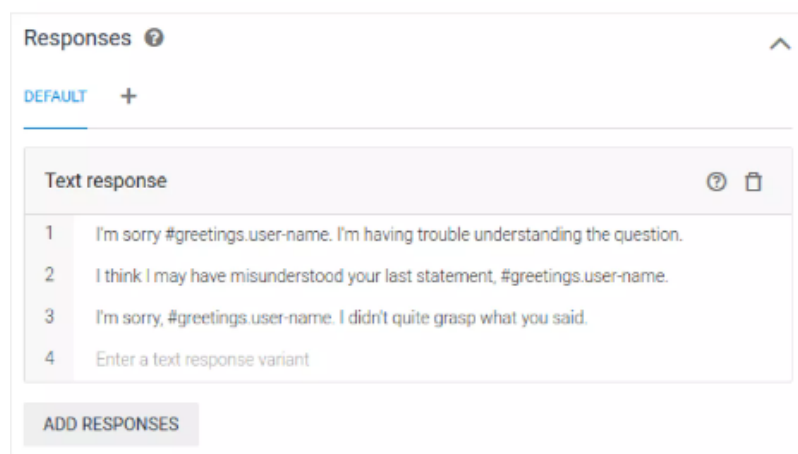
Parameters là các tham số thường được trích xuất trong câu nói của người dùng, được liên kết với các entities. Mỗi tham số là 1 giá trị được trích xuất từ văn bản của người dùng qua việc sử dụng các entities tương ứng, ví dụ: tham số address, kiểu @sys.location, khi đó, trong văn bản của người dùng có Hà Nội, hay Đà Nẵng, Tokyo, New York, Paris,... thì sẽ được Dialogflow xử lý để gán vào cho tham số address.

Khi định nghĩa tham số, chúng ta cần quan tâm đến

- Constant: Là giá trị mà chúng ta nhập vào ô value, tham số sẽ luôn luôn mang giá trị này.
- Default: đây giá trị mặc định được gán cho tham số nếu như trong văn bản của user không nhắc tới.
- IsList: giúp Dialogflow xác định cả 3 giá trị sẽ được gán vào cho cũng 1 tham số.
- Required: Khi required được lựa chọn, nếu trong văn bản của người dùng mà Dialogflow không tìm được giá trị để gán vào cho tham số, Dialogflow sẽ đánh dấu để yêu cầu người dùng đưa vào. Khi đó câu hỏi từ Chatbot để yêu cầu người dùng cung cấp thông tin sẽ được viết tại mục Prompt.

Response:

Response là những câu trả lời tương ứng với những câu hỏi người dùng đặt ra ở trên, chúng ta có thể sử dụng giá trị của parameters để trả lời cho người dùng. Để sử dụng chúng ta sử dụng theo định vị @parameter_name.



Hình 2.4.2.7: Hình ảnh mục Responses

Fulfillment:

Cài đặt để cho phép bạn gửi thông tin từ intent đó khi nó được người dùng yêu cầu đến web service và nhận thông tin xử lý từ nó. Thông tin được gửi dưới định dạng Json có cấu trúc như sau:



Hình 2.4.2.8: Hình ảnh của mục Fulfillment

Trong đó, có 4 mục chính là:

- responseId: Id duy nhất cho mỗi yêu cầu.
- session: Id làm việc duy nhất.
- queryResult: kết quả yêu cầu đoạn thoại.
- original DetectIntentRequest: Yêu cầu đến từ một nền tảng tích hợp.

c. Entities

Khái niệm: Entities là những công cụ được dùng để trích xuất các giá trị của tham số từ ngôn ngữ tự nhiên. Bất kỳ những gì mà bạn muốn biết từ nội dung của người dùng đều sẽ có một Entity tương ứng.



Hình 2.4.2.9: Hình ảnh của mục Entities

- Phân loại:

Dialogflow định nghĩa ra ba loại Entities:

1. System (là các entities được tạo bởi Dialogflow).
2. Developer (là các entities được tạo bởi lập trình viên).
3. User (là entities được tạo ra cho mỗi lần hỏi từ phía người dùng).

System Entities:

Đây là những entities được tạo sẵn bởi Dialogflow (giống như trong ngôn ngữ lập trình thì String, Number, Char, Double,.... là những kiểu dữ liệu được tạo sẵn bởi ngôn ngữ lập trình) để tạo điều kiện cho lập trình viên dễ dàng hơn trong việc xử lý những nghiệp vụ cơ bản của mọi loại Chatbot. Tùy thuộc vào mỗi loại ngôn ngữ như: Tiếng Anh, tiếng Pháp, hay tiếng Nhật,... mà số lượng system entities của mỗi ngôn ngữ sẽ khác nhau.

Developer Entities:

Đây là những entities do lập trình viên tạo ra.

Cấu tạo của entities:

- Cấu trúc entity gồm 2 phần:
 1. Reference value (Là giá trị mà developer cần nhận được).
 2. Synonyms (Là những từ đồng nghĩa).

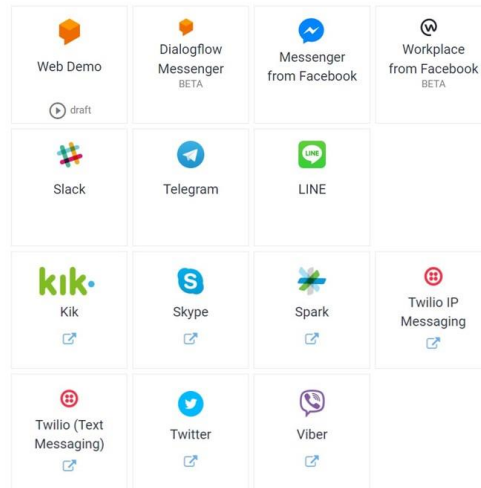
Ý nghĩa của nó là khi người dùng nói có đề cập đến bất kỳ văn bản nào giống như những văn bản trong phân synonyms thì nó đều được ánh xạ về giá trị reference.

d. Fulfillment

Fulfillment (hay Webhook) là một API ở Backend mà chúng ta cần cung cấp để Dialogflow gọi đến khi cần xử lý nghiệp vụ trước khi Dialogflow trả lời lại cho người dùng. Service của bạn sẽ nhận POST request từ Dialogflow bất kỳ khi nào người dùng nhắn tin đến Dialogflow.

e. Integrations

Dưới đây là các mục được Dialogflow tích hợp vào các nền tảng nhắn tin khác. Có rất nhiều nền tảng được Dialogflow hỗ trợ.



Hình 2.4.2.10: Các nền tảng tích hợp trong Intergrations

2.5 RASBERRY PI 4

2.5.1 Giới thiệu về Board Raspberry Pi

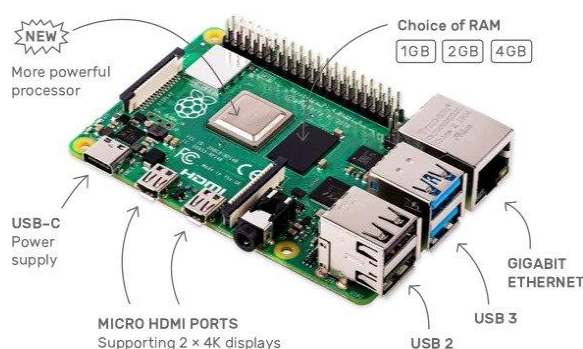


Hình 2.5.1: Board Raspberry Pi

Raspberry Pi là một chiếc máy tính tí hon giá chỉ từ 35\$ chạy hệ điều hành Linux ra mắt vào tháng 2 năm 2012. Ban đầu Raspberry Pi được phát triển dựa trên ý tưởng tiến sĩ Eben Upton tại đại học Cambridge muốn tạo ra một chiếc máy tính giá rẻ để học sinh có thể dễ dàng tiếp cận và khám phá thế giới tin học. Raspberry Pi (RPi) là một máy tính siêu nhỏ, chỉ có kích thước như 1 chiếc thẻ ATM rút tiền. Bạn chỉ cần 1 bàn phím, 1 tivi hoặc 1 màn hình có cổng HDMI/DVI, 1 nguồn USB 5V và 1 dây micro USB là đã có thể sử dụng RPi như 1

máy tính bình thường. Với RPi, bạn có thể sử dụng các ứng dụng văn phòng, nghe nhạc, xem phim độ nét cao (tới 1024p).

2.5.2 Phần cứng của Raspberry Pi 4



Hình 2.5.2.1: Phần cứng của Raspberry Pi 4

Thông số kĩ thuật	Nội dung
Nguồn cung cấp	Điện áp 5V thông qua cổng Micro USB hoặc GPIO.
Soc	Broadcom BCM2711.
CPU	ARM Cortex-A72 64-bit 4 nhân 1.5GHz.
GPU	Broadcom VideoCore VI, hỗ trợ OpenGL ES 3.x.
RAM	1GB/2GB/4GB LPDDR4-3200 SDRAM (tùy chọn).
Cổng USB	2 cổng USB 2.0, 2 cổng USB 3.0.
Ngõ vào video	2 cổng micro HDMI hỗ trợ độ phân giải lên đến 4K.
Ngõ ra video	Hỗ trợ độ phân giải lên đến 4K qua cổng micro HDMI.
Ngõ vào audio	Cổng âm thanh 3.5mm.

Khe cắm thẻ nhớ	Khe cắm thẻ nhớ microSD.
Kết nối mạng	Gigabit Ethernet, Wi-Fi 802.11ac, Bluetooth 5.0.
Ngõ ra khác	GPIO, I2C, SPI, UART.

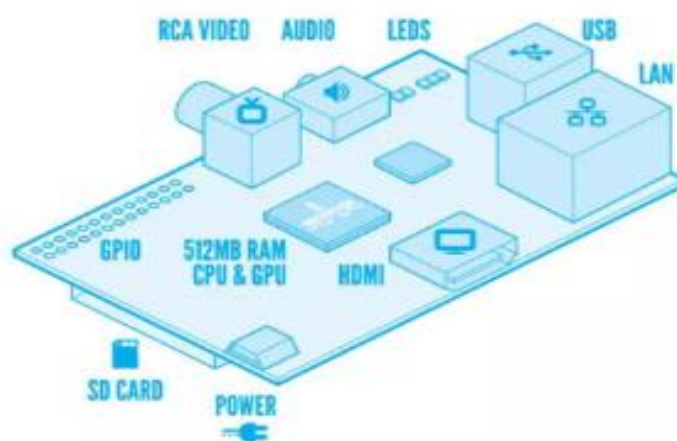
Bảng 2.5: Thông số kỹ thuật của Raspberry Pi 4

Raspberry Pi 4 sử dụng chip Broadcom BCM2711 là trung tâm của bo mạch. Đây là một chip SoC (System-on-Chip) tích hợp CPU ARM Cortex-A72 64-bit quad-core chạy ở tốc độ 1.5GHz. Chip BCM2711 không chỉ chứa CPU mà còn tích hợp GPU (VideoCore VI) và các thành phần khác như bộ điều khiển bộ nhớ, bộ điều khiển giao tiếp, v.v. Chip này đóng vai trò quan trọng trong việc điều khiển và hoạt động của Raspberry Pi 4.

a. Nguồn cung cấp

Kit Raspberry Pi 4 model B (gọi tắt là kit raspberry Pi 4) được cấp nguồn qua cổng microUSB như hình 2.1, với mức điện áp là 5V DC, dòng cấp khuyến nghị là 800 mA khi không gắn ngoại vi. Các trường hợp còn lại, ta nên sử dụng nguồn có dòng cấp từ 1000 mA đến 2000 mA.

b. Các cổng giao tiếp



Hình 2.5.2.2: Các cổng giao tiếp trên Raspberry Pi 4

Trên kit sử dụng các cổng giao tiếp sau:

Dưới đây là thông tin chi tiết về các cổng giao tiếp của Raspberry Pi 4 Model B:

- HDMI (High-Definition Multimedia Interface):

Hỗ trợ các độ phân giải từ 640x350 đến 3840x2160 (4K) với cả hai chuẩn màu PAL và NTSC.

Dòng cung cấp cho cổng HDMI có thể thay đổi tùy thuộc vào độ phân giải và nội dung hiển thị.

- CSI (Camera Serial Interface):

Có cổng kết nối với camera, gồm 15 chân, cho phép kết nối với các module camera thông thường hoặc camera NoIR.

- Audio:

Hỗ trợ xuất âm thanh qua cổng Audio chuẩn 3.5mm.

- Cổng Ethernet:

Sử dụng cổng Ethernet chuẩn Gigabit (10/100/1000 Mbit/s) cho kết nối mạng.

- Khe cắm thẻ nhớ microSD:

Dùng để lưu trữ dữ liệu và hệ điều hành của Raspberry Pi 4.

- Cổng USB:

Có 4 cổng USB chuẩn A.

Dòng cung cấp tối đa cho mỗi cổng USB là 1.2A (1200mA), có thể điều chỉnh từ 0.6A đến 1.2A thông qua tập tin **/boot/config.txt**.

Sử dụng IC AP2553W6 để quản lý năng lượng trên các cổng USB.Pi với màn hình cảm ứng để hiển thị và sử dụng Raspberry một cách trực quan nhất. Chúng ta có thể thực hiện các tác vụ tương đương như khi sử dụng chuột và bàn phím.

C, Các chân giao tiếp GPIO của kit



Hình 2.5.2.3: Sơ đồ chân GPIO

Trên Raspberry Pi 4 Model B, các chân GPIO được sắp xếp theo các hàng và cột trên bo mạch. Dưới đây là một số thông tin cụ thể về GPIO trên Raspberry Pi 4:

- Số lượng chân GPIO: Raspberry Pi 4 có tổng cộng 40 chân GPIO.

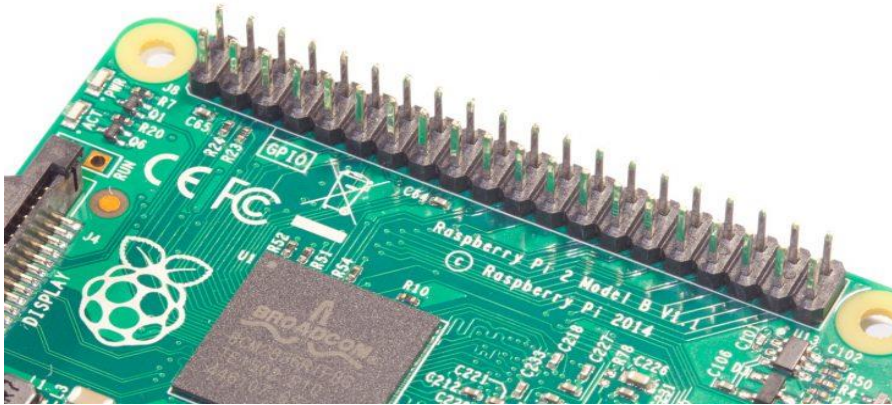
Chức năng của các chân GPIO: Các chân GPIO có thể được cấu hình để thực hiện nhiều chức năng khác nhau, bao gồm đầu vào kỹ thuật số, đầu ra kỹ thuật số, PWM (Pulse Width Modulation), SPI (Serial Peripheral Interface), I2C (Inter-Integrated Circuit), UART (Universal Asynchronous Receiver-Transmitter), và nhiều chức năng khác.

