EDITH (VIRTUAL ASSISTANT)

A

Mini Project Report

Submitted in partial fulfilment of the
Requirements for the award of the Degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE & ENGINEERING

By

SAI SUMAN CHITTURI
1602-18-733-097

PRANEETH KAPILA
1602-18-733-116

Department of Computer Science & Engineering

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

2020-2021

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Hyderabad-500 031

ACKNOWLEDGEMENT

We are thankful to the College Management for Encouraging us to do our Mini Project titled Meta Stream. We extend our heart-felt gratitude to our Professor, M. Sashi Kumar and our Head of the Dept. Dr. T. Adilakshmi for their Invaluable Guidance and Support. Their guidance was unforgettable and their constructive suggestions helped us in finishing the project Effectively.

TABLE OF CONTENTS

LIST OF FIGURES

EDITH (VIRTUAL ASSISTANT)

Abstract

The main aim of the work is to develop an economically effective and performance wise efficient virtual assistant using Raspberry Pi for home automation based on the concepts of Internet of Things, Speech Recognition, Natural Language Processing and Artificial Intelligence. People who are using it can give voice inputs and the device responds through voice commands by itself. It can fetch the date, time, weather, play your favourite music and fetch search results from the internet.

The Raspberry Pi processes the speech inputs online given by the user through the mic and converts it into text and executes the command. The whole project is put in action through a python script which includes online Speech to Text conversion and Text to Speech conversion codes written.

The device will respond to the user in a casual manner so that the user has a friendly experience with the device and feels it like his or her own assistant. This device makes the day-by-day processes easier.

Introduction

People today are living the busiest life. While managing lot of activities, they keep forgetting few. They find it difficult to plan and manage activities. Moreover, under such lifestyle, they are not having enough access to immediate Entertainment.

This creates a need for a Virtual Assistant. Virtual Assistants, today, are capable of planning activities, setting reminders, alarms, playing music, telling jokes and facts etc.

To Solve these problems, we built a Virtual Assistant (Edith) which is capable of planning activities, setting reminders, alarms, playing music, telling jokes and facts etc.

This work is constructed based on the basic concepts on Internet of things (IoT) and Natural Language Processing (NLP). This system is designed to provide a user-friendly experience as well as an easier interface so that anyone can use this effortlessly. This project has been implemented with the help of Raspberry Pi 3 Model B, Google Assistant API, News API and Web scraping tools.

Natural Language Processing is simply a bridge-way that reduces the distance between human communication and machine communication. The main objective of NLP is to make the machines understand the natural human language so that the usage becomes very much comfortable. In technical terms, NLP is the algorithm which analyses and synthesizes human speech. This algorithm is based on artificial intelligence and computational linguistics.

SOFTWARE REQUIREMENTS SPECIFICATION

The software is designed to be light-weighted so that it doesn't be a burden on the machine running it. This system is being build keeping in mind the generally available hardware and software compatibility. Here are the minimum hardware and software requirements for Edith - virtual assistant.

Hardware Requirements:

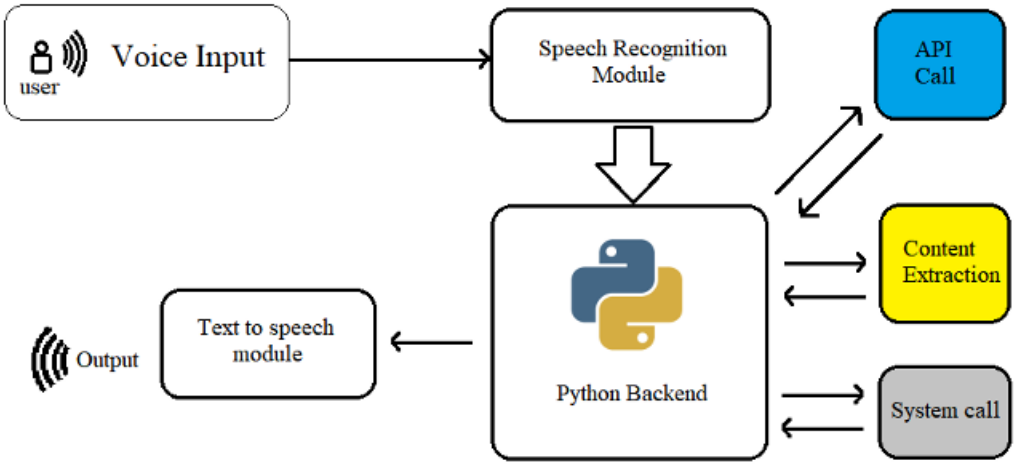
- Raspberry Pi 3 B+
- USB Microphone
- USB Speaker
- Interfacing cables – HDMI, Ethernet Cable

Software Requirements:

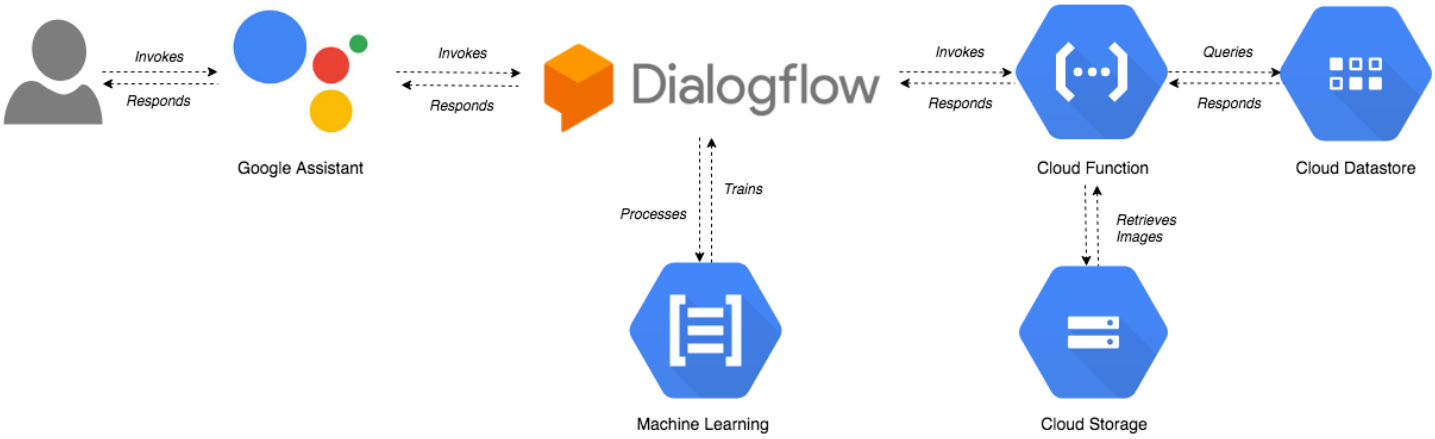
- Raspbian OS
- Python 3
- Google Voice Assistant API must be configured in the machine
- Chromium based Web Browsers
- Python3 Modules Required
    - GRPC Client
    - Google OAuth lib tool
    - gTTS – Google Text to speech
    - Play sound
    - Pafy - YouTube Scraper
    - Speech Recognition
    - Python3 Virtual Environment
    - Port Audio 19
    - PyAudio