Điện áp và dòng cung cấp: Điện áp hoạt động của GPIO là 3.3V. Mỗi chân GPIO có thể cung cấp dòng tối đa là 16mA và tổng dòng tối đa cho tất cả các chân là 50mA.

- Bố trí GPIO:

Các chân GPIO được sắp xếp thành hai hàng dọc trên bo mạch. Hàng GPIO 1 (GPIO 2 đến GPIO 27) và Hàng GPIO 2 (GPIO 28 đến GPIO 45) được gán cho các chức năng GPIO thông thường, còn một số chân có chức năng đặc biệt như nút nhấn, LED chỉ báo, UART, SPI, I2C, và nhiều chức năng khác.

Các chân GPIO cung cấp khả năng linh hoạt cao cho Raspberry Pi 4, cho phép người dùng tương tác với nhiều loại thiết bị ngoại vi và mạch điều khiển để phát triển các ứng dụng và dự án điện tử đa dạng.



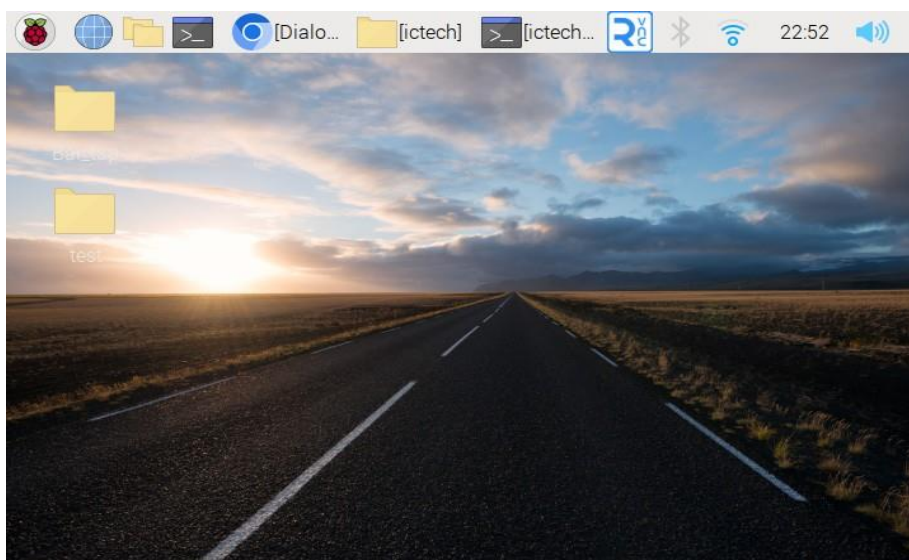
Hình 2.5.2.4: Hình ảnh các chân GPIO thực tế

2.5.3 Hệ điều hành Raspbian cho kit Raspberry Pi

Các thế hệ Raspberry Pi đều sử dụng hệ điều hành dựa trên nền tảng Linux. Phần cứng GPU được truy cập thông qua Image Firmware và được nạp vào GPU, vào lúc khởi động từ thẻ nhớ SD. Linux là tên gọi của một họ các hệ điều hành có mã nguồn mở. Lúc đầu, nó được phát triển để chạy trên các máy tính PC dựa trên kiến trúc Intel X86 nhưng hiện nay, Linux đã được port qua nhiều kiến trúc khác. Cái tên Linux được gọi do nó sử dụng nhân hệ điều hành là Linux kernel. Linux kernel được giới thiệu lần đầu vào tháng 10 năm 1991, được viết và duy trì bởi Linus Torvalds. Từ Linux, người ta đã phát triển nhiều hệ điều hành khác nhau như Ubuntu, Debian, CentOS, Red Hat,... gọi chung là các bản phân phối.

Raspbian là một hệ điều hành với bản phân phối mã nguồn mở, được xây dựng dựa trên Debian, nhưng đã được tối ưu để phù hợp với phần cứng của kit Raspberry Pi. Phiên bản đầu tiên ra đời vào tháng 6 năm 2012. Hiện nay, Raspbian vẫn đang là một trong những bản phân phối phổ biến, được cộng đồng đón nhận và hỗ trợ nhiều nhất dành cho Raspberry Pi, với hơn 35000 gói phần mềm có sẵn.

Raspbian sử dụng môi trường Desktop (Desk Environments). Nó được phát triển từ giao diện GNOME 2, cung cấp một giao diện người dùng (GUI) hoàn chỉnh bằng cách sử dụng chung một toolkit và thư viện để xây dựng các thành phần đồ họa.



Hình 2.5.3: Giao diện của hệ điều hành Raspbian

Trên kit Raspberry Pi, quá trình boot hệ điều hành Raspbian được trợ giúp bởi GPU nằm trên SoC. GPU chứa một lõi RISC nhỏ để có thể chạy các đoạn code ở trên ROM. Nhờ những đoạn code này, GPU có thể khởi tạo cho chính bản thân nó cũng như cho thẻ nhớ SD-nơi chứa hệ điều hành của kit.

2.6 ĐỘNG CƠ SERVO

2.6.1 Giới thiệu

Servo là một loại động cơ điện đặc biệt. Khác với các động cơ thông thường xoay liên tục khi được cung cấp điện, servo chỉ quay khi được điều khiển (thông qua tín hiệu PWM) trong một khoảng góc nhất định, thường từ 0° đến 180° , mặc dù một số servo có thể quay liên tục từ 0° đến 360° . Mỗi loại servo có kích thước, trọng lượng và cấu trúc khác nhau. Một số nhẹ chỉ khoảng 9 gram (thường được sử dụng trên máy bay mô hình), trong khi một số khác có mã lực cao (lên đến vài chục Newton-mét) hoặc có bộ giảm tốc và cấu trúc mạnh mẽ.



Hình 2.6: Hình ảnh động cơ Servo

2.6.2 Nguyên lý hoạt động

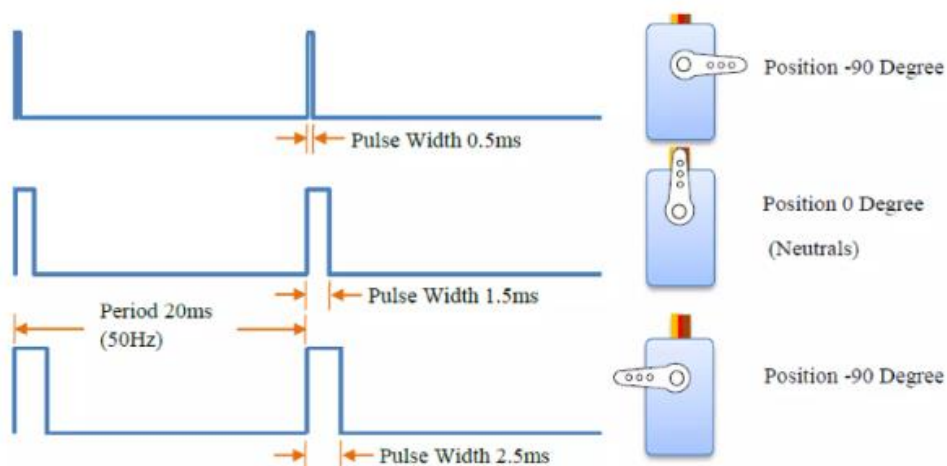
Động cơ servo được thiết kế những hệ thống hồi tiếp vòng kín. Tín hiệu đầu ra của động cơ được nối với một mạch điều khiển. Khi động cơ quay, vận tốc và vị trí sẽ được phản hồi trở lại mạch điều khiển. Trường hợp nếu có bất kỳ lý do nào ngăn cản chuyển động quay của động cơ, cơ cấu hồi tiếp sẽ nhận thấy tín hiệu ra chưa đạt được vị trí mong muốn. Mạch điều khiển tiếp tục chỉnh sai lệch cho động cơ đạt được điểm chính xác.

- Sơ đồ chân:

Chân	Màu (thường gặp)
VCC (+5v)	Đỏ
GND	Đen, Nâu
Tín hiệu điều khiển	Cam, Vàng

Bảng 2.6.2: Các chân của Servo

- Điều khiển động cơ servo bằng PWM



Hình 2.6.2.1: Điều khiển động cơ Servo bằng Duty Cycle

- Công thức tính:

$$\text{Duty Cycle} = \frac{\text{thời ở mức cao của xung}}{\text{chu kỳ xung}}$$

- Quy định góc quay:

Duty Cycle(Xung)	Góc quay
2.5%	0°
7.5%	90°
12.5%	180°

Bảng 2.6.2.2: Hình ảnh xung tương ứng với góc quay

Duty Cycle là phần trăm ở mức cao của xung trong 1 chu kỳ xung.

- Ví dụ: tần số $f = 50 \text{ Hz} \Rightarrow T = 1/f = 1/50 = 20 \text{ (ms)}$.

Duty Cycle = $\text{time}/20\text{ms} = 2.5\% \Rightarrow \text{time} = 0.5 \text{ ms}$ (góc 0°).

2.7 CẢM BIẾN NHIỆT ĐỘ, ĐỘ ẨM DHT11

DHT11 Là cảm biến nhiệt độ, độ ẩm được dùng rất phổ biến hiện nay, lấy dữ liệu thông qua giao tiếp 1-wire (giao tiếp digital 1-wire truyền dữ liệu duy nhất). Cảm biến được tích hợp bộ tiền xử lý tín hiệu giúp dữ liệu nhận về 1 cách chính xác mà không cần phải qua bất kỳ tính toán nào.

Đặc điểm:

- Điện áp hoạt động : 3V -> 5V (DC)
- Dải độ ẩm hoạt động: 20% -> 90% RH, sai số $\pm 5\%$ RH
- Dải nhiệt độ hoạt động : 0°C ~ 50°C, sai số $\pm 2^\circ \text{C}$
- Tần số lấy mẫu tối đa: 1 Hz
- Khoảng cách truyền tối đa: 20 m

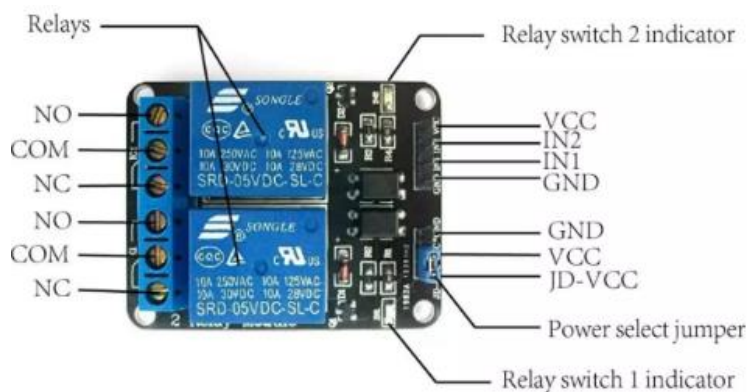
Sơ đồ chân Cảm biến DHT11 gồm 2 chân cấp nguồn, và 1 chân tín hiệu như hình bên dưới:



Hình 2.7: Hình ảnh cảm biến DHT11

2.8 MODULE RELAY 2 CHANNEL 5V

Module Relay 2-Channel 5V được dùng nhiều trong các ứng dụng đóng ngắt các thiết bị tiêu thụ dòng điện lớn ($<10A$). Module có thể đóng ngắt đồng thời hai kênh bằng tín hiệu điều khiển (với mức điện áp 3V3 hoặc 5V) từ các vi điều khiển khác nhau như: Arduino, 8051, AVR, PIC, DSP, ARM, ARM, MSP430, logic TTL,...Đặc biệt module này được cách ly bằng optocoupler giúp cung cấp bảo vệ tốt hơn cho các vi điều khiển, đồng thời tăng tính ổn định và an toàn cho hệ thống.



Hình 2.8: Module Relay 2 channel 5V

Module này được kết nối với các board điều khiển thông qua một cổng header 4 chân như sau :

1. VCC cung cấp nguồn cho các opto.
2. GND kết nối với GND của board điều khiển.

3. IN1 và IN2 dùng để điều khiển relay 1 và relay 2, tích cực mức thấp.

Ngoài ra còn một 3 chân header được dùng để cấp nguồn cho relay, header này sẽ có một jumper dùng để kết nối chân VCC với chân RY_VCC mục đích dùng chung nguồn VCC (5V) từ header 4 chân cho relay, thông thường jumper được nối lại với nhau. Nếu bạn muốn cách ly tín hiệu điều khiển với nguồn cấp cho relay, bạn có thể loại bỏ jumper này và cung cấp nguồn riêng 5V cho chân RY_VCC.

Thông số kỹ thuật:

- Đóng ngắt được dòng điện cao: AC250V 10A, DC30V 10A
- 2 led báo trạng thái relay
- Điện áp điều khiển: 5V
- Kích thước: 50x45 mm

CHƯƠNG 3: THIẾT KẾ VÀ XÂY DỰNG MÔ HÌNH HỆ THỐNG

3.1 GIỚI THIỆU

Nhà thông minh được thiết kế sử dụng Raspberry Pi 4 để điều khiển và giám sát các thiết bị trong nhà thông qua việc giao tiếp với Chatbot. Khi nhận được yêu cầu từ người dùng, Chatbot sẽ gửi thông tin đến Raspberry Pi để thực hiện điều khiển các thiết bị phần cứng tương ứng, Đồng thời, Raspberry Pi cũng sẽ xử lý và tạo ra câu trả lời phản hồi cho người dùng thông qua Chatbot. Điều này tạo ra một hệ thống thông minh và tự động cho việc điều khiển và giám sát các thiết bị trong nhà.

Các yêu cầu điều khiển từ người dùng bao gồm :

- Đóng mở các thiết bị như đèn, quạt, cửa
- Điều khiển ON/OFF bóng đèn.

Thực hiện đóng mở cửa bằng động cơ servo

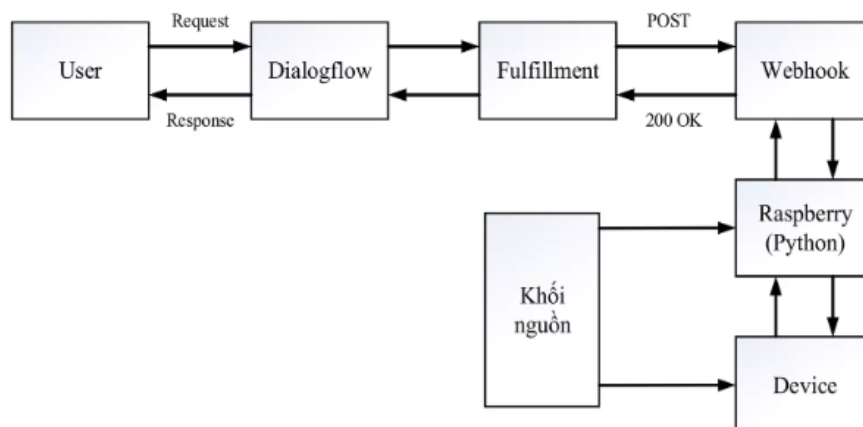
Gửi thông tin về nhiệt độ, độ ẩm trong căn phòng khi có yêu cầu của người dùng.

Phần 3.7 và 3.8 của chương này đề cập đến việc nhóm thực hiện xây dựng phần mềm và xây dựng mô hình phần cứng Smarthome.

3.2 TÍNH TOÁN VÀ THIẾT KẾ HỆ THỐNG

3.2.1 Thiết kế sơ đồ khối hệ thống

a. Sơ đồ khối hệ thống

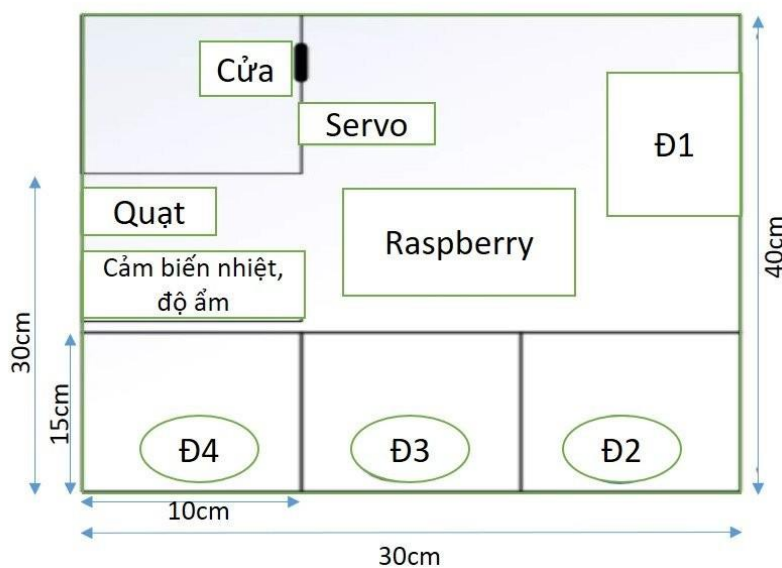


Hình 3.2.1.1: Sơ đồ khối hệ thống

Trong đó:

- **User** là người sử dụng trò chuyện trực tiếp với Chatbot.
- **Dialogflow** nhận yêu cầu từ người dùng và sau đó xử lý để trả lời cho yêu cầu đó.
- **Fulfillment** là nơi cho phép Dialogflow gửi thông tin đến webhook.
- **Webhook** là nơi tiếp nhận thông tin từ Dialogflow.
- **Raspberry Pi** nhận thông tin từ webhook và xử lý để điều khiển các thiết bị phần cứng, đồng thời trả về response cho Dialogflow.
- **Device** là các thiết bị công suất thực hiện yêu cầu điều khiển của người dùng.
- **Khởi nguồn** cung cấp nguồn cho Raspberry Pi và các thiết bị công suất hoạt động.

b, Sơ đồ mô hình hệ thống.



Hình 3.2.1.2: Sơ đồ mô hình thiết kế

3.2.2 Tính toán và thiết kế mạch

Phần này nhóm tiến hành thiết kế, tính toán ra sơ đồ nguyên lí cho từng khối.

a. Thiết kế khối Dialogflow

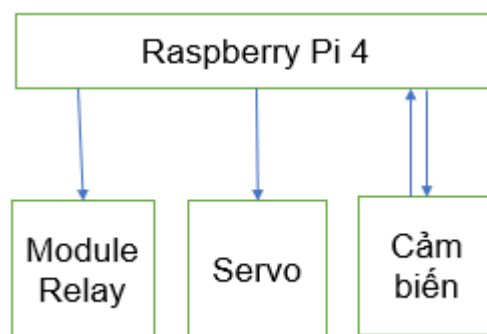
Đây là khối giúp Chatbot trò chuyện với người dùng và phân tích câu nói của họ bằng Natural Language Processing(NLP) ,sau đó chuyển đổi thành dữ liệu dưới định dạng file JSON. Dữ liệu này sẽ được gửi đến webhook để xử lí và Chatbot nhận câu trả lời tương ứng từ webhook để hiển thị cho người dùng.

b. Thiết kế khối điều khiển Raspberry

Đây là khối quan trọng nhất trong hệ thống, có chức năng tạo ra một webhook để nhận dữ liệu được gửi từ Dialogflow thông qua internet Sau đó, nó tiến hành phân tích và xử lý dữ liệu để điều khiển các thiết bị trong nhà theo yêu cầu của người dùng. Đồng thời, nó cũng xử lý để gửi lại câu trả lời cho Dialogflow để hiển thị cho người dùng.

c. Thiết kế khối điều khiển

Sơ đồ các khối trong mạch điều khiển :



Hình 3.2.2: Sơ đồ các khối điều khiển động lực

Module Relay:

Các thiết bị trong nhà sử dụng điện áp xoay chiều 220V, do đó để điều khiển đóng ngắt chúng, ta sử dụng Module Relay. Vì đầu ra điều khiển của Raspberry Pi là 3,3V, nên ta chọn mạch Relay có áp điều khiển là 5V để đảm bảo tương thích và an toàn trong quá trình điều khiển.

$$\text{Ta có cường độ dòng điện định mức qua đèn là : } I_{dm} = \frac{P}{U} = \frac{12}{220} = 0.05 \text{ A} \quad (3.1)$$

$$\text{Ta chọn } I_{cont} > 2.5 \times I_{dm} = 0.125 \text{ A. Ta chọn Module relay có } I = 10 \text{ A} \quad (3.2)$$

Khối cảm biến :

Để giám sát nhiệt độ, độ ẩm trong nhà và trả về kết quả khi có yêu cầu từ người dùng, ta có thể dùng một cảm biến DHT11, cảm biến này sử dụng nguồn được cấp trực tiếp từ Raspberry Pi.

Khởi Servo :

Để thực hiện yêu cầu đóng mở cửa, ta sử dụng động cơ servo SG90 để điều khiển góc quay phù hợp thực hiện yêu cầu đóng mở cửa. Động cơ servo SG90 được lựa chọn vì khả năng đáp ứng linh hoạt, độ chính xác cao và khả năng điều khiển góc quay từ 0 đến 180 độ, phù hợp cho việc điều khiển cửa.

d. Thiết kế khối nguồn

Sử dụng nguồn 220V AC cung cấp trực tiếp cho raspberry pi 4 và các thiết bị sử dụng điện xoay chiều.

3.3 XÂY DỰNG MÔ HÌNH PHẦN MỀM

3.3.1 LẬP TRÌNH TRÊN RASPBERRY PI 4

a, Giới thiệu ngôn ngữ lập trình Python

Python là một ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng, do Guido van Rossum tạo ra. Nó dễ dàng để tìm hiểu và đang nổi lên như một trong những ngôn ngữ lập trình nhập môn tốt nhất cho người lần đầu tiếp xúc với ngôn ngữ lập trình. Python hoàn toàn tạo kiểu động và sử dụng cơ chế cấp phát bộ nhớ tự động. Python có cấu trúc dữ liệu cấp cao mạnh mẽ và cách tiếp cận đơn giản nhưng hiệu quả đối với lập trình hướng đối tượng. Cú pháp lệnh của Python là điểm cộng vô cùng lớn vì sự rõ ràng, dễ hiểu và cách gõ linh động làm cho nó nhanh chóng trở thành một ngôn ngữ lý tưởng để viết script và phát triển ứng dụng trong nhiều lĩnh vực, ở hầu hết các nền tảng.



Hình 3.3.1a: Ngôn ngữ Python

b. Lịch sử phát triển của Python

Python là một ngôn ngữ khá cũ được tạo ra bởi Guido Van Rossum. Được thiết kế vào cuối những năm 1980 và được ra lần đầu tiên vào tháng 2 năm 1991.

Các phiên bản Python đã mà nhóm tìm hiểu được

Phiên bản	Ngày phát hành
Python 1.0 (bản phát hành chuẩn đầu tiên) Python 1.6 (Phiên bản 1.x cuối cùng)	01/1994 05/09/2000
Python 2.0 (Giới thiệu list comprehension) Python 2.7 (Phiên bản 2.x cuối cùng)	16/10/2000 03/07/2010
Python 3.0 (Loại bỏ cấu trúc và mô-đun trùng lặp) Python 3.6 Tính năng nổi bật <code>formatted string literals (f-strings)</code>	03/12/2008 23/12/2016
Python 3.10 (Tính năng nổi bật <code>Pattern Matching (PEP 634)</code>) Python 3.12.3 (Bản mới nhất tính đến thời điểm viết bài)	03/08/2021 09/04/2024

Bảng 3.3.1b: Các phiên bản Python đã phát hành

c. Đặc điểm của ngôn ngữ lập trình Python

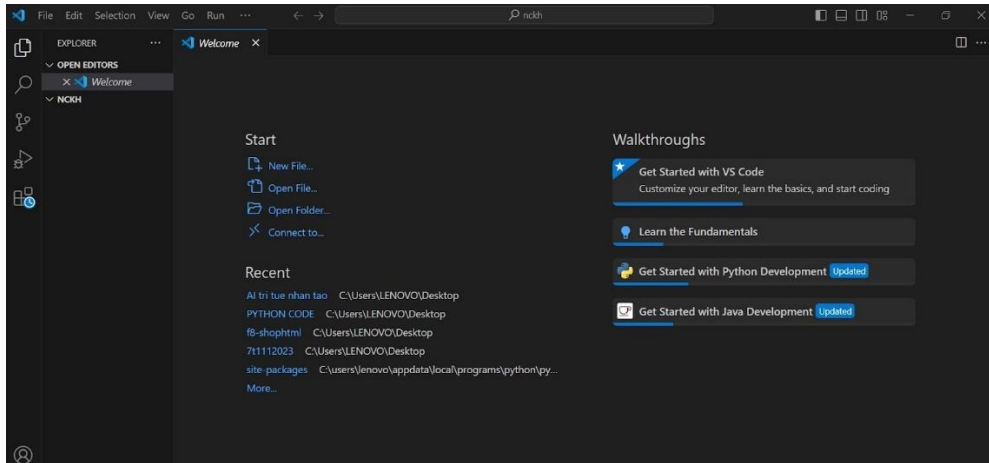
- Đây là một ngôn ngữ lập trình đơn giản ,dễ học: Python có cú pháp rất đơn giản, rõ ràng. Nó dễ đọc và viết, hơn rất nhiều khi so với những ngôn ngữ lập trình khác như C++, C#,Java.
- Miễn phí, mã nguồn mở: Bạn có thể tự do sử dụng và phân phối Python, thậm chí là dùng cho mục đích thương mại. Vì đây là mã nguồn mở, bạn không chỉ có thể sử dụng các phần mềm, chương trình được viết trong python mà còn có thể thay đổi mã nguồn của nó .
- Có khả năng di chuyển: Các chương trình Python có thể di chuyển từ nền tảng này sang nền tảng khác được và chạy nó mà không có bất kỳ thay đổi nào. Nó chạy trên hầu hết tất cả các nền tảng như Windows, macOS, Linux.

- Có thể mở rộng và nhúng: Giả sử một ứng dụng đòi hỏi sự phức tạp rất lớn, bạn có thể dễ dàng kết hợp các phần code bằng C, C++ và những ngôn ngữ khác (có thể gọi được từ C) vào code Python. Điều này sẽ cung cấp cho ứng dụng của bạn những tính năng tốt hơn cũng như khả năng scripting mà những ngôn ngữ lập trình khác khó có thể làm được.
- Ngôn ngữ thông dịch cấp cao: Không giống như C/C++, với Python, bạn không phải lo lắng những nhiệm vụ khó khăn như quản lý bộ nhớ, dọn dẹp những dữ liệu thừa ,... Khi ta chạy code Python, nó sẽ tự động chuyển đổi code sang dạng ngôn ngữ máy tính có thể hiểu được.
- Có thư viện tiêu chuẩn lớn để giải quyết những tác vụ phổ biến: Python có một số lượng lớn thư viện tiêu chuẩn giúp cho công việc lập trình của bạn trở nên dễ thở hơn rất nhiều, đơn giản vì không phải tự viết tất cả code. Những thư viện này đã được kiểm tra kỹ lưỡng và được sử dụng bởi hàng trăm người.
- Hướng đối tượng: Mọi thứ trong Python đều là hướng đối tượng. Lập trình hướng đối tượng (OOP) giúp giải quyết những vấn đề phức tạp một cách trực quan. Với OOP, bạn có thể phân chia những vấn đề phức tạp thành những tập nhỏ hơn bằng cách tạo ra các đối tượng.