SYSTEM DESIGN

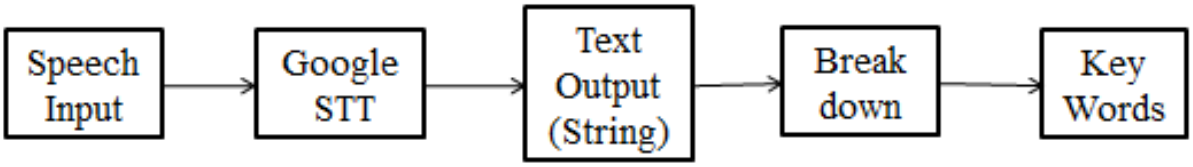
1. Overall Architecture



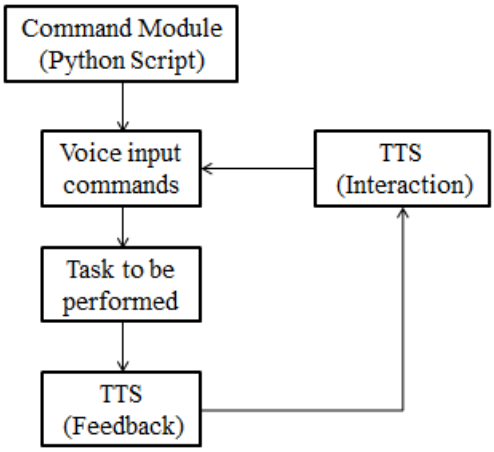
2. Google Assistant Backend Services



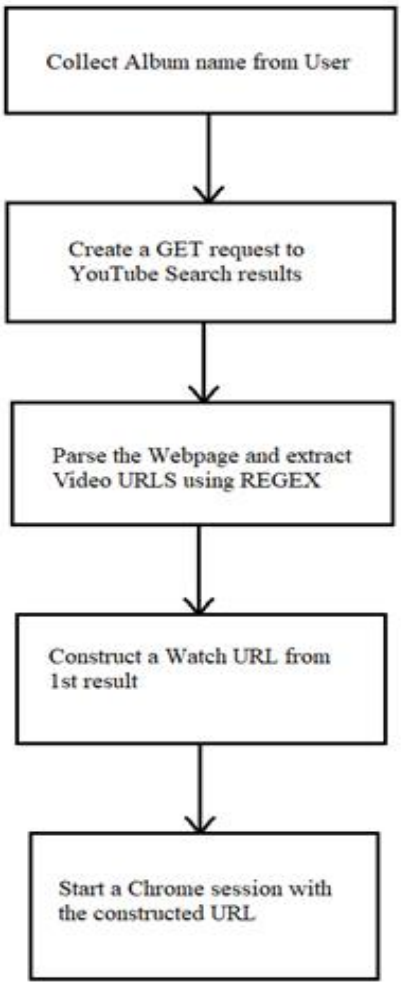
3. Speech to text Module



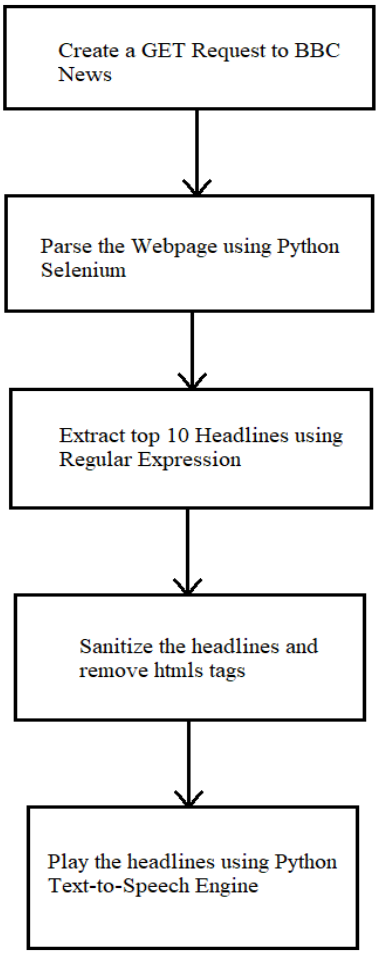
4. Text to Speech Module



5. Music Playback Module



6. News Module



PSEUDO ALGORITHM / IMPLEMENTATION / CODE

Code:

1. Assitant.py

```
"""Sample that implements a gRPC client for the Google Assistant API."""

import concurrent.futures
import json
import logging
import os
import os.path
import pathlib2 as pathlib
import sys
import time
import uuid

import click
import grpc
import google.auth.transport.grpc
import google.auth.transport.requests
import google.oauth2.credentials

from google.assistant.embedded.v1alpha2 import (
    embedded_assistant_pb2,
    embedded_assistant_pb2_grpc
)
from tenacity import retry, stop_after_attempt, retry_if_exception

try:
    from . import (
        assistant_helpers,
        audio_helpers,
        browser_helpers,
        device_helpers
    )
except (SystemError, ImportError):
    import assistant_helpers
    import audio_helpers
    import browser_helpers
    import device_helpers

from Play_Music import play_song
from News import NewsFromBBC

ASSISTANT_API_ENDPOINT = 'embeddedassistant.googleapis.com'
END_OF_UTTERANCE = embedded_assistant_pb2.AssistResponse.END_OF_UTTERANCE
DIALOG_FOLLOW_ON = embedded_assistant_pb2.DialogStateOut.DIALOG_FOLLOW_ON
CLOSE_MICROPHONE = embedded_assistant_pb2.DialogStateOut.CLOSE_MICROPHONE
PLAYING = embedded_assistant_pb2.ScreenOutConfig.PLAYING
DEFAULT_GRPC_DEADLINE = 60 * 3 + 5


class SampleAssistant(object):
    """Sample Assistant that supports conversations and device actions.
    Args:
      device_model_id: identifier of the device model.
      device_id: identifier of the registered device instance.
      conversation_stream(ConversationStream): audio stream
```

```
    for recording query and playing back assistant answer.
  channel: authorized gRPC channel for connection to the
    Google Assistant API.
  deadline_sec: gRPC deadline in seconds for Google Assistant API call.
  device_handler: callback for device actions.
"""

def __init__(self, language_code, device_model_id, device_id,
             conversation_stream, display,
             channel, deadline_sec, device_handler):
    self.language_code = language_code
    self.device_model_id = device_model_id
    self.device_id = device_id
    self.conversation_stream = conversation_stream
    self.display = display

    # Opaque blob provided in AssistResponse that,
    # when provided in a follow-up AssistRequest,
    # gives the Assistant a context marker within the current state
    # of the multi-Assist()-RPC "conversation".
    # This value, along with MicrophoneMode, supports a more natural
    # "conversation" with the Assistant.
    self.conversation_state = None
    # Force reset of first conversation.
    self.is_new_conversation = True

    # Create Google Assistant API gRPC client.
    self.assistant = embedded_assistant_pb2_grpc.EmbeddedAssistantStub(
        channel
    )
    self.deadline = deadline_sec

    self.device_handler = device_handler

def __enter__(self):
    return self

def __exit__(self, etype, e, traceback):
    if e:
        return False
    self.conversation_stream.close()

def is_grpc_error_unavailable(e):
    is_grpc_error = isinstance(e, grpc.RpcError)
    if is_grpc_error and (e.code() == grpc.StatusCode.UNAVAILABLE):
        logging.error('grpc unavailable error: %s', e)
        return True
    return False

@retry(reraise=True, stop=stop_after_attempt(3),
       retry=retry_if_exception(is_grpc_error_unavailable))
def assist(self):
    """Send a voice request to the Assistant and playback the response.
    Returns: True if conversation should continue.
    """
    continue_conversation = False
    device_actions_futures = []

    self.conversation_stream.start_recording()
    logging.info('Recording audio request.')
```

```
        def iter_log_assist_requests():
            for c in self.gen_assist_requests():
                assistant_helpers.log_assist_request_without_audio(c)
                yield c
            logging.debug('Reached end of AssistRequest iteration.')

        # This generator yields AssistResponse proto messages
        # received from the gRPC Google Assistant API.
        user_transcripts = ['0']
        for resp in self.assistant.Assist(iter_log_assist_requests(),
                                          self.deadline):
            assistant_helpers.log_assist_response_without_audio(resp)
            if resp.event_type == END_OF_UTTERANCE:
                logging.info('End of audio request detected.')
                logging.info('Stopping recording.')
                self.conversation_stream.stop_recording()

            if resp.speech_results:
                user_transcripts[0] = ' '.join(r.transcript
                                               for r in resp.speech_results)
                logging.info('Transcript of user request: "%s".',
                             ' '.join(r.transcript
                                               for r in resp.speech_results))
            if len(resp.audio_out.audio_data) > 0:
                if not self.conversation_stream.playing:
                    self.conversation_stream.stop_recording()
                    self.conversation_stream.start_playback()
                    logging.info('Playing assistant response.')
                    if "news" in user_transcripts[0]:
                        NewsFromBBC()
                        exit(0)
                    elif "play" in user_transcripts[0] and "game" not in
user_transcripts[0]:
                        song_name = user_transcripts[0][5:]
                        song_name = song_name.replace(" ", "+")
                        play_song(song_name)
                    else:

self.conversation_stream.write(resp.audio_out.audio_data)
                self.conversation_stream.write(resp.audio_out.audio_data)
            if resp.dialog_state_out.conversation_state:
                conversation_state =
resp.dialog_state_out.conversation_state
                logging.debug('Updating conversation state.')
                self.conversation_state = conversation_state
            if resp.dialog_state_out.volume_percentage != 0:
                volume_percentage = resp.dialog_state_out.volume_percentage
                logging.info('Setting volume to %s%%', volume_percentage)
                self.conversation_stream.volume_percentage =
volume_percentage
            if resp.dialog_state_out.microphone_mode == DIALOG_FOLLOW_ON:
                continue_conversation = True
                logging.info('Expecting follow-on query from user.')
            elif resp.dialog_state_out.microphone_mode == CLOSE_MICROPHONE:
                continue_conversation = False
            if resp.device_action.device_request_json:
                device_request = json.loads(
                    resp.device_action.device_request_json
                )
                fs = self.device_handler(device_request)
                if fs:
```

```
                    device_actions_futures.extend(fs)
            if self.display and resp.screen_out.data:
                system_browser = browser_helpers.system_browser
                system_browser.display(resp.screen_out.data)

        if len(device_actions_futures):
            logging.info('Waiting for device executions to complete.')
            concurrent.futures.wait(device_actions_futures)

        logging.info('Finished playing assistant response.')
        self.conversation_stream.stop_playback()
        return continue_conversation

    def gen_assist_requests(self):
        """Yields: AssistRequest messages to send to the API."""

        config = embedded_assistant_pb2.AssistConfig(
            audio_in_config=embedded_assistant_pb2.AudioInConfig(
                encoding='LINEAR16',
                sample_rate_hertz=self.conversation_stream.sample_rate,
            ),
            audio_out_config=embedded_assistant_pb2.AudioOutConfig(
                encoding='LINEAR16',
                sample_rate_hertz=self.conversation_stream.sample_rate,

volume_percentage=self.conversation_stream.volume_percentage,
            ),
            dialog_state_in=embedded_assistant_pb2.DialogStateIn(
                language_code=self.language_code,
                conversation_state=self.conversation_state,
                is_new_conversation=self.is_new_conversation,
            ),
            device_config=embedded_assistant_pb2.DeviceConfig(
                device_id=self.device_id,
                device_model_id=self.device_model_id,
            )
        )
        if self.display:
            config.screen_out_config.screen_mode = PLAYING
        # Continue current conversation with later requests.
        self.is_new_conversation = False
        # The first AssistRequest must contain the AssistConfig
        # and no audio data.
        yield embedded_assistant_pb2.AssistRequest(config=config)
        for data in self.conversation_stream:
            # Subsequent requests need audio data, but not config.
            yield embedded_assistant_pb2.AssistRequest(audio_in=data)


@click.command()
@click.option('--api-endpoint', default=ASSISTANT_API_ENDPOINT,
              metavar='<api endpoint>', show_default=True,
              help='Address of Google Assistant API service.')
@click.option('--credentials',
              metavar='<credentials>', show_default=True,
              default=os.path.join(click.get_app_dir('google-oauthlib-
tool'),
                                   'credentials.json'),
              help='Path to read OAuth2 credentials.')
@click.option('--project-id',
              metavar='<project id>',
```

```
              help=('Google Developer Project ID used for registration '
                    'if --device-id is not specified'))
@click.option('--device-model-id',
              metavar='<device model id>',
              help=(('Unique device model identifier, '
                     'if not specifed, it is read from --device-config')))
@click.option('--device-id',
              metavar='<device id>',
              help=(('Unique registered device instance identifier, '
                     'if not specified, it is read from --device-config, '
                     'if no device_config found: a new device is registered
'
                     'using a unique id and a new device config is
saved')))
@click.option('--device-config', show_default=True,
              metavar='<device config>',
              default=os.path.join(
                  click.get_app_dir('googlesamples-assistant'),
                  'device_config.json'),
              help='Path to save and restore the device configuration')
@click.option('--lang', show_default=True,
              metavar='<language code>',
              default='en-US',
              help='Language code of the Assistant')
@click.option('--display', is_flag=True, default=False,
              help='Enable visual display of Assistant responses in HTML.')
@click.option('--verbose', '-v', is_flag=True, default=False,
              help='Verbose logging.')
@click.option('--input-audio-file', '-i',
              metavar='<input file>',
              help='Path to input audio file. '
              'If missing, uses audio capture')
@click.option('--output-audio-file', '-o',
              metavar='<output file>',
              help='Path to output audio file. '
              'If missing, uses audio playback')
@click.option('--audio-sample-rate',
              default=audio_helpers.DEFAULT_AUDIO_SAMPLE_RATE,
              metavar='<audio sample rate>', show_default=True,
              help='Audio sample rate in hertz.')
@click.option('--audio-sample-width',
              default=audio_helpers.DEFAULT_AUDIO_SAMPLE_WIDTH,
              metavar='<audio sample width>', show_default=True,
              help='Audio sample width in bytes.')
@click.option('--audio-iter-size',
              default=audio_helpers.DEFAULT_AUDIO_ITER_SIZE,
              metavar='<audio iter size>', show_default=True,
              help='Size of each read during audio stream iteration in
bytes.')
@click.option('--audio-block-size',
              default=audio_helpers.DEFAULT_AUDIO_DEVICE_BLOCK_SIZE,
              metavar='<audio block size>', show_default=True,
              help=('Block size in bytes for each audio device '
                    'read and write operation.'))
@click.option('--audio-flush-size',
              default=audio_helpers.DEFAULT_AUDIO_DEVICE_FLUSH_SIZE,
              metavar='<audio flush size>', show_default=True,
              help=('Size of silence data in bytes written '
                    'during flush operation'))
@click.option('--grpc-deadline', default=DEFAULT_GRPC_DEADLINE,
              metavar='<grpc deadline>', show_default=True,
```

```
              help='gRPC deadline in seconds')
@click.option('--once', default=False, is_flag=True,
              help='Force termination after a single conversation.')
def main(api_endpoint, credentials, project_id,
         device_model_id, device_id, device_config,
         lang, display, verbose,
         input_audio_file, output_audio_file,
         audio_sample_rate, audio_sample_width,
         audio_iter_size, audio_block_size, audio_flush_size,
         grpc_deadline, once, *args, **kwargs):
    """Samples for the Google Assistant API.
    Examples:
      Run the sample with microphone input and speaker output:
        $ python -m googlesamples.assistant
      Run the sample with file input and speaker output:
        $ python -m googlesamples.assistant -i <input file>
      Run the sample with file input and output:
        $ python -m googlesamples.assistant -i <input file> -o <output
file>
    """
    # Setup logging.
    logging.basicConfig(level=logging.DEBUG if verbose else logging.INFO)

    # Load OAuth 2.0 credentials.
    try:
        with open(credentials, 'r') as f:
            credentials = google.oauth2.credentials.Credentials(token=None,

**json.load(f))
            http_request = google.auth.transport.requests.Request()
            credentials.refresh(http_request)
    except Exception as e:
        logging.error('Error loading credentials: %s', e)
        logging.error('Run google-oauthlib-tool to initialize '
                      'new OAuth 2.0 credentials.')
        sys.exit(-1)

    # Create an authorized gRPC channel.
    grpc_channel = google.auth.transport.grpc.secure_authorized_channel(
        credentials, http_request, api_endpoint)
    logging.info('Connecting to %s', api_endpoint)

    # Configure audio source and sink.
    audio_device = None
    if input_audio_file:
        audio_source = audio_helpers.WaveSource(
            open(input_audio_file, 'rb'),
            sample_rate=audio_sample_rate,
            sample_width=audio_sample_width
        )
    else:
        audio_source = audio_device = (
            audio_device or audio_helpers.SoundDeviceStream(
                sample_rate=audio_sample_rate,
                sample_width=audio_sample_width,
                block_size=audio_block_size,
                flush_size=audio_flush_size
            )
        )
    if output_audio_file:
        audio_sink = audio_helpers.WaveSink(
```

```
            open(output_audio_file, 'wb'),
            sample_rate=audio_sample_rate,
            sample_width=audio_sample_width
        )
    else:
        audio_sink = audio_device = (
            audio_device or audio_helpers.SoundDeviceStream(
                sample_rate=audio_sample_rate,
                sample_width=audio_sample_width,
                block_size=audio_block_size,
                flush_size=audio_flush_size
            )
        )
    # Create conversation stream with the given audio source and sink.
    conversation_stream = audio_helpers.ConversationStream(
        source=audio_source,
        sink=audio_sink,
        iter_size=audio_iter_size,
        sample_width=audio_sample_width,
    )

    if not device_id or not device_model_id:
        try:
            with open(device_config) as f:
                device = json.load(f)
                device_id = device['id']
                device_model_id = device['model_id']
                logging.info("Using device model %s and device id %s",
                             device_model_id,
                             device_id)
        except Exception as e:
            logging.warning('Device config not found: %s' % e)
            logging.info('Registering device')
            if not device_model_id:
                logging.error('Option --device-model-id required '
                              'when registering a device instance.')
                sys.exit(-1)
            if not project_id:
                logging.error('Option --project-id required '
                              'when registering a device instance.')
                sys.exit(-1)
            device_base_url = (
                'https://%s/v1alpha2/projects/%s/devices' % (api_endpoint,
                                                             project_id)
            )
            device_id = str(uuid.uuid1())
            payload = {
                'id': device_id,
                'model_id': device_model_id,
                'client_type': 'SDK_SERVICE'
            }
            session = google.auth.transport.requests.AuthorizedSession(
                credentials
            )
            r = session.post(device_base_url, data=json.dumps(payload))
            if r.status_code != 200:
                logging.error('Failed to register device: %s', r.text)
                sys.exit(-1)
            logging.info('Device registered: %s', device_id)

pathlib.Path(os.path.dirname(device_config)).mkdir(exist_ok=True)
```

```
            with open(device_config, 'w') as f:
                json.dump(payload, f)

    device_handler = device_helpers.DeviceRequestHandler(device_id)

    @device_handler.command('action.devices.commands.OnOff')
    def onoff(on):
        if on:
            logging.info('Turning device on')
        else:
            logging.info('Turning device off')

    @device_handler.command('com.example.commands.BlinkLight')
    def blink(speed, number):
        logging.info('Blinking device %s times.' % number)
        delay = 1
        if speed == "SLOWLY":
            delay = 2
        elif speed == "QUICKLY":
            delay = 0.5
        for i in range(int(number)):
            logging.info('Device is blinking.')
            time.sleep(delay)

    with SampleAssistant(lang, device_model_id, device_id,
                         conversation_stream, display,
                         grpc_channel, grpc_deadline,
                         device_handler) as assistant:
        # If file arguments are supplied:
        # exit after the first turn of the conversation.
        if input_audio_file or output_audio_file:
            assistant.assist()
            return

        # If no file arguments supplied:
        # keep recording voice requests using the microphone
        # and playing back assistant response using the speaker.
        # When the once flag is set, don't wait for a trigger. Otherwise,
wait.
        wait_for_user_trigger = not once
        while True:
            if wait_for_user_trigger:
                click.pause(info='Press Enter to send a new request...')
            continue_conversation = assistant.assist()
            # wait for user trigger if there is no follow-up turn in
            # the conversation.
            wait_for_user_trigger = not continue_conversation

            # If we only want one conversation, break.
            if once and (not continue_conversation):
                break


if __name__ == '__main__':
    main()
```