3.3.2. Giới thiệu phần mềm lập trình Visual Studio Code

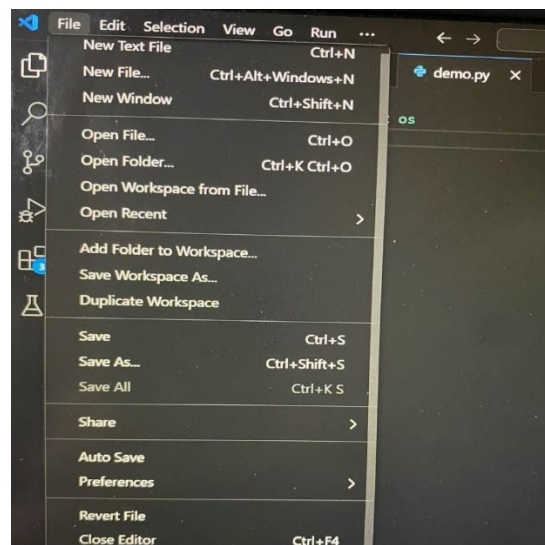


Hình 3.3.2: Logo của Visual Studio Code



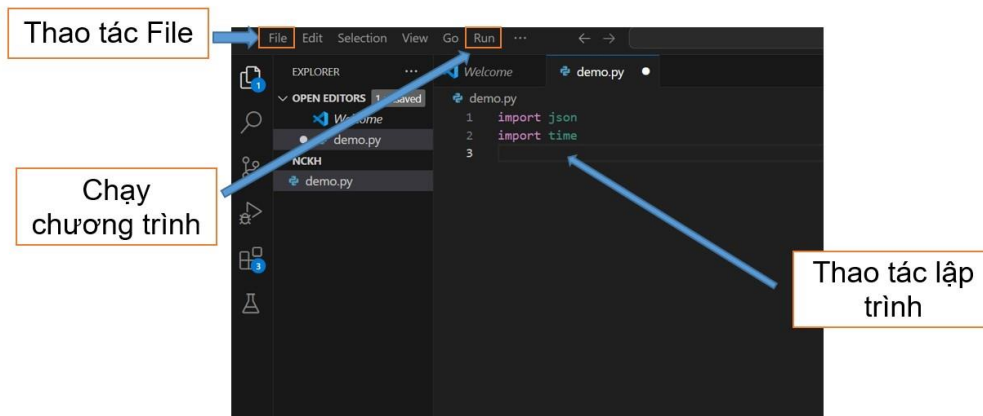
Hình 3.3.2.1: Giao diện Visual Studio Code

Giao diện Visual Studio Code rất đơn giản và dễ sử dụng, để tạo File mới hoặc Open một file có sẵn ta click chọn File rồi chọn các mục tương ứng.



Hình 3.3.2.2: Thao tác file trên giao diện Visual Studio Code

Hình dưới đây chỉ ra các chức năng các thành phần và thanh công cụ của Visual Studio Code

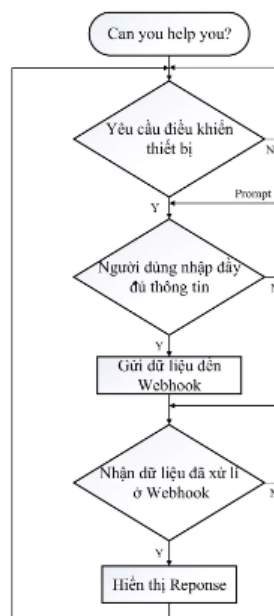


Hình 3.3.2.3: Các phần của giao diện

3.3.3 Xây dựng dialogflow

a, Lưu đồ giải thuật Dialogflow

Lưu đồ khi người dùng yêu cầu điều khiển các thiết bị đến Chatbot thông qua text để trò chuyện :



Hình 3.3.3.1: Lưu đồ thiết kế thực hiện trên Dialogflow

Giải thích lưu đồ:

Bước 1: Người dùng truy cập internet và nhận link để trò chuyện với chatbot..

Bước 2: Phân tích câu nói người dùng để phát hiện yêu cầu điều khiển thiết bị . Nếu thông tin không đầy đủ, Chatbot sẽ yêu cầu người dùng cung cấp thêm thông tin.

Bước 3: Chatbot lưu thông tin dưới dạng file JSON và gửi đến địa chỉ webhook đã được lưu ở mục Fulfillment để xử lý.

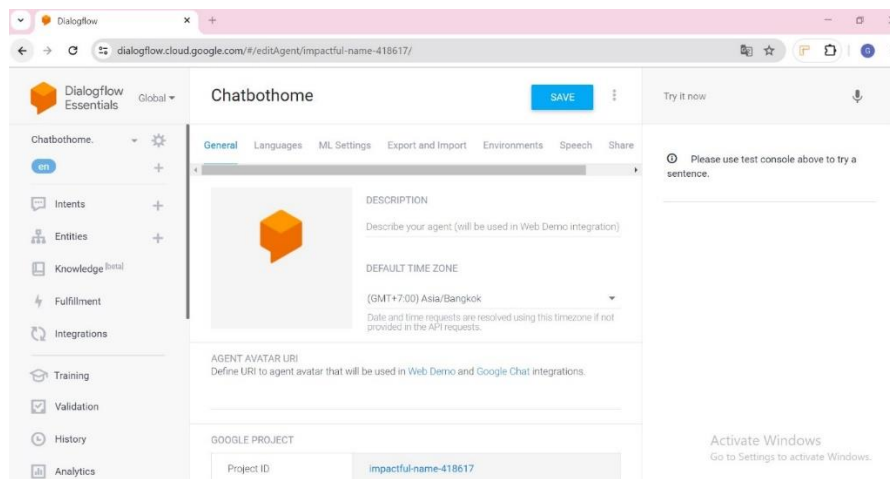
Bước 4: Chatbot nhận kết quả trả lại từ webhook và hiển thị cho người dùng.

Bước 5: Chatbot sẵn sàng nhận yêu cầu trò chuyện hay điều khiển tiếp theo từ người dùng.

b, Thiết kế cho yêu cầu điều khiển ON-OFF

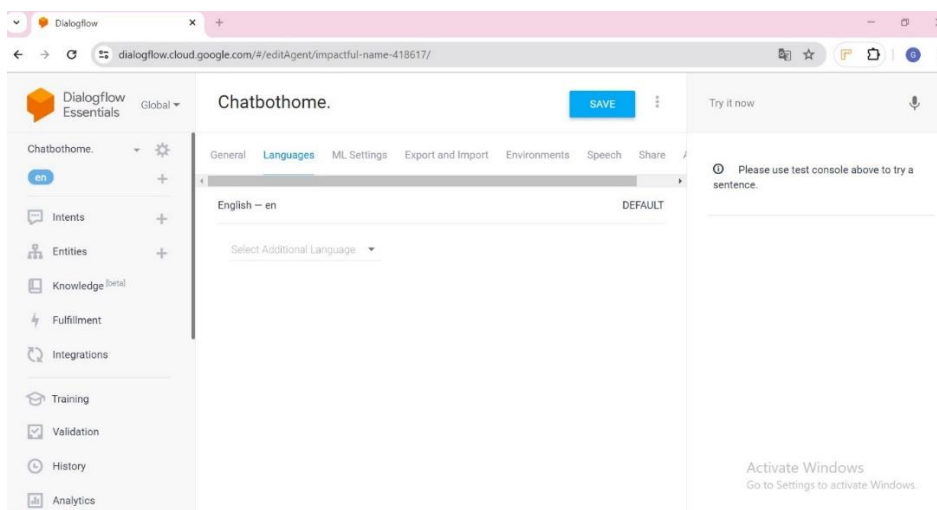
- Tạo mới Agent

Để bắt đầu với Dialogflow, ta cần tạo một Agent mới. Trong trường hợp này, nhóm em đã tạo 1 Agent với tên là "Chatbothome". Sau khi đã chọn các cài đặt cần thiết chẳng hạn như múi giờ +7, ta có thể tiến hành lưu Agent của mình để tiếp tục cấu hình và phát triển nó.



Hình 3.3.3.2: Tạo mới Agent

Tiến hành chuyển sang tab **Languages** để chọn ngôn ngữ.



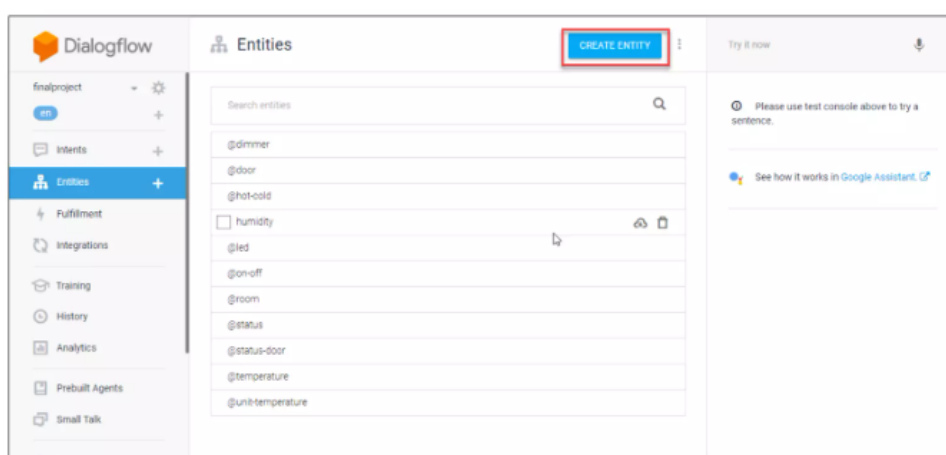
Hình 3.3.3.3: Cài đặt các thông số cho Agent

Sau khi đã thiết lập các thông số, ta tiến hành lưu lại và chuyển sang phần giao diện hỗ trợ cho lập trình viên để tạo 1 Chatbot.

- Tạo entities

Sau khi đã tạo mới Agent, ta có thể chuyển sang tab "Entities" để tạo mới các thực thể. Trong trường hợp này, ta cần xác định hai thực thể cần làm việc là "điều khiển đèn nào" và "trạng thái cần điều khiển của đèn đó". Dựa trên điều này, mình sẽ tạo hai thực thể là "@on-off" và "@led" (tên thực thể được lập trình viên tự đặt sao cho dễ hiểu và sử dụng nhất).

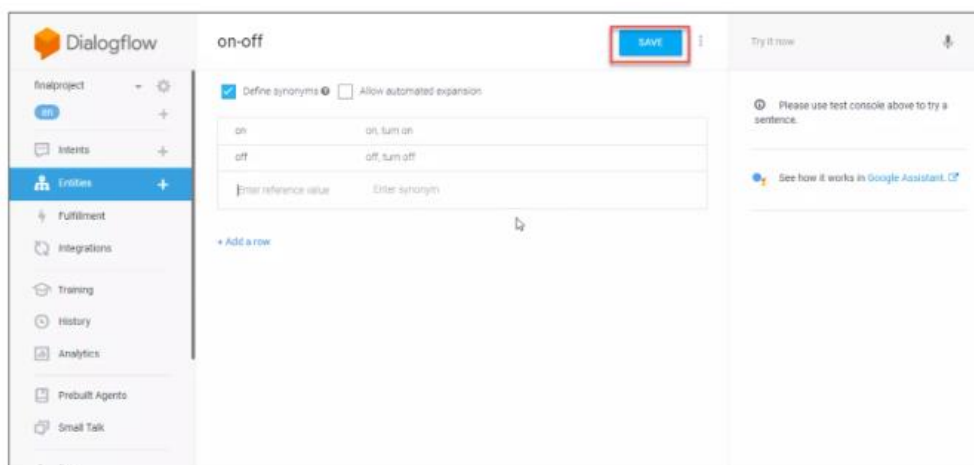
Sau khi đã xác định được những entities cần tạo, ta tiến hành tạo mới entities.



Hình 3.3.3.4: Thực hiện tạo mới Entities

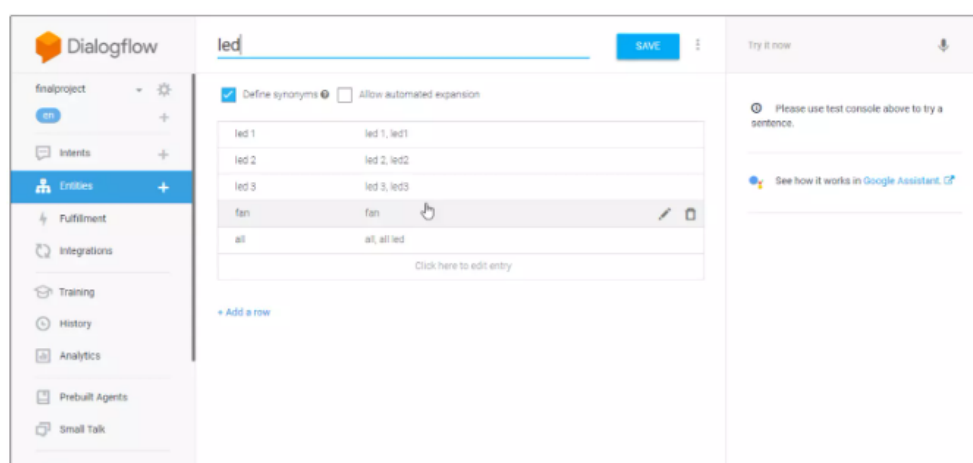
Sau khi nhấn vào tạo mới, tiếp theo ta tiến hành đặt tên cho entity và nhập các giá trị. Như hình dưới đây, đèn chỉ có 2 trạng thái là ON/OFF nên ta điền giá trị ON và OFF vào reference

value, còn phần Synonym, ta điền vào các từ đồng nghĩa hoặc ngầm hiểu là điều khiển ON/OFF đèn tương ứng như hình dưới đây:



Hình 3.3.3.5: Hình ảnh Entities on – off

Sau khi đã hoàn thành, ta tiến hành ấn Save để lưu lại. Ta cũng thực hiện tương tự đối với Entity @led.

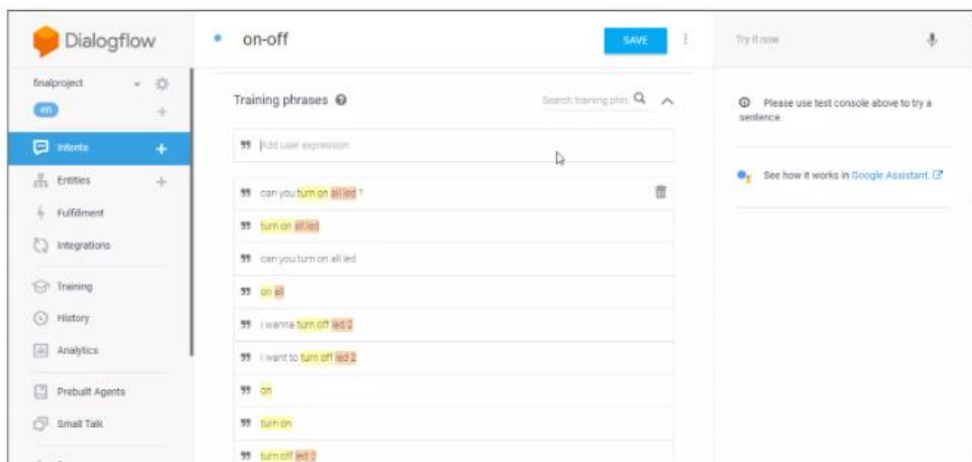


Hình 3.3.3.6: Hình ảnh Entities led

- Tạo Intent

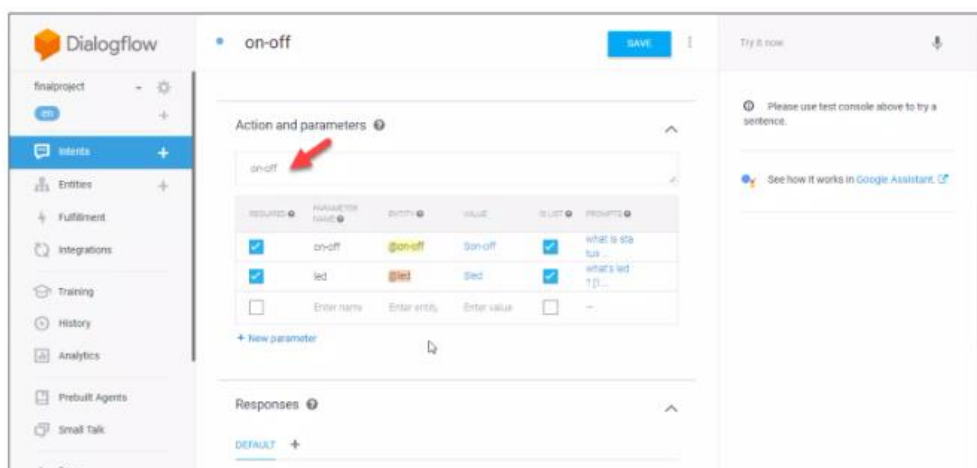
Để phân tích lời nói của người dùng và xử lý yêu cầu của họ, ta cần tạo mới các intent trong Dialogflow. Tương tự như việc tạo mới Entities, bạn nhấp vào nút "**Create Intent**" để bắt đầu tạo mới Intent.