2. Music_Playback.py

```
# This program plays the music and videos from YouTube

import os                # For system calls
import signal            # For SIGTERM value
import pafy              # Used to collect duration of video
import re                # Used to parse the webpage of search results
import urllib.request    # To create a request to URL
import urllib.parse      # To encode URLs
import subprocess        # To Play song on Chrome
import time              # For sleep

def play_song(msg):

    # song name from user
    song = urllib.parse.urlencode({"search_query" : msg})

    # fetch the ?v=query_string
    result = urllib.request.urlopen("http://www.youtube.com/results?" +
song)

    # make the url of the first result song
    search_results = re.findall(r'\/watch\?v=(.{11})',
result.read().decode())

    # make the final url of song; selects the very first result from
youtube result
    url = "https://www.youtube.com/watch?v=" + search_results[0]

    # Extract the length of video from Metadata
    video = pafy.new(url)
    length = video.length

    # Start a chrome session with the Video URL
    pid = subprocess.Popen("google-chrome " + url, shell = True).pid

    # Wait until the video finishes
    time.sleep(length + 5)

    # Terminate the Chrome session
    os.killpg(os.getpgid(pid), signal.SIGTERM)
```

3. News.py

```
# Reads out News Headlines from BBC
import requests
import gtts
import playsound
import os

def NewsFromBBC():

      # BBC news api
      # Following query parameters are used
      # source, sortBy and apiKey

      query_params = {
      "source": "bbc-news",
      "sortBy": "top",
      "apiKey": "<Your API Key>"
      }

      # URL of News API
      main_url = " https://newsapi.org/v1/articles"

      # fetching data in json format
      res = requests.get(main_url, params=query_params)
      open_bbc_page = res.json()

      # getting all articles in a string article
      article = open_bbc_page["articles"]

      # empty list which will
      # contain all trending news
      results = []

      for ar in article:
            results.append(ar["title"])

      for i in range(len(results)):

            # Convert Text to Speech using Google Text-to-speech Engine
            myobj = gtts.gTTS(text = results[i], lang = 'en')

            # Save it in Mp3 Format
            myobj.save("News.mp3")

            # Play the music file
            playsound.playsound("News.mp3")

            # Remove the music file after reading
            os.system("rm News.mp3")
```

SCREENSHOTS

1. Weather Updates

Raspberry Pi [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

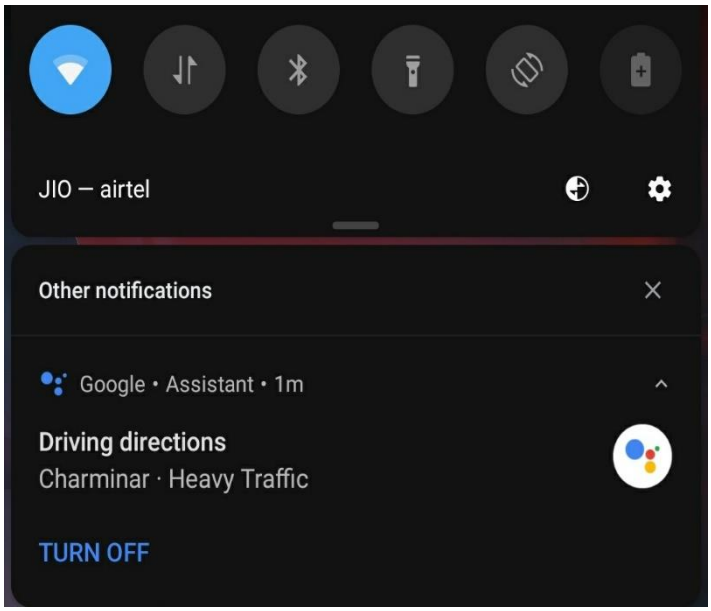
```
(env) pi@raspberrypi:~ $ googlesamples-assistant-pushtotalk --project-id edith-b5fb1 --device-model-id edith-b5fb1-edith-z7xol7
INFO:root:Connecting to embeddedassistant.googleapis.com
INFO:root:Using device model edith-b5fb1-edith-z7xol7 and device id 275563c8-c10d-11eb-b5de-0800274bb4e8
Press Enter to send a new request...
INFO:root:Recording audio request.
INFO:root:Transcript of user request: "how".
INFO:root:Transcript of user request: "how is".
INFO:root:Transcript of user request: "how is the".
INFO:root:Transcript of user request: "how is the way".
INFO:root:Transcript of user request: "how is the weather".
INFO:root:Transcript of user request: "how  is the weather".
INFO:root:Transcript of user request: "how is  the weather".
INFO:root:End of audio request detected.
INFO:root:Stopping recording.
INFO:root:Transcript of user request: "how is the weather".
INFO:root:Playing assistant response.
INFO:root:Finished playing assistant response.
Press Enter to send a new request...
INFO:root:Recording audio request.
INFO:root:Transcript of user request: "how".
INFO:root:Transcript of user request: "how is".
INFO:root:Transcript of user request: "how is the".
INFO:root:Transcript of user request: "how is the way".
INFO:root:Transcript of user request: "how is the weather".
INFO:root:Transcript of user request: "how  is the weather".
INFO:root:Transcript of user request: "how is  the weather".
INFO:root:Transcript of user request: "how is  the weather in".
INFO:root:Transcript of user request: "how is the  weather in".
INFO:root:Transcript of user request: "how is the  weather in New".
INFO:root:Transcript of user request: "how is the weather  in New York".
INFO:root:Transcript of user request: "how is the weather in  New York".
INFO:root:End of audio request detected.
INFO:root:Stopping recording.
INFO:root:Transcript of user request: "how is the weather in New  York".
INFO:root:Transcript of user request: "how is the weather in New York".
INFO:root:Playing assistant response.
INFO:root:Finished playing assistant response.
Press Enter to send a new request...
```

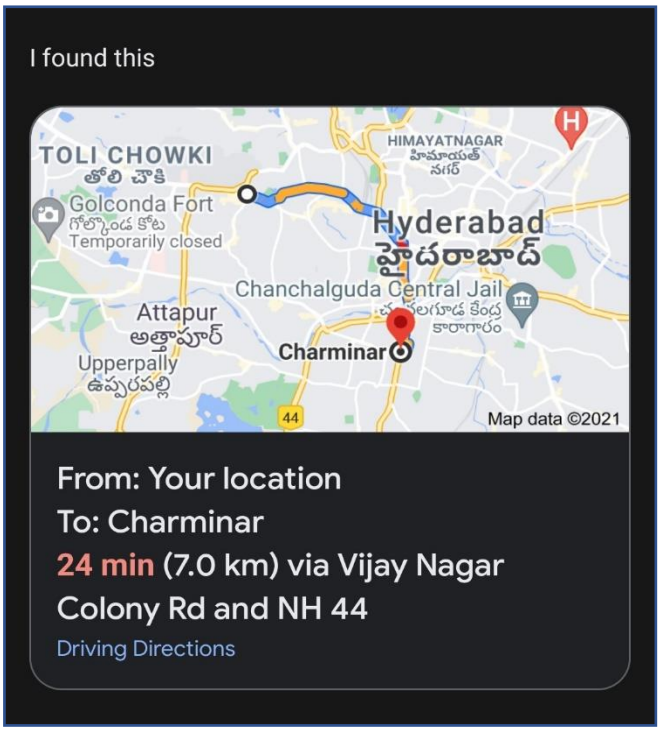
2. Traffic Updates

```
Press Enter to send a new request...
INFO:root:Recording audio request.
INFO:root:Transcript of user request: "how".
INFO:root:Transcript of user request: "how is".
INFO:root:Transcript of user request: "how is the".
INFO:root:Transcript of user request: "how is the way".
INFO:root:Transcript of user request: "how is the weather".
INFO:root:Transcript of user request: "how  is the weather".
INFO:root:Transcript of user request: "how is  the weather".
INFO:root:End of audio request detected.
INFO:root:Stopping recording.
INFO:root:Transcript of user request: "how is the weather".
INFO:root:Playing assistant response.
INFO:root:Finished playing assistant response.
Press Enter to send a new request...
INFO:root:Recording audio request.
INFO:root:Transcript of user request: "how".
INFO:root:Transcript of user request: "how is".
INFO:root:Transcript of user request: "how is the".
INFO:root:Transcript of user request: "how is the way".
INFO:root:Transcript of user request: "how is the weather".
INFO:root:Transcript of user request: "how  is the weather".
INFO:root:Transcript of user request: "how is  the weather".
INFO:root:Transcript of user request: "how is  the weather in".
INFO:root:Transcript of user request: "how is the  weather in".
INFO:root:Transcript of user request: "how is the  weather in New".
INFO:root:Transcript of user request: "how is the weather  in New York".
INFO:root:Transcript of user request: "how is the weather in  New York".
INFO:root:End of audio request detected.
INFO:root:Stopping recording.
INFO:root:Transcript of user request: "how is the weather in New  York".
INFO:root:Transcript of user request: "how is the weather in New York".
INFO:root:Playing assistant response.
INFO:root:Finished playing assistant response.
Press Enter to send a new request...
INFO:root:Recording audio request.
INFO:root:Transcript of user request: "how".
INFO:root:Transcript of user request: "how is".
INFO:root:Transcript of user request: "how is the".
INFO:root:Transcript of user request: "how  is the traffic".
INFO:root:Transcript of user request: "how is  the traffic".
INFO:root:Transcript of user request: "how is the  traffic".
INFO:root:End of audio request detected.
INFO:root:Stopping recording.
INFO:root:Transcript of user request: "how is the traffic".
INFO:root:Playing assistant response.
INFO:root:Finished playing assistant response.
Press Enter to send a new request...
```