Ta tiến hành thao tác để tạo ra Intent **on-off** giúp Chatbot hiểu yêu cầu ON/OFF các thiết bị từ người dùng và đưa ra hành động hợp lí.



Hình 3.3.3.7: Tạo mới Intent

Ở mục **Training Phrases**, ta tiến hành nhập các câu nói mà người dùng có thể nói để điều khiển thiết bị, Dialogflow sẽ tự động **Highlight** các giá trị của tham số trong **entities** vừa tạo ở trên để trích xuất lấy dữ liệu từ voice (câu nói) của người dùng, từ đó xử lí và đưa ra kết quả phù hợp. Ta cũng có thể chỉnh sửa thủ công bằng cách bôi đen các từ muốn Dialogflow gắn tham số. Các giá trị tham số của **Training Phrases** sẽ được hiển thị trong phần “**Action and parameters**”.

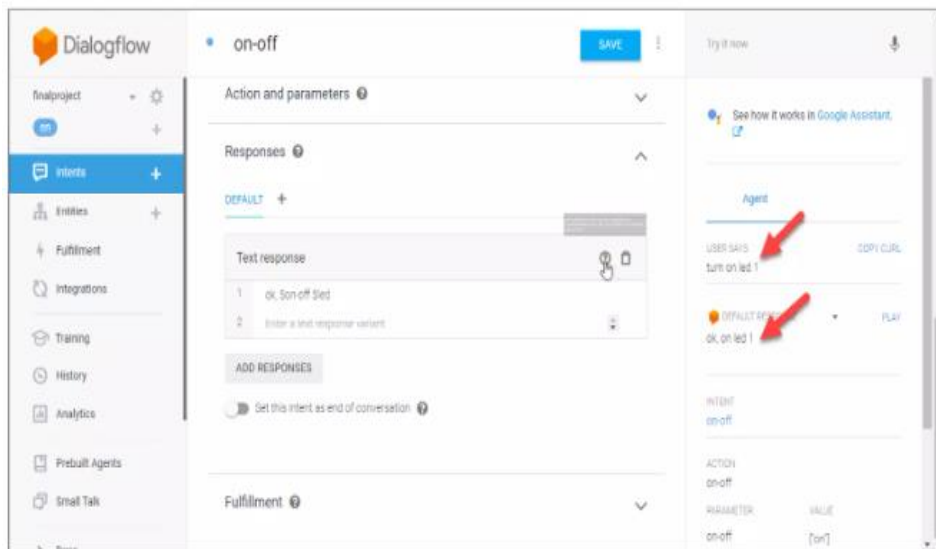


Hình 3.3.3.8: Bảng Action và Parameters

Ở đây **action** là **on:off**. Action này giúp ta xác định đâu là Intent được gửi tới webhook để có thể xử lí và đưa ra kết quả phù hợp với yêu cầu người dùng. Và như ta thấy ở trên thì hai tham số là on-off và led vừa tạo ở entity được sử dụng để lấy giá trị dữ liệu từ người dùng. Ở

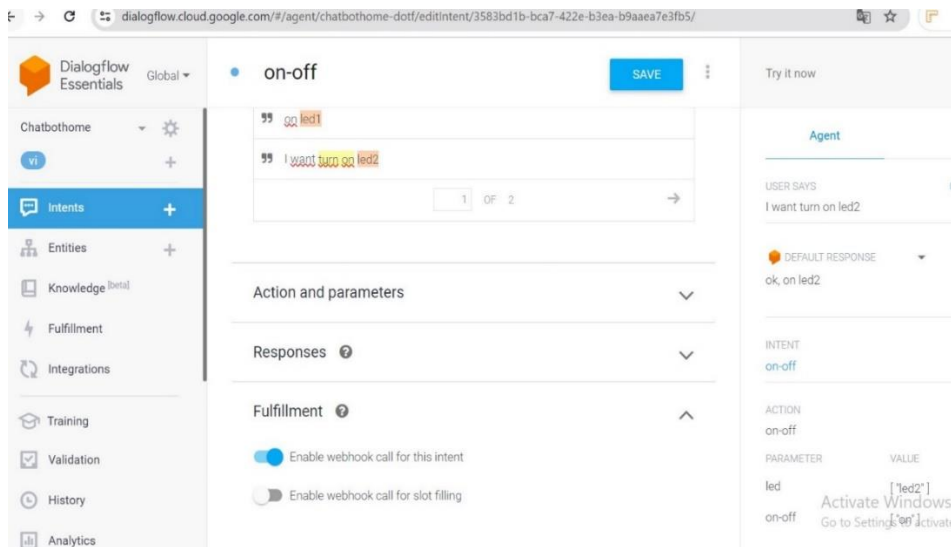
đây, ta yêu cầu User (người dùng) cung cấp đầy đủ thông tin bằng việc tích vào mục **Required**.

Tiếp theo là mục **Response**, đây là nơi chúng ta sẽ phản hồi lại các câu trả lời tương ứng cho người dùng mỗi khi Intent được gọi. Ở đây, nhóm em sử dụng giá trị của tham số trong chính câu nói của người dùng để trả lời. Như hình dưới đây, ta có thể thấy được sau khi người dùng yêu cầu **“Turn on led 1”** thì Chatbot sẽ trả lời các yêu cầu của người dùng theo định dạng ta đã đặt ở **“Text response”**.



Hình 3.3.3.9: Hình ảnh mục Responses

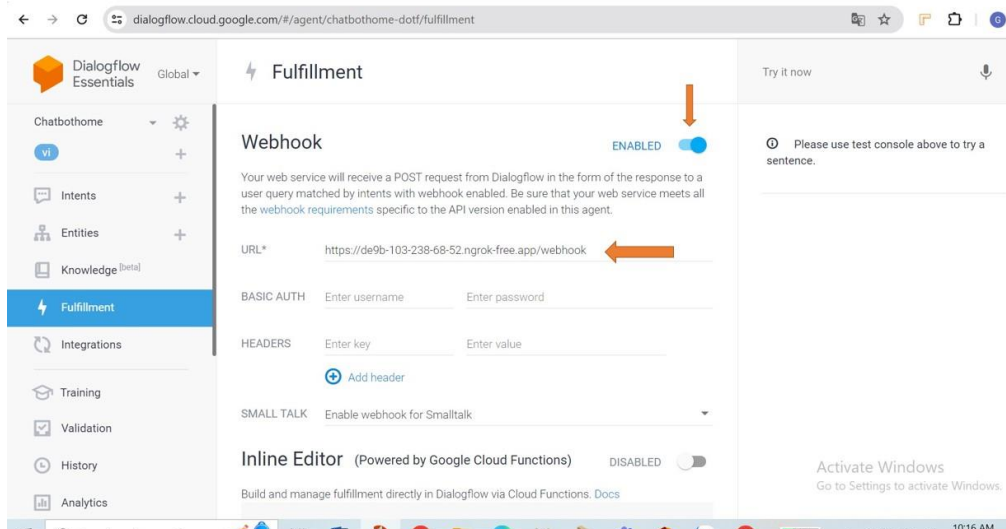
Cuối cùng, trong phần **"fulfillment"**, bạn nhấp vào **"Enable webhook call for this intent"** để khi Intent được kích hoạt, thông tin sẽ được chuyển đổi thành định dạng JSON và gửi đến webhook. Server sẽ nhận và xử lý thông tin này, sau đó trả về cho người dùng câu trả lời phù hợp với yêu cầu của họ thay vì chỉ đơn giản trả lời theo định dạng mặc định được cung cấp ở phần **"Response"** trên giao diện.



Hình 3.3.3.10: Kích hoạt Fulfill cho Intent

Sau khi đã hoàn thành ta tiến hành **“Save”** để lưu lại cho Intent.

- Fulfillment



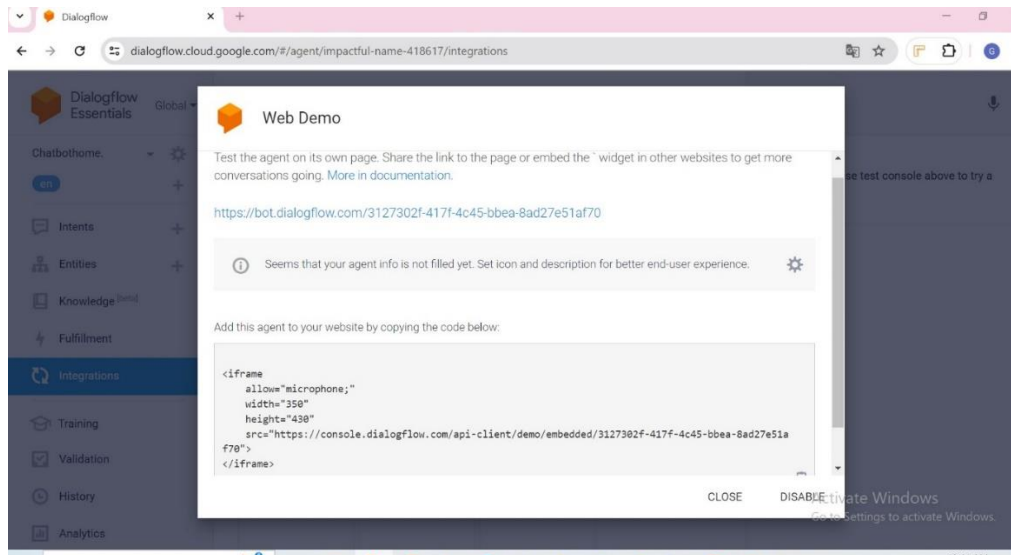
Hình 3.3.3.11: Đặt link webhook cho Fulfillment

Đây là nơi đặt link của webhook, Dialogflow sẽ gọi đến địa chỉ này mỗi khi có Intent đã kích hoạt webhook được gọi đến. Đây sẽ là nơi tiếp nhận thông tin để server có thể xử lý và trả lại kết quả cho Dialogflow để hiển thị cho người dùng.

Ta cần kích hoạt Enable ở mục webhook và đặt link của webhook cần gọi vào mục URL, tiến hành nhấn Save để hoàn tất.

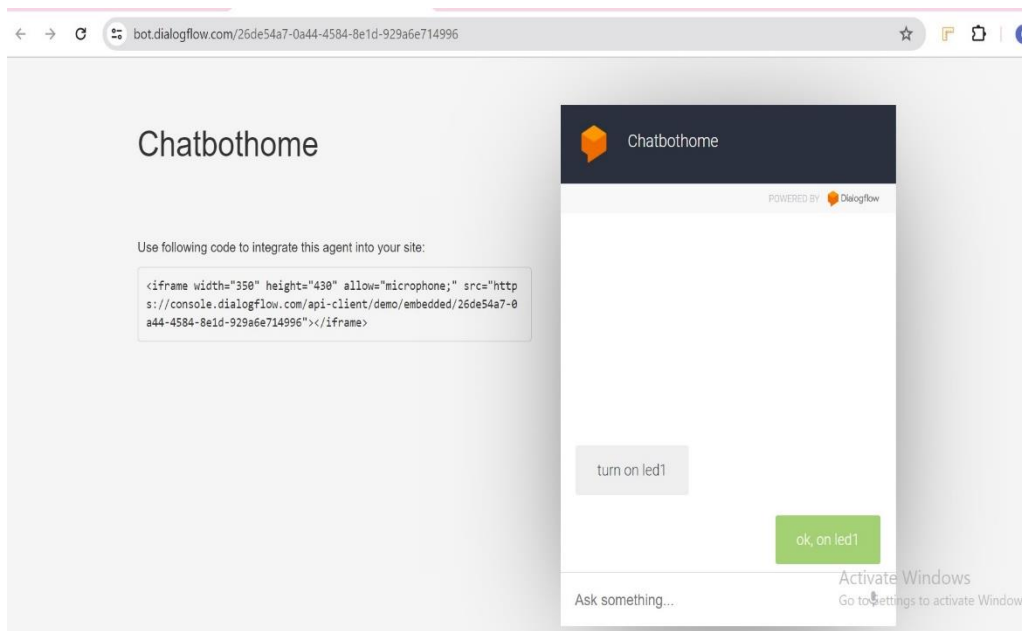
- Integrations

Ta tích hợp Chatbot vào nền tảng nhắn tin, ở đây nhóm em sử dụng webdemo do Dialogflow hỗ trợ để tạo giao diện nhắn tin với người dùng. Enable web demo để kích hoạt tính năng này.



Hình 3.3.3.12: Phần tùy chỉnh trong Integrations

Sau khi enable web demo, chúng ta sẽ thấy đường link. Chúng ta sẽ trò chuyện với Chatbot thông qua đường link này.