3. Directions to a place

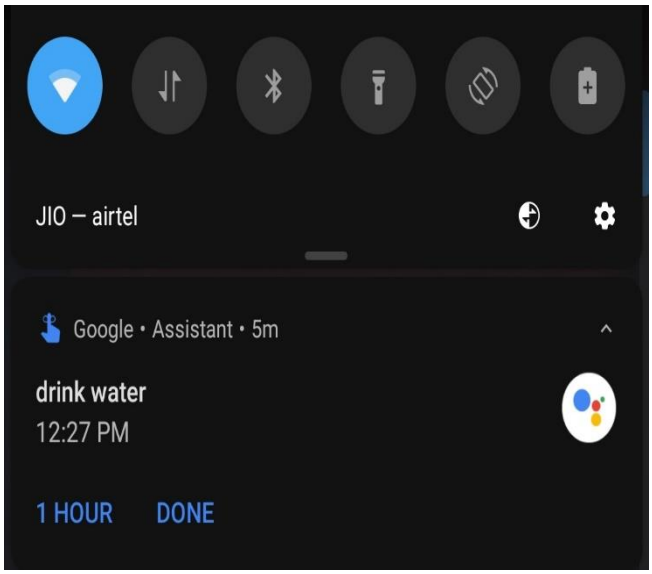
```
Press Enter to send a new request...
INFO:root:Recording audio request.
INFO:root:Transcript of user request: "how".
INFO:root:Transcript of user request: "how is".
INFO:root:Transcript of user request: "how is the".
INFO:root:Transcript of user request: "how  is the traffic".
INFO:root:Transcript of user request: "how is  the traffic".
INFO:root:Transcript of user request: "how is the  traffic".
INFO:root:End of audio request detected.
INFO:root:Stopping recording.
INFO:root:Transcript of user request: "how is the traffic".
INFO:root:Playing assistant response.
INFO:root:Finished playing assistant response.
Press Enter to send a new request...
INFO:root:Recording audio request.
INFO:root:Transcript of user request: "give".
INFO:root:Transcript of user request: "give me".
INFO:root:Transcript of user request: "give  me directions".
INFO:root:Transcript of user request: "give me  directions".
INFO:root:Transcript of user request: "give me  direction".
INFO:root:Transcript of user request: "give me  directions".
INFO:root:Transcript of user request: "give me  directions to".
INFO:root:Transcript of user request: "give me  directions to 6".
INFO:root:Transcript of user request: "give me directions  to 6".
INFO:root:Transcript of user request: "give me directions  to 4".
INFO:root:Transcript of user request: "give me directions  to Charmi".
INFO:root:Transcript of user request: "give me directions to  Charminar".
INFO:root:End of audio request detected.
INFO:root:Stopping recording.
INFO:root:Transcript of user request: "give me directions to Charminar".
INFO:root:Playing assistant response.
INFO:root:Finished playing assistant response.
Press Enter to send a new request...
```

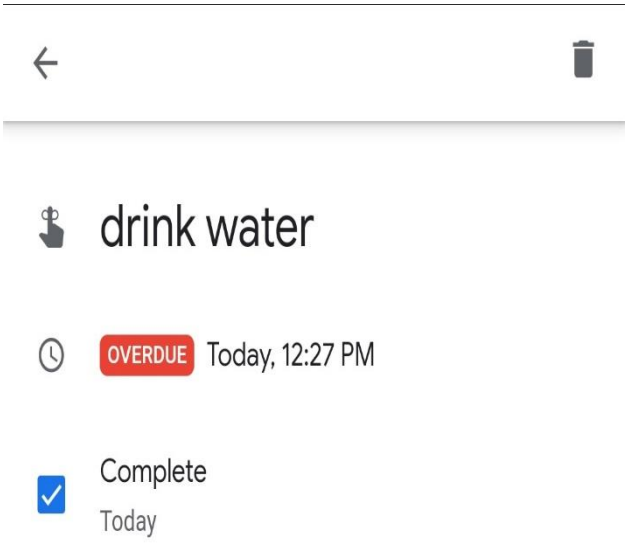




4. Setting Reminder

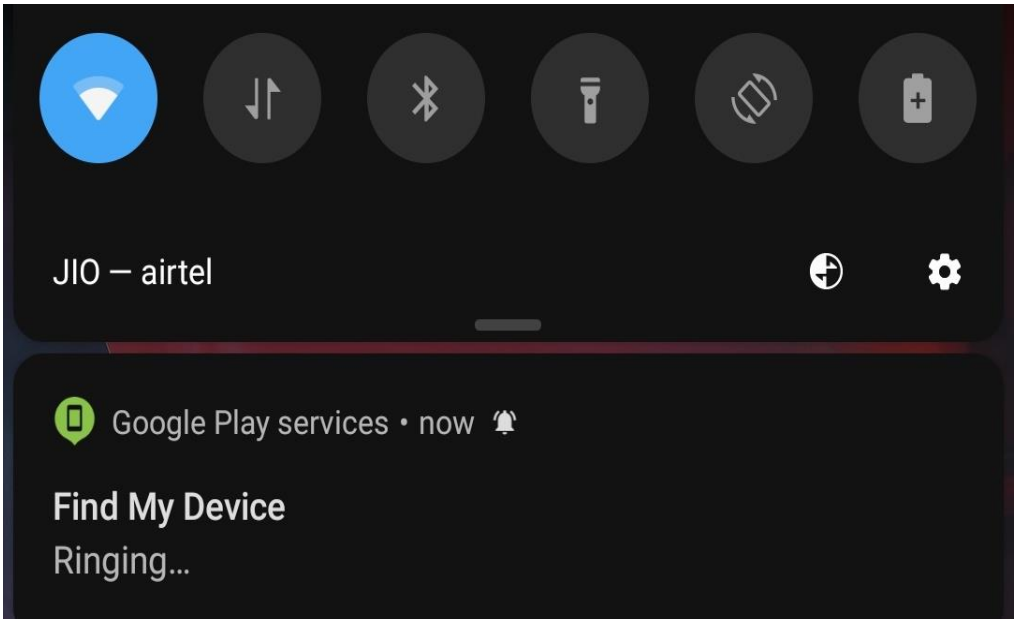
```
Press Enter to send a new request...
INFO:root:Recording audio request.
INFO:root:Transcript of user request: "set".
INFO:root:Transcript of user request: "set a".
INFO:root:Transcript of user request: "set a reminder".
INFO:root:Transcript of user request: "set  a reminder".
INFO:root:End of audio request detected.
INFO:root:Stopping recording.
INFO:root:Transcript of user request: "set a  reminder".
INFO:root:Transcript of user request: "set a reminder".
INFO:root:Playing assistant response.
INFO:root:Expecting follow-on query from user.
INFO:root:Finished playing assistant response.
INFO:root:Recording audio request.
INFO:root:Transcript of user request: "draw".
INFO:root:Transcript of user request: "dream".
INFO:root:Transcript of user request: "drink".
INFO:root:Transcript of user request: "drinkware".
INFO:root:Transcript of user request: "drink war".
INFO:root:Transcript of user request: "drink water".
INFO:root:End of audio request detected.
INFO:root:Stopping recording.
INFO:root:Transcript of user request: "drink water".
INFO:root:Expecting follow-on query from user.
INFO:root:Playing assistant response.
INFO:root:Finished playing assistant response.
INFO:root:Recording audio request.
INFO:root:Transcript of user request: "in".
INFO:root:Transcript of user request: "infinite".
INFO:root:Transcript of user request: "in 5".
INFO:root:Transcript of user request: "in 5 mein".
INFO:root:Transcript of user request: "in 5 min".
INFO:root:Transcript of user request: "in 5 minutes".
INFO:root:Transcript of user request: "in 5  minutes".
INFO:root:End of audio request detected.
INFO:root:Stopping recording.
INFO:root:Transcript of user request: "in 5 minutes".
INFO:root:Playing assistant response.
INFO:root:Finished playing assistant response.
Press Enter to send a new request...
```