Hình 3.3.3.13: Giao diện trò chuyện Chatbot

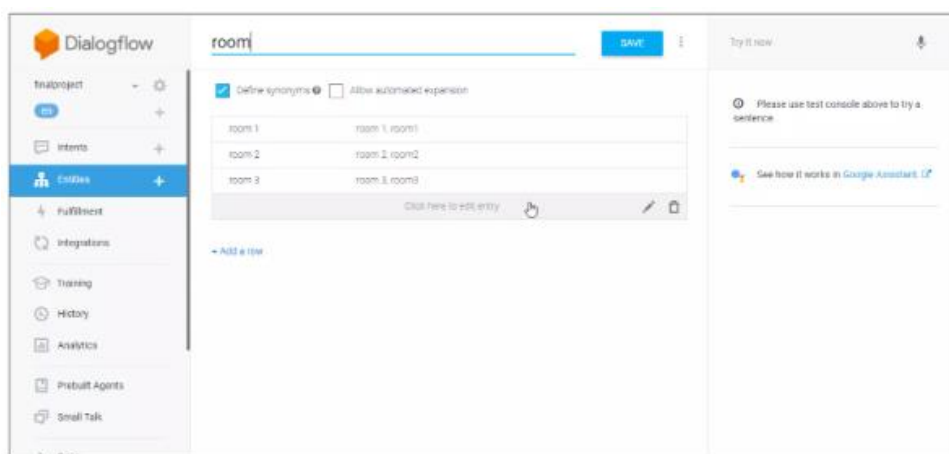
Đây là giao diện khi ta click vào đường dẫn ở trên, ta trò chuyện với Chatbot thông qua hộp thoại này.

c, Mở rộng cho điều khiển ON-OFF

- Thêm mới entities

Một số trường hợp thay vì yêu cầu điều khiển trực tiếp các đèn thì người dùng có thể yêu cầu bật hay tắt thiết bị ở phòng nào, lúc nào chúng ta cần lập trình cho chatbot hiểu được người dùng đang muốn gì để đáp ứng đúng yêu cầu người dùng.

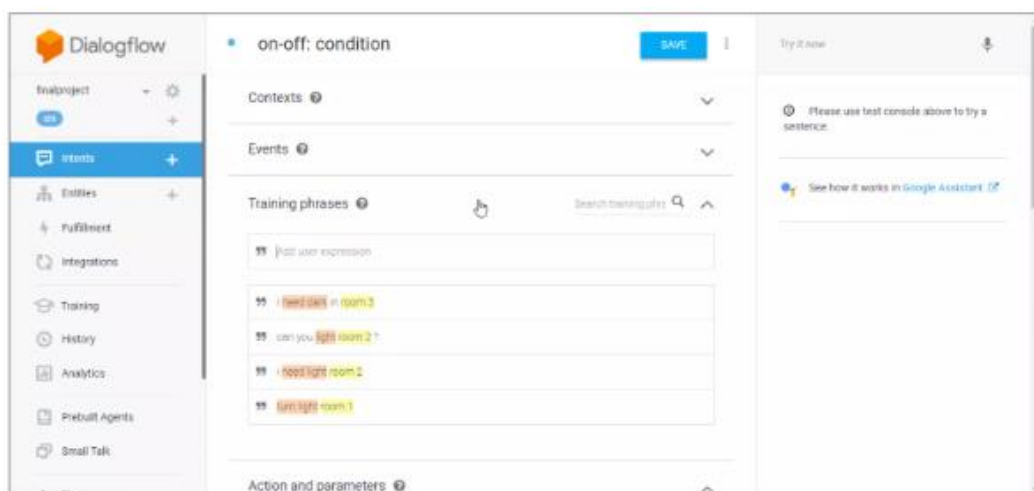
Lúc này ta phải tạo mới entities là **@room** để đáp ứng nhu cầu thấp sáng hay tắt của từng phòng.



Hình 3.3.3.14: Thêm mới entities room

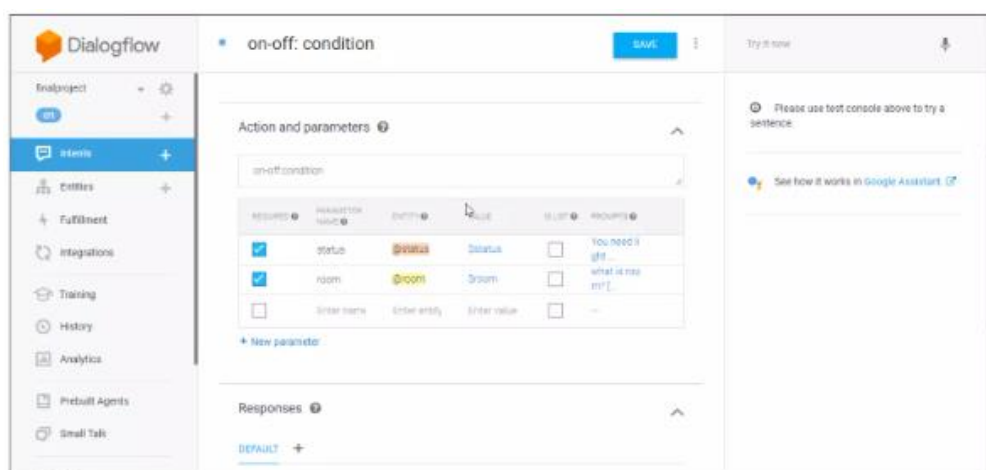
- Thêm mới Intent

Chúng ta tiến hành thêm mới Intent và thao tác tương tự như các bước ở trên. Ở đây nhóm em đã tạo ra Intent **on-off:condition**. Sau đây là hình ảnh của Intent này.



Hình 3.3.3.15: Thêm mới Intent on-off condition

Vẫn tương tự như trường hợp ON-OFF thông thường như ở trên, khi người dùng cung cấp thiếu thông tin thì Chatbot sẽ hỏi người dùng để cung cấp đầy đủ thông tin.



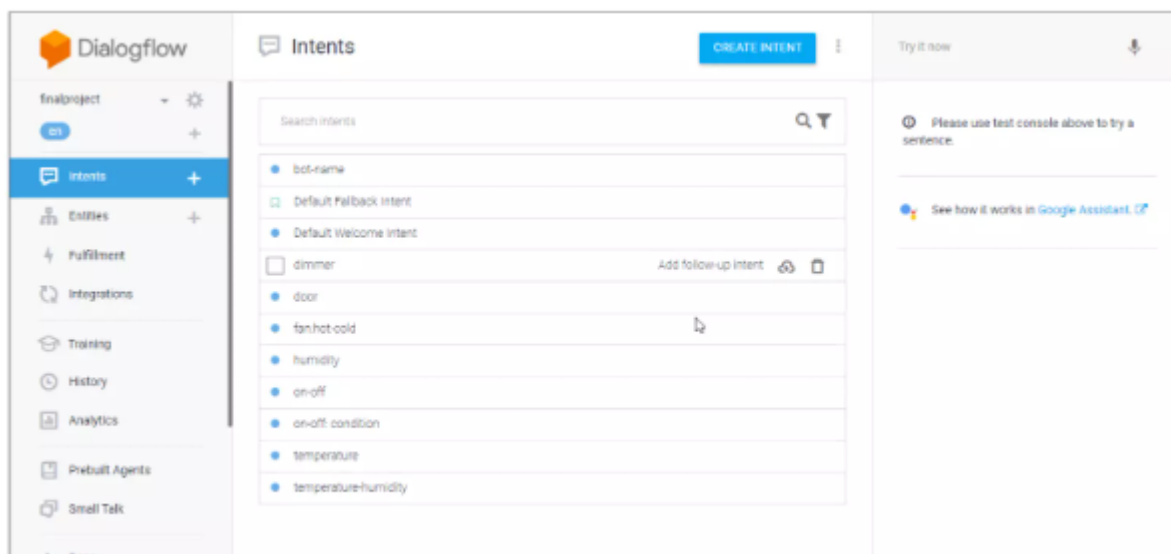
Hình 3.3.3.16: Kết quả thao tác trên Intent mới

Các bước tiếp theo, nhóm em thực hiện tương tự như trường hợp ON-OFF thông thường ở trên.

d, Kết quả hoàn chỉnh trên Dialogflow

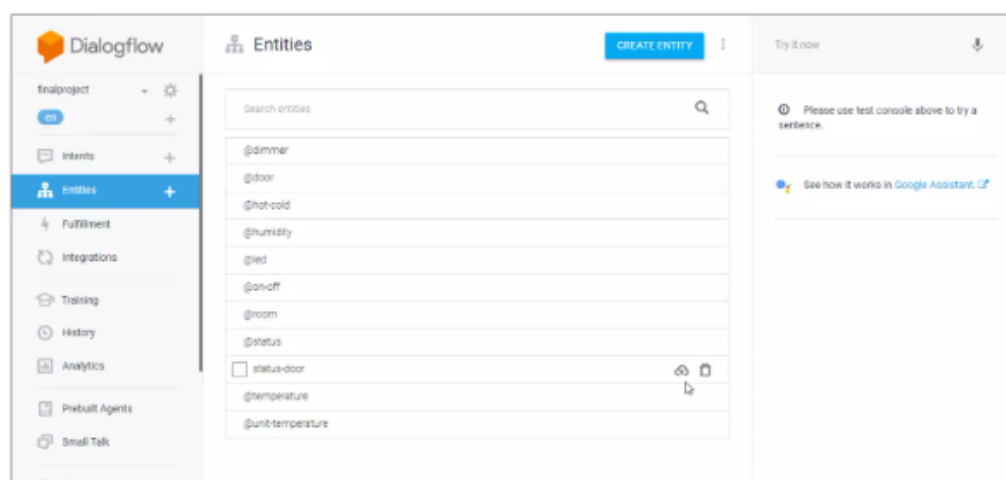
Chúng ta thực hiện tương tự như các bước ở trên để đáp ứng những yêu cầu của người dùng như đóng mở thiết bị, hay đọc giá trị từ cảm biến. Sau đây là một số hình ảnh thực hành hoàn chỉnh trên Dialogflow.

- Intents



Hình 3.3.3.17: Kết quả hoàn thành Intent

- Entities



Hình 3.3.3.18: Kết quả hoàn thành Entities

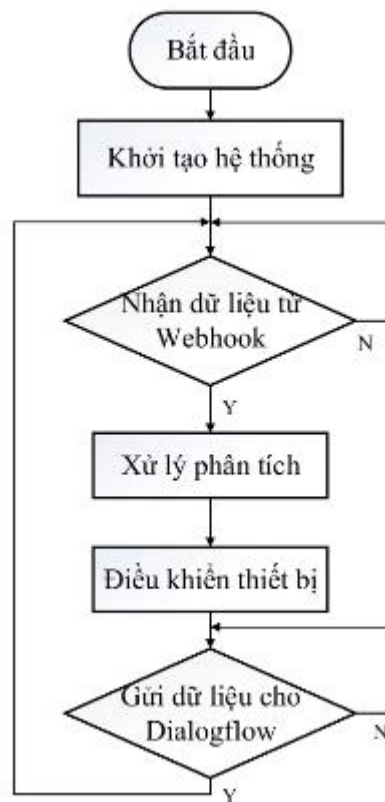
3.3.4 Lưu đồ giải thuật

Giải thích lưu đồ:

Các bước thực hiện được miêu tả bởi lưu đồ dưới (Hình 3.3.4).

- Bước 1: Ta cấp nguồn cho hệ thống, hệ thống bắt đầu khởi động và kết nối với mạng internet.
- Bước 2: Sau khi khởi tạo hệ thống, nếu có dữ liệu được từ Dialogflow gửi đến thì hệ thống sẽ thực hiện xử lý, phân tích dữ liệu nhận được, từ đó thực hiện yêu cầu điều khiển thiết bị và tạo ra câu trả lời phù hợp với từng yêu cầu điều khiển từ người dùng.
- Bước 3: Tiếp đến Raspberry sẽ gửi câu trả lời đến Dialogflow để hiển thị câu trả lời cho người dùng.
- Bước 4: Sẵn sàng nhận dữ liệu tiếp theo từ Dialogflow để xử lý và điều khiển thiết bị.

Lưu đồ điều khiển trên raspberry:



Hình 3.3.4: Lưu đồ điều khiển trên Raspberry Pi

3.3.5 Viết chương trình hệ thống

Ở đây em tiến hành viết chương trình tạo webhook để nhận dữ liệu từ Dialogflow và sử dụng dữ liệu đó để điều khiển, đồng thời gửi phản hồi về cho Dialogflow. Dưới đây là một

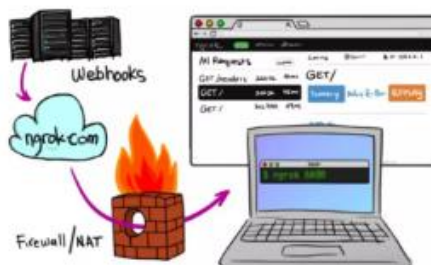
đoạn ngắn của chương trình tạo webhook, vì source code khá dài nên nhóm em xin phép để đầy đủ source code ở phần phụ lục của báo cáo.

```
*IDLE Shell 3.10.4*
Edit Shell Debug Options Window Help
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64
AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more information.
> import json
. import time
. import Adafruit_DHT
. from flask import Flask, request, make_response
. import RPi.GPIO as GPIO
.
. GPIO.setwarnings(False)
. GPIO.setmode(GPIO.BCM)
. GPIO.setup(18, GPIO.OUT) # LED 1
. GPIO.setup(23, GPIO.OUT) # LED 2
. GPIO.setup(24, GPIO.OUT) # LED 3
. GPIO.setup(25, GPIO.OUT) # led4
.
. GPIO.setup(21, GPIO.OUT) #fan
.
. # Khai báo chân GPIO cho servo
. servo_pin = 20
. GPIO.setup(servo_pin, GPIO.OUT)
. pwm = GPIO.PWM(servo_pin, 50)
. pwm.start(0)
. # pwm = GPIO.PWM(25, 500)
. # pwm.start(100)
.
. # Hàm để đặt góc quay của servo
. def set_angle(angle):
.     duty = angle / 20 + 2
.     GPIO.output(servo_pin, True)
.     time.sleep(0.01)
.     GPIO.output(servo_pin, False)
```

Hình 3.3.5: Một đoạn chương trình hệ thống

3.3.6 Public server lên internet bằng ngrok

Ở phần trên ta chỉ mới tạo ra webhook ở local host, để public server lên internet ta cần dùng ngrok. Vậy ngrok là gì ? Ngrok là công cụ tạo đường hầm (tunnel) giữa localhost của bạn và internet. Giúp người khác mạng có thể truy cập được localhost của bạn thông qua custom domain của ngrok.