5. Ring my phone feature

```
INFO:root:Recording audio request.
INFO:root:Transcript of user request: "draw".
INFO:root:Transcript of user request: "dream".
INFO:root:Transcript of user request: "drink".
INFO:root:Transcript of user request: "drinkware".
INFO:root:Transcript of user request: "drink war".
INFO:root:Transcript of user request: "drink water".
INFO:root:End of audio request detected.
INFO:root:Stopping recording.
INFO:root:Transcript of user request: "drink water".
INFO:root:Expecting follow-on query from user.
INFO:root:Playing assistant response.
INFO:root:Finished playing assistant response.
INFO:root:Recording audio request.
INFO:root:Transcript of user request: "in".
INFO:root:Transcript of user request: "infinite".
INFO:root:Transcript of user request: "in 5".
INFO:root:Transcript of user request: "in 5 mein".
INFO:root:Transcript of user request: "in 5 min".
INFO:root:Transcript of user request: "in 5 minutes".
INFO:root:Transcript of user request: "in 5  minutes".
INFO:root:End of audio request detected.
INFO:root:Stopping recording.
INFO:root:Transcript of user request: "in 5 minutes".
INFO:root:Playing assistant response.
INFO:root:Finished playing assistant response.
Press Enter to send a new request...
INFO:root:Recording audio request.
INFO:root:Transcript of user request: "ring".
INFO:root:Transcript of user request: "Ring My".
INFO:root:Transcript of user request: "ring my phone".
INFO:root:Transcript of user request: "ring  my phone".
INFO:root:End of audio request detected.
INFO:root:Stopping recording.
INFO:root:Transcript of user request: "ring my phone".
INFO:root:Playing assistant response.
INFO:root:Finished playing assistant response.
Press Enter to send a new request...
```



6. Jokes and Facts

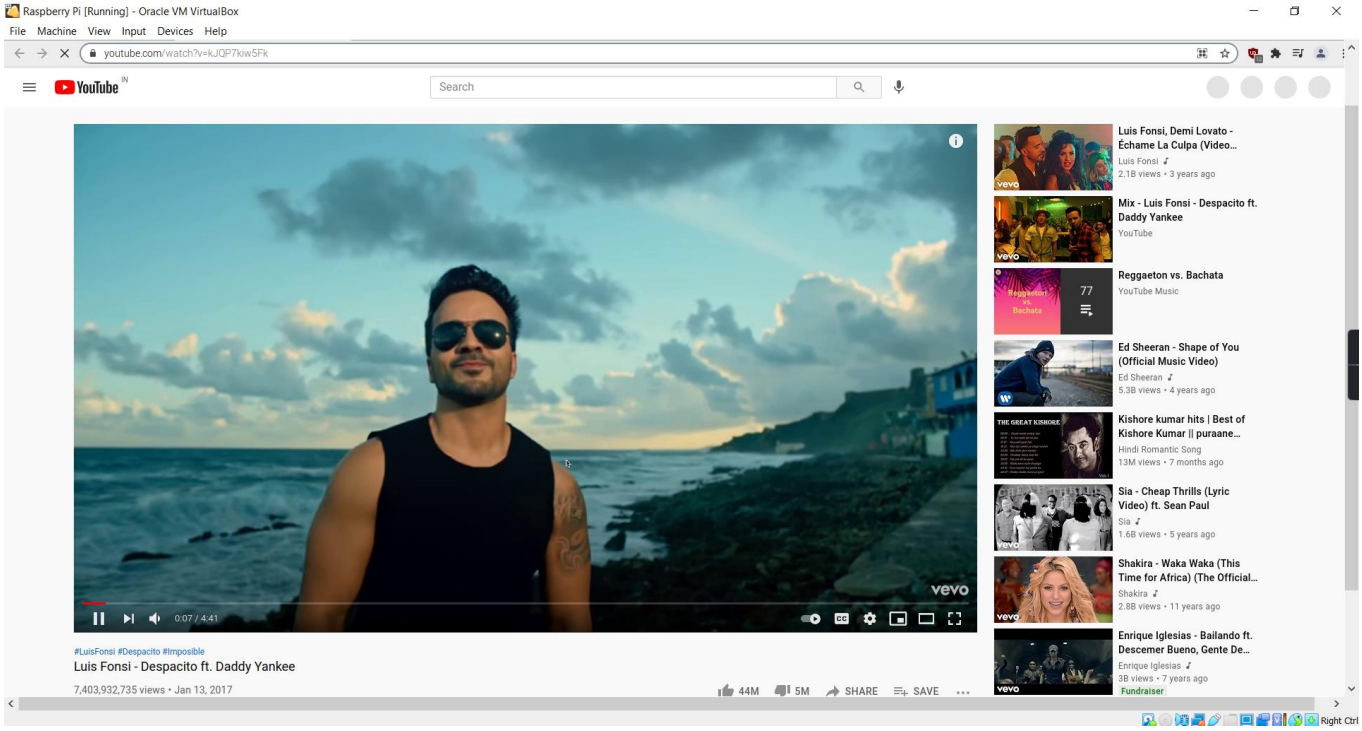
```
INFO:root:Playing assistant response.
INFO:root:Finished playing assistant response.
Press Enter to send a new request...
INFO:root:Recording audio request.
INFO:root:Transcript of user request: "tell".
INFO:root:Transcript of user request: "tell me".
INFO:root:Transcript of user request: "tell me a".
INFO:root:Transcript of user request: "tell me a joke".
INFO:root:Transcript of user request: "tell  me a joke".
INFO:root:Transcript of user request: "tell me  a joke".
INFO:root:End of audio request detected.
INFO:root:Stopping recording.
INFO:root:Transcript of user request: "tell me a  joke".
INFO:root:Transcript of user request: "tell me a joke".
INFO:root:Playing assistant response.
INFO:root:Finished playing assistant response.
Press Enter to send a new request...
INFO:root:Recording audio request.
INFO:root:Transcript of user request: "tell".
INFO:root:Transcript of user request: "tell me".
INFO:root:Transcript of user request: "tell me an".
INFO:root:Transcript of user request: "tell me an a".
INFO:root:Transcript of user request: "tell me an in".
INFO:root:Transcript of user request: "tell  me an".
INFO:root:Transcript of user request: "tell me  an interest".
INFO:root:Transcript of user request: "tell me  an interesting".
INFO:root:Transcript of user request: "tell me an  interesting".
INFO:root:Transcript of user request: "tell me an  interesting fact".
INFO:root:End of audio request detected.
INFO:root:Stopping recording.
INFO:root:Transcript of user request: "tell me an interesting fact".
INFO:root:Playing assistant response.
INFO:root:Finished playing assistant response.
Press Enter to send a new request...
```