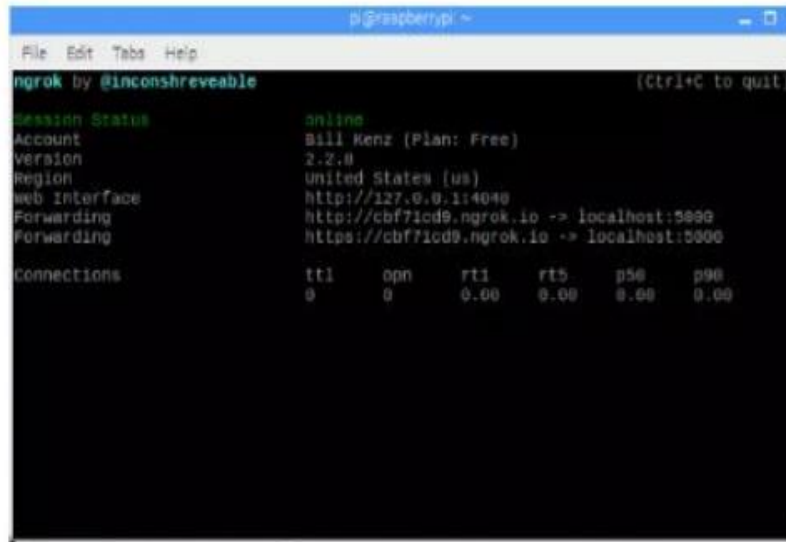


Hình 3.3.6.1: Giới thiệu ngrok

Ngrok có bản trả phí lẫn miễn phí, tuy nhiên bản miễn phí sẽ có nhiều hạn chế. Để có thể sử dụng ngrok ta tiến hành các bước sau:

- Đăng ký tài khoản free trên trang web: <https://ngrok.com/>
- Download ngrok về máy, tiến hành giải nén.
- Vào folder bạn vừa giải nén, chạy câu lệnh `/ngrok http [port bạn muốn mở]`.

Ví dụ kết quả sau khi chạy port 5000 như hình dưới:



```
p@raspberrypi ~$ ngrok
ngrok by @inconshreveable          (Ctrl+C to quit)

Session Status               online
Account                      Bill Kenz (Plan: Free)
Version                      2.2.0
Region                      United States (us)
Web Interface                http://127.0.0.1:4040
Forwarding                   http://cbf71cd9.ngrok.io -> localhost:5000
                             https://cbf71cd9.ngrok.io -> localhost:5000
Connections                  ttl    opn    rt1    rt5    p50    p90
                                0      0      0.00   0.00   0.00   0.00
```

Hình 3.3.6.2: Hình ảnh chạy ngrok

3.4 XÂY DỰNG MÔ HÌNH PHẦN CỨNG

3.4.1 Đóng gói bộ điều khiển

Để bảo vệ raspberry pi 4 khỏi tác động bên ngoài, chúng em dùng vỏ nhựa để bảo vệ raspberry. Đồng thời trên vỏ cũng trang bị quạt tản nhiệt mini để giúp raspberry hoạt động tốt hơn. Dưới đây là một số hình ảnh :

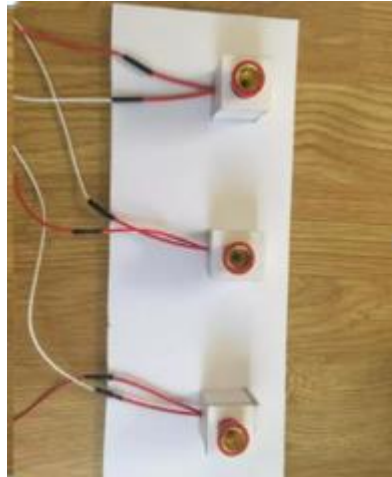


Hình 3.4.1: Vỏ nhựa kết hợp quạt tản nhiệt

3.4.2 Thi công mô hình

Để tạo mô hình nhà thông minh, nhóm em sử dụng vật liệu là bìa cứng. Sử dụng các dụng cụ để cắt ghép các phần lại với nhau để tạo mô hình ngôi nhà hoàn chỉnh.

- Thực hiện đính các thiết bị đèn vào một khối cố định như hình dưới.



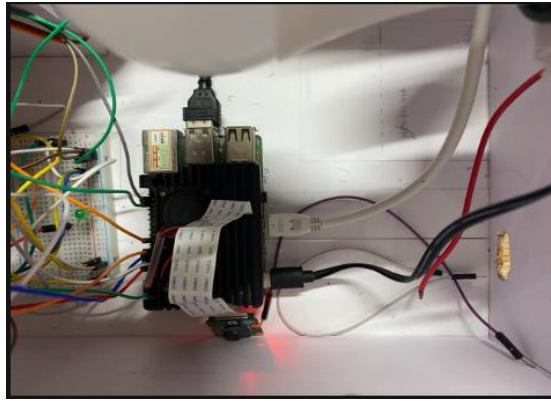
Hình 3.4.2.1: Thi công khối đèn xoay chiều

- Sau đó kết nối đèn với thiết bị điều khiển công suất Relay.
- Tiếp đến là lắp ráp cảm biến DHT11 và động cơ Servo vào đúng vị trí.

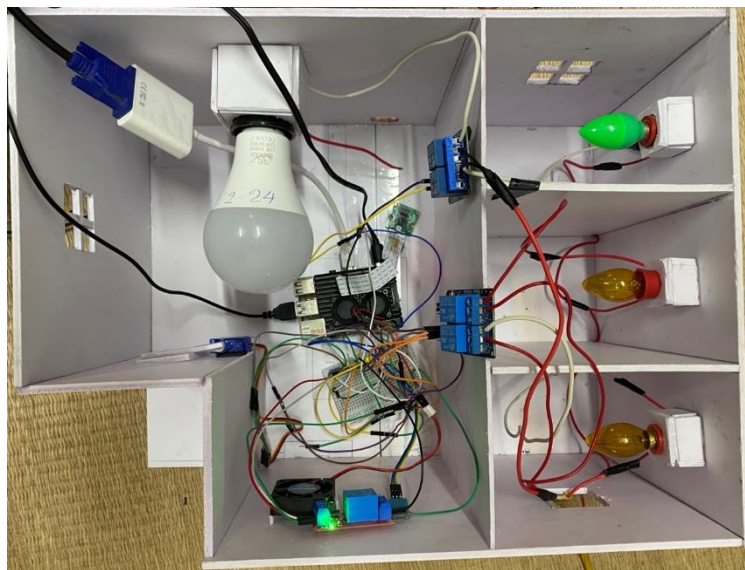


Hình 3.4.2.2: Động cơ servo đặt vào đúng vị trí

- Cuối cùng ta kết nối các bộ phận với bộ điều khiển Raspberry pi 4.



Hình 3.4.2.3: Kết nối với Raspberry Pi



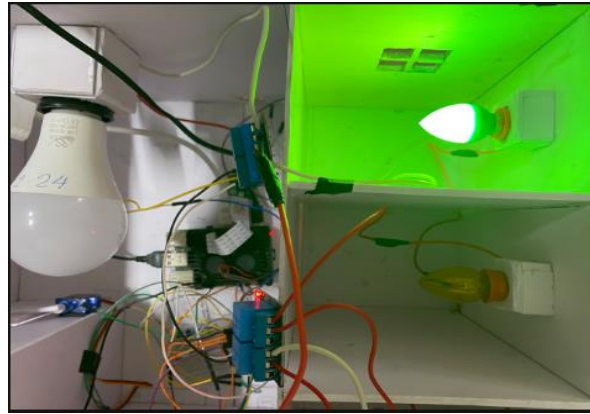
Hình 3.4.2.4: Hình ảnh mô hình thi công hoàn chỉnh

3.5 KẾT QUẢ VẬN HÀNH HỆ THỐNG

Sau khi người dùng chat :”Turn on Led 2” thì lập tức đèn 2 mở, đồng thời Chatbot trả lời lại cho người dùng.

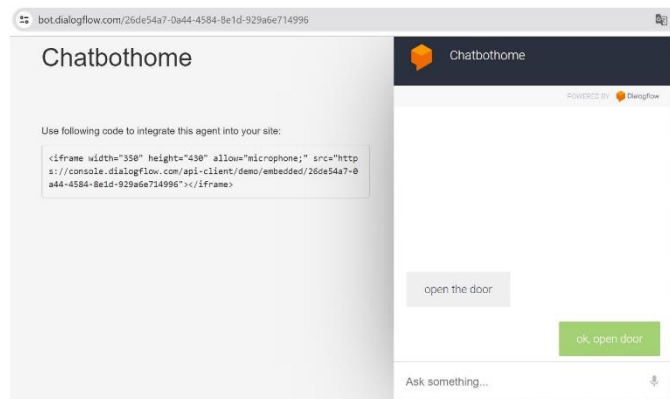


Hình 3.5.1: Điều khiển thiết bị thông qua giao diện



Hình 3.5.2: Kết quả sau khi nhận yêu cầu điều khiển

Tương tự khi người dùng yêu cầu mở cửa.

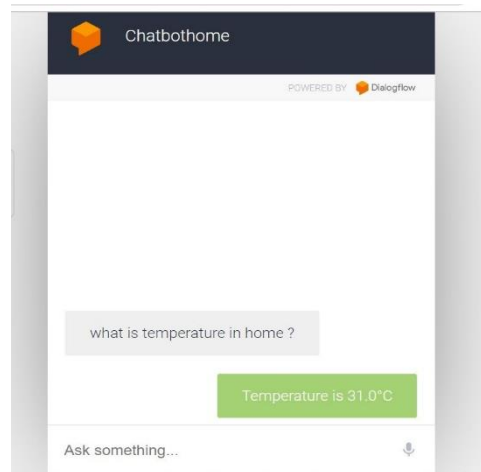


Hình 3.5.3: Yêu cầu điều khiển đóng mở cửa



Hình 3.5.4: Thực hiện mở cửa

Và trả lời thông tin nhiệt độ, độ ẩm khi người dùng thực hiện yêu cầu.



Hình 3.5.5: Người dùng thông tin nhiệt độ

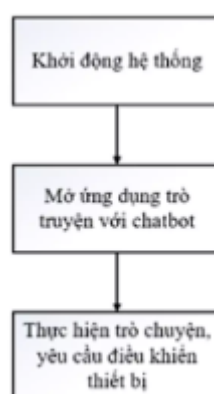
3.6 HƯỚNG DẪN SỬ DỤNG ,THAO TÁC

3.6.1 Hướng dẫn sử dụng

- Bước 1: Cấp nguồn cho hệ thống, hệ thống hoạt động sử dụng nguồn 220V AC.
- Bước 2: Sau khi khởi động hệ thống, tiến hành mở ứng dụng Chatbot và trò chuyện với Chatbot thông qua text message.
- Bước 3: Trò chuyện và yêu cầu điều khiển thiết bị đối với Chatbot.

3.6.2 Quy trình thao tác

Để dễ dàng thao tác cho người sử dụng , ta có lưu đồ vận hành hệ thống:



Hình 3.6: Lưu đồ hướng dẫn sử dụng

3.7 KẾT QUẢ NHẬN XÉT ĐÁNH GIÁ

Để đáp ứng được yêu cầu của dự án, nhóm chúng tôi đã tập trung nghiên cứu và hoàn thành các đề tài được chỉ định. Quá trình nghiên cứu kéo dài 15 tuần, trong thời gian đó nhóm không chỉ học được nhiều kiến thức mới mà còn ôn lại rất nhiều kiến thức cũ, bị lãng quên. Đây là những gì chúng tôi đã học được.

3.7.1 DIALOGFLOW

a, Dialogflow

Tìm hiểu cách sử dụng và tận dụng các tính năng mạnh mẽ mà Dialogflow cung cấp cho các lập trình viên để tạo các chatbot thông minh. Hiểu ý nghĩa từng phần trong Dialogflow, chẳng hạn như:

- Agent
- Intents
- Contexts
- Events
- Training Phrases
- Action and parameters
- Responses
- Fulfillment
- Entities
- Integrations
- Training
- History
- Analytics
- Prebuilt Agents

b, Các khái niệm liên quan

- Hiểu các khái niệm như API, webhooks, định dạng file Json và cách trích xuất dữ liệu từ định dạng file Json đáp ứng yêu cầu của đề tài.

3.7.2 RASPBERRY PI 4

a, Raspberry Pi 4

- Tìm hiểu cách cài đặt hệ điều hành cho Raspberry Pi, đặt IP tĩnh cho mạng Lan và điều khiển Raspberry Pi qua VNC khi Raspberry Pi 4 không có kết nối mạng.
- Cài đặt và cập nhật các gói cần thiết cho Raspberry Pi của bạn.
- Lập trình bằng ngôn ngữ Python, cài đặt các thư viện Python cần thiết để lập trình trong chuyên đề này. Tìm hiểu cách sử dụng các chức năng của thư viện, chẳng hạn như:
 - GPIO.setwarnings()
 - GPIO.setmode()
 - GPIO.setup()
 - GPIO.PWM()

Hiểu các điện áp do Raspberry Pi 4 cung cấp và ý nghĩa của các chân trên Raspberry, ví dụ chân Raspberry chỉ có thể đọc tín hiệu số làm đầu vào, không có chân analog như trên Arduino. và học các chuẩn giao tiếp trên Raspberry

b, Công cụ ngrok

Biết sử dụng ngrok để vào public server xem xét và quản lý các task thực hiện trên server.

3.7.3 PHẦN CỨNG

a. Động cơ Servo

Hiểu được việc sử dụng xung điều khiển động cơ servo sẽ giúp bạn hiểu được nguyên lý cơ bản của việc điều chỉnh góc quay để đóng mở cửa.

b, Module Relay 2 kênh 5V

Hiểu cách hoạt động của các mô-đun role và tính toán dòng điện cần thiết để kích hoạt role. Từ đó các nhiệm vụ điều khiển thiết bị được thực hiện.

c, Module cảm biến DHT11

Tìm hiểu cách sử dụng mô-đun cảm biến nhiệt độ và độ ẩm DHT11 để cung cấp dữ liệu nhiệt độ và độ ẩm hiện tại cho dự án .

3.7.4 NHẬN XÉT VÀ ĐÁNH GIÁ

Nhóm thực hiện dự án đã hoàn thành các yêu cầu của dự án và dự án nghiên cứu có những ưu điểm sau:

- Việc ứng dụng trí tuệ nhân tạo và học máy vào nhiều lĩnh vực của đời sống đã trở thành xu hướng nổi bật trong những năm gần đây, đặc biệt là trong ứng dụng thực tế đã mang lại những kết quả đáng kể.
- Chi phí xây dựng hệ thống không cao và tương đối dễ sử dụng, vận hành. Quá trình bảo trì hệ thống rất đơn giản
- Giao diện Chatbot dễ sử dụng, đặc biệt có thể tích hợp với tính năng nhắn tin Facebook, Skype, Slack và các nền tảng nhắn tin khác để tăng tính thực tế và gần gũi với người dùng.
- Hệ thống điều khiển thời gian thực, có thể điều khiển ngay lập tức các thiết bị trong nhà khi người dùng có yêu cầu điều khiển.
- Dựa trên lịch sử trò chuyện, người dùng có thể xem lịch sử đóng mở của các thiết bị trong nhà mình.

Thay vào đó, hệ thống cũng tồn tại một số hạn chế như:

- Hệ thống điều khiển được thực hiện thông qua Internet nên cần có kết nối mạng ổn định và lâu dài.
- Hệ thống không an toàn lắm và bất kỳ ai có liên kết trò chuyện đều có thể trò chuyện với chatbot.
- Hệ thống sử dụng API Dialogflow của Google để lập trình dễ dàng và nếu Google ngừng cung cấp API này thì hệ thống sẽ không hoạt động.
- Khi sử dụng ngrok mới, địa chỉ URL sẽ thay đổi nên mỗi khi thay đổi, bạn cần cung cấp địa chỉ webhook mới trong phần thực hiện của Dialogflow.
- Độ trễ của hệ thống phụ thuộc vào tốc độ và độ ổn định của đường truyền

3.8 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

3.8.1 KẾT LUẬN

Nhóm thấy rằng hệ thống hoạt động tốt và đáp ứng các yêu cầu đã nêu ở phần giới hạn trong Chương 1. Đây là một hệ thống đơn giản, dễ sử dụng, sử dụng tính năng trò chuyện để điều khiển các thiết bị gia dụng qua Internet và có thể trò chuyện với người dùng.

Mẫu Thiết Kế Nhà Đơn Giản rất đơn giản và dễ tìm, giúp bạn dễ dàng hình dung được

hình ảnh ngôi nhà ngoài thực tế. Cả chương trình mô-đun và tài liệu đều rõ ràng. Hệ thống đảm bảo an toàn cho người dùng. Nhưng hệ thống vẫn còn hạn chế: phải kết nối Internet, tính bảo mật chưa cao, chatbot có thể trò chuyện đơn giản với người dùng, chưa thông minh khi hỏi người dùng những câu hỏi phức tạp, khó hiểu.

3.8.2 HƯỚNG PHÁT TRIỂN

Để tạo ra một trải nghiệm thân thiện và tiện lợi hơn cho người dùng, chúng ta có thể tích hợp chatbot vào các nền tảng nhắn tin phổ biến như Facebook Messenger, Skype, Slack, v.v. Điều này giúp người dùng dễ dàng trò chuyện với chatbot mà không cần phải chuyển đổi giữa các ứng dụng khác nhau.

Bên cạnh việc trả lời các câu hỏi cơ bản, chatbot cũng có thể trở nên thông minh hơn bằng cách tích hợp các API khác như API thời tiết, API đọc báo, API mua sắm, và đặc biệt là tích hợp API của Chat GPT - một ứng dụng chatbot nổi tiếng. Từ đó, chatbot không chỉ là một trợ lý ảo mà còn có thể cung cấp thông tin, hỗ trợ mua sắm và thậm chí là giải trí cho người dùng.

Ngoài ra, chúng ta cũng có thể tích hợp API Voice để chatbot có thể giao tiếp với người dùng bằng giọng nói thay vì tin nhắn văn bản thông thường. Điều này mang lại trải nghiệm người dùng gần gũi và tự nhiên hơn trong việc tương tác với chatbot.

Thêm vào đó, chúng ta cũng có thể mở rộng chức năng của chatbot để điều khiển nhiều thiết bị khác nhau trong ngôi nhà, tùy thuộc vào nhu cầu cụ thể của mỗi dự án.

TÀI LIỆU THAM KHẢO

- [1] Trần Thu Hà, “Giáo trình điện tử cơ bản”, NXB ĐH Quốc Gia Tp.HCM, 2013.
- [2] Hoàng Ngọc Văn, “Giáo trình thực hành điện tử công suất”, Trường ĐH Sư Phạm Kỹ Thuật Tp.HCM, 2015.
- [3] Matt Richardson and Shawn Wallace, "Getting Started with Raspberry Pi", Pushlised by O'Reilly Media, Inc., 2012.
- [4] Cao Văn Tiến và Nguyễn Văn Nghĩa, “Điều khiển thiết bị qua internet dùng raspberry pi 2 thông qua wifi”, Trường ĐH SPKT TP.HCM, 2016.
- [5] Nick Quinlan, "What's a webhook", www.sendgrid.com, 2014.
- [6] Sachin Kumar, "How to create a chatbot using Dialogflow Enterprise Edition and Dialogflow API V2", 2018.
- [7] Tài liệu trên website Dialogflow: <https://dialogflow.com/docs/getting-started>
- [8] Trang chủ của Raspberry: <https://www.raspberrypi.org>

PHỤ LỤC

Code python:

```
import json
import time
import Adafruit_DHT
from flask import Flask, request, make_response
import RPi.GPIO as GPIO

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT) # LED 1
GPIO.setup(23, GPIO.OUT) # LED 2
GPIO.setup(24, GPIO.OUT) # LED 3
GPIO.setup(25, GPIO.OUT) # LED 4
GPIO.setup(21, GPIO.OUT) # FAN

# Khai báo chân GPIO cho servo
servo_pin = 20
GPIO.setup(servo_pin, GPIO.OUT)
pwm = GPIO.PWM(servo_pin, 50)
pwm.start(0)
# Hàm để đặt góc quay của servo
def set_angle(angle):
    duty = angle / 20 + 2
    GPIO.output(servo_pin, True)
    pwm.ChangeDutyCycle(duty)
    time.sleep(1)
    GPIO.output(servo_pin, False)
    pwm.ChangeDutyCycle(0)

app = Flask(__name__)

@app.route('/on1')
def turn_on1():
```

```

GPIO.output(18, GPIO.HIGH) # Bật LED 1
# Tắt LED 2 (nếu cần)
GPIO.output(23, GPIO.LOW)
return make_response(json.dumps({"status": "on"}), 200)

@app.route('/off1')
def turn_off1():
    GPIO.output(18, GPIO.LOW) # Tắt LED 1
    return make_response(json.dumps({"status": "off"}), 200)

@app.route('/on2')
def turn_on2():
    GPIO.output(23, GPIO.HIGH) # Bật LED 2
    # Tắt LED 1 (nếu cần)
    GPIO.output(18, GPIO.LOW)
    return make_response(json.dumps({"status": "on"}), 200)

@app.route('/off2')
def turn_off2():
    GPIO.output(23, GPIO.LOW) # Tắt LED 2
    return make_response(json.dumps({"status": "off"}), 200)
# led 3
@app.route('/on3')
def turn_on3():
    GPIO.output(24, GPIO.HIGH) # Bật LED 3

    return make_response(json.dumps({"status": "on"}), 200)

@app.route('/off3')
def turn_off3():
    GPIO.output(24, GPIO.LOW) # Tắt LED 3
    return make_response(json.dumps({"status": "off"}), 200)
# led4
@app.route('/on4')
def turn_on4():
    GPIO.output(25, GPIO.HIGH) # Bật led4

```

```

    return make_response(json.dumps({"status": "on"}), 200)

@app.route('/off4')
def turn_off4():
    GPIO.output(25, GPIO.LOW) # Tắt led4
    return make_response(json.dumps({"status": "off"}), 200)
# quat
@app.route('/onfan2')
def turn_onfan2():
    GPIO.output(21, GPIO.HIGH) # Bật quạt
    #
    return make_response(json.dumps({"status": "on"}), 200)

@app.route('/off-fan2')
def turn_off_fan2():
    GPIO.output(21, GPIO.LOW) # Tắt quạt
    return make_response(json.dumps({"status": "off"}), 200)
# door
@app.route('/open_door')
def open_door():
    return make_response(json.dumps({"status": "open"}), 200)

@app.route('/close_door')
def close_door():
    return make_response(json.dumps({"status": "close"}), 200)

@app.route('/status')
def get_status():
    status1 = "on" if GPIO.input(18) else "off"
    status2 = "on" if GPIO.input(23) else "off"
    status3 = "on" if GPIO.input(24) else "off"
    status4 = "on" if GPIO.input(25) else "off"

    return make_response(json.dumps({"status1": status1, "status2": status2, "status3":
status3, "status4": status4}), 200)

```

```

# status cho fan
@app.route('/status2')
def get_status2():
    status5 = "on" if GPIO.input(21) else "off"
    return make_response(json.dumps({"status5": status5}), 200)

# Nhiet do
@app.route('/temperature')
def get_temperature():
    unit = request.args.get('unit', default='C', type=str)
    humidity, temperature = Adafruit_DHT.read_retry(Adafruit_DHT.DHT11, 27)
    if humidity is not None and temperature is not None:
        if unit.upper() == 'F':
            temperature = temperature * 1.8 + 32
            unit_display = '°F'
        else:
            unit_display = '°C'
        return make_response(json.dumps({"temperature": temperature, "unit":
unit_display}), 200)
    else:
        return make_response(json.dumps({"error": "Failed to retrieve temperature
data"}), 500)

#
@app.route('/webhook', methods=['POST'])
def webhook():
    req = request.get_json(silent=True, force=True)
    action = req.get("queryResult", {}).get("action", "")
    print("Request:")
    print(json.dumps(req, indent=4))
    res = makeWebhookResult(req)
    res = json.dumps(res, indent=4)
    print("Response:")
    print(res)
    return make_response(res)

def makeWebhookResult(req):
    speech = "" # Khởi tạo speech với một chuỗi rỗng

```



```

action = req.get("queryResult", {} ).get("action", "")

if action == "temperature":
    _, temperature = Adafruit_DHT.read_retry(Adafruit_DHT.DHT11, 27)
    if temperature is not None:
        speech = f"Temperature is {temperature} °C"
    else:
        speech = "Failed to retrieve temperature data"
elif action == "humidity":
    humidity, _ = Adafruit_DHT.read_retry(Adafruit_DHT.DHT11, 27)
    if humidity is not None:
        speech = f"Humidity is {humidity}%"
    else:
        speech = "Failed to retrieve humidity data"
elif action == "dht11":
    humidity, temperature = Adafruit_DHT.read_retry(Adafruit_DHT.DHT11, 27)
    if humidity is not None and temperature is not None:
        speech = f"Temperature is {temperature} °C and Humidity is {humidity}%"
    else:
        speech = "Failed to retrieve temperature and humidity data"
# kiểm tra neu hanh dong la open
elif action == "open-close":
    open_close_action = req.get("queryResult", {} ).get("parameters", {} ).get("open-
close", [])[0]
    if open_close_action == "open":
        # Thực hiện hành động mở cửa
        for angle in range(180, -1, -180):
            set_angle(angle)
            open_door()
            speech = "ok, open door"
    elif open_close_action == "close":
        # Thực hiện hành động đóng cửa
        close_door()
        speech = "ok, close door"
        # Điều khiển servo từ 0 độ đến 180 độ
        for angle in range(0, 181, 180):

```

```

        set_angle(angle)
    else:
        speech = "Unknown action"

    elif action == "turn_on_led1":
        GPIO.output(18, GPIO.HIGH) # Bật LED 1
        GPIO.output(23, GPIO.LOW) # Tắt LED 2
        speech = "ok, on led1"
    elif action == "turn_off_led1":
        GPIO.output(18, GPIO.LOW) # Tắt LED 1
        speech = "ok, off led1"
    # bat led 3
    elif action == "turn_on_led3":
        GPIO.output(24, GPIO.HIGH) # Bật LED 3
        speech = "ok, on led3"
    elif action == "turn_off_led3":
        GPIO.output(24, GPIO.LOW) # Tắt LED 3
        speech = "ok, off led3"
    # bat led 4
    elif action == "turn_on_led4":
        GPIO.output(25, GPIO.HIGH) # Bật LED 4
        speech = "ok, on led4"
    elif action == "turn_off_led4":
        GPIO.output(25, GPIO.LOW) # Tắt LED 4
        speech = "ok, off led4"
    # bat quat
    elif action == "turn_on_fan":
        GPIO.output(21, GPIO.HIGH) # Bật LED 4
        speech = "ok, on fan"
    elif action == "turn_off_fan":
        GPIO.output(21, GPIO.LOW) # Tắt LED 4
        speech = "ok, off fan"
    elif action == "on-off":
        led_state = req.get("queryResult", {}).get("parameters", {}).get("on-off", [])[0]
        led_name = req.get("queryResult", {}).get("parameters", {}).get("led", [])[0]
        if led_state == "on":

```

```

if led_name == "led1":
    GPIO.output(18, GPIO.HIGH)
elif led_name == "led2":
    GPIO.output(23, GPIO.HIGH)
speech = f"ok, on {led_name}"

elif led_state == "off":
    if led_name == "led1":
        GPIO.output(18, GPIO.LOW)
    elif led_name == "led2":
        GPIO.output(23, GPIO.LOW)
    # elif led_name == "led3":
    #     GPIO.output(24, GPIO.HIGH)
    speech = f"ok, off {led_name}"

if led_state == "on":
    if led_name == "led3":
        GPIO.output(24, GPIO.HIGH)
    elif led_name == "led4":
        GPIO.output(25, GPIO.HIGH)
    speech = f"ok, on {led_name}"
elif led_state == "off":
    if led_name == "led3":
        GPIO.output(24, GPIO.LOW)
    elif led_name == "led4":
        GPIO.output(25, GPIO.LOW)
    speech = f"ok, off {led_name}"
# -----
elif action == "on-off-fan":
    led_state_fan = req.get("queryResult", {}).get("parameters", {}).get("on-off-fan",
[])[0]
    led_name_fan = req.get("queryResult", {}).get("parameters", {}).get("fan", [])[0]
    if led_state_fan == "turn on the":
        if led_name_fan == "fan":
            GPIO.output(21, GPIO.HIGH)
            speech = f"ok, on {led_name_fan}"

```

```

elif led_state_fan == "turn off the":
    if led_name_fan == "fan":
        GPIO.output(21, GPIO.LOW)
#door
elif action == "open-close":
    door_state = req.get("queryResult", {}).get("parameters", {}).get("open-close",
[])[0]
    door_name = req.get("queryResult", {}).get("parameters", {}).get("door", [])[0]
    if door_state == "open the":
        if door_name == "door":
            GPIO.output(20, GPIO.HIGH)
            speech = f"ok, open {door_name}"
    elif door_state == "close the":
        if door_name == "door":
            GPIO.output(20, GPIO.LOW)
            speech = f"ok, closed {door_name}"
    else:
        speech = "Unknown action"
#
return {"fulfillmentText": speech}
#
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        payload = request.json
        user_response = payload.get("queryResult", {}).get("queryText", "")
        bot_response = payload.get("queryResult", {}).get("fulfillmentText", "")
        if user_response or bot_response:
            print("User: " + user_response)
            print("Bot: " + bot_response)
            return "Message received."
        else:
            return 'Chào mừng bạn đến với trang web của tôi!'
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)

```