7. News and Sports Updates

```
INFO:root:Finished playing assistant response.
Press Enter to send a new request...
INFO:root:Recording audio request.
INFO:root:Transcript of user request: "cric".
INFO:root:Transcript of user request: "cricket".
INFO:root:Transcript of user request: "cricket up".
INFO:root:Transcript of user request: "cricket".
INFO:root:Transcript of user request: "cricket update".
INFO:root:Transcript of user request: "cricket  update".
INFO:root:End of audio request detected.
INFO:root:Stopping recording.
INFO:root:End of audio request detected.
INFO:root:Stopping recording.
INFO:root:Transcript of user request: "cricket update".
INFO:root:Playing assistant response.
INFO:root:Finished playing assistant response.
Press Enter to send a new request...
INFO:root:Recording audio request.
INFO:root:Transcript of user request: "play".
INFO:root:Transcript of user request: "play new".
INFO:root:Transcript of user request: "play news".
INFO:root:Transcript of user request: "play  news".
INFO:root:End of audio request detected.
INFO:root:Stopping recording.
INFO:root:Transcript of user request: "play news".
INFO:root:Playing assistant response.
1 Iran election: Hardliner Raisi set to win in first round
2 Myanmar coup: UN calls for arms embargo against military
3 Troubled US teens left traumatised by tough love camps
4 Alex Harvill: US daredevil dies during world record attempt
5 Portland riots: Police crowd-control team resigns after officer indicted
6 'I was beating the crocodile on its snout'
7 The ethnic armies training Myanmar's protesters
8 Abortion rights: US Catholic bishops face clash with Biden
9 Milkha Singh: India's 'Flying Sikh' dies from Covid
10 Troubled US teens left traumatised by tough love camps
```

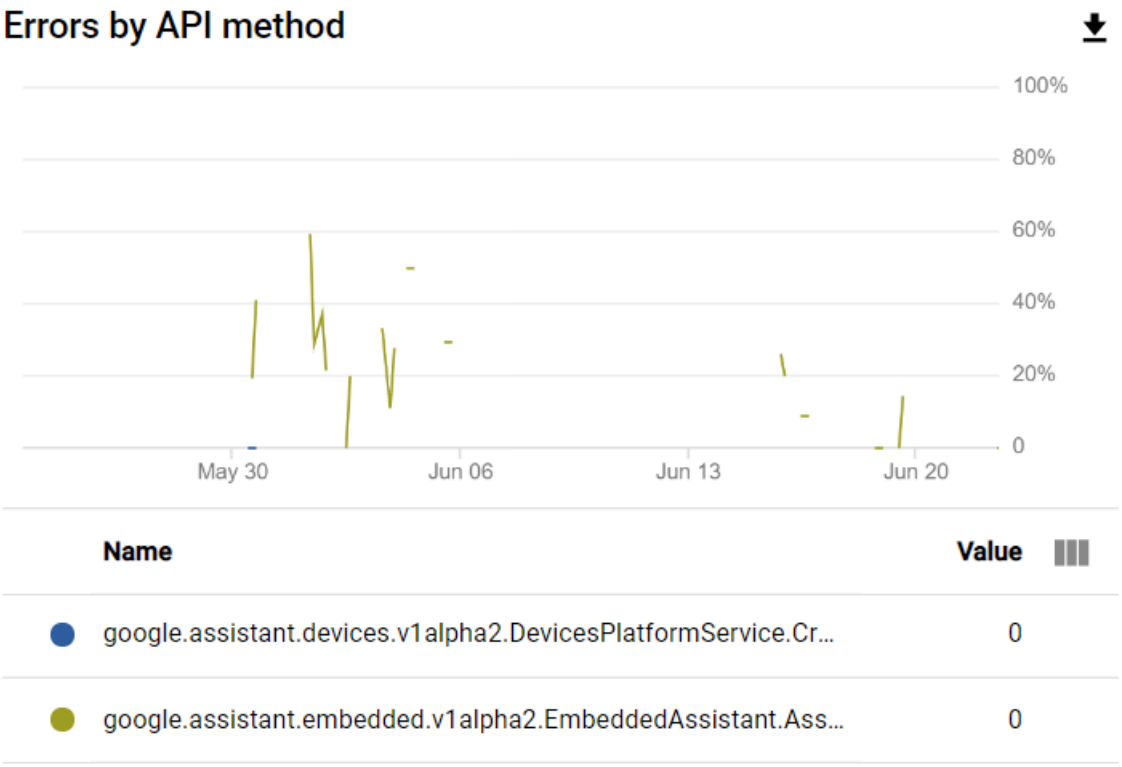
8. Music Playback

```
INFO:root:Stopping recording.
INFO:root:Transcript of user request: "play news".
INFO:root:Playing assistant response.
1 Iran election: Hardliner Raisi set to win in first round
2 Myanmar coup: UN calls for arms embargo against military
3 Troubled US teens left traumatised by tough love camps
4 Alex Harvill: US daredevil dies during world record attempt
5 Portland riots: Police crowd-control team resigns after officer indicted
6 'I was beating the crocodile on its snout'
7 The ethnic armies training Myanmar's protesters
8 Abortion rights: US Catholic bishops face clash with Biden
9 Milkha Singh: India's 'Flying Sikh' dies from Covid
10 Troubled US teens left traumatised by tough love camps
(env) pi@raspberrypi:~ $ googlesamples-assistant-pushtotalk --project-id edith-b5fb1 --device-model-id edith-b5fb1-edith-z7xol7
INFO:root:Connecting to embeddedassistant.googleapis.com
INFO:root:Using device model edith-b5fb1-edith-z7xol7 and device id 275563c8-c10d-11eb-b5de-0800274bb4e8
Press Enter to send a new request...
INFO:root:Recording audio request.
INFO:root:Transcript of user request: "play".
INFO:root:Transcript of user request: "play guess".
INFO:root:Transcript of user request: "play desk".
INFO:root:Transcript of user request: "play despacito".
INFO:root:Transcript of user request: "play  despacito".
INFO:root:Transcript of user request: "play  despacito on".
INFO:root:Transcript of user request: "play  despacito".
INFO:root:End of audio request detected.
INFO:root:Stopping recording.
INFO:root:Transcript of user request: "play despacito".
INFO:root:Playing assistant response.
```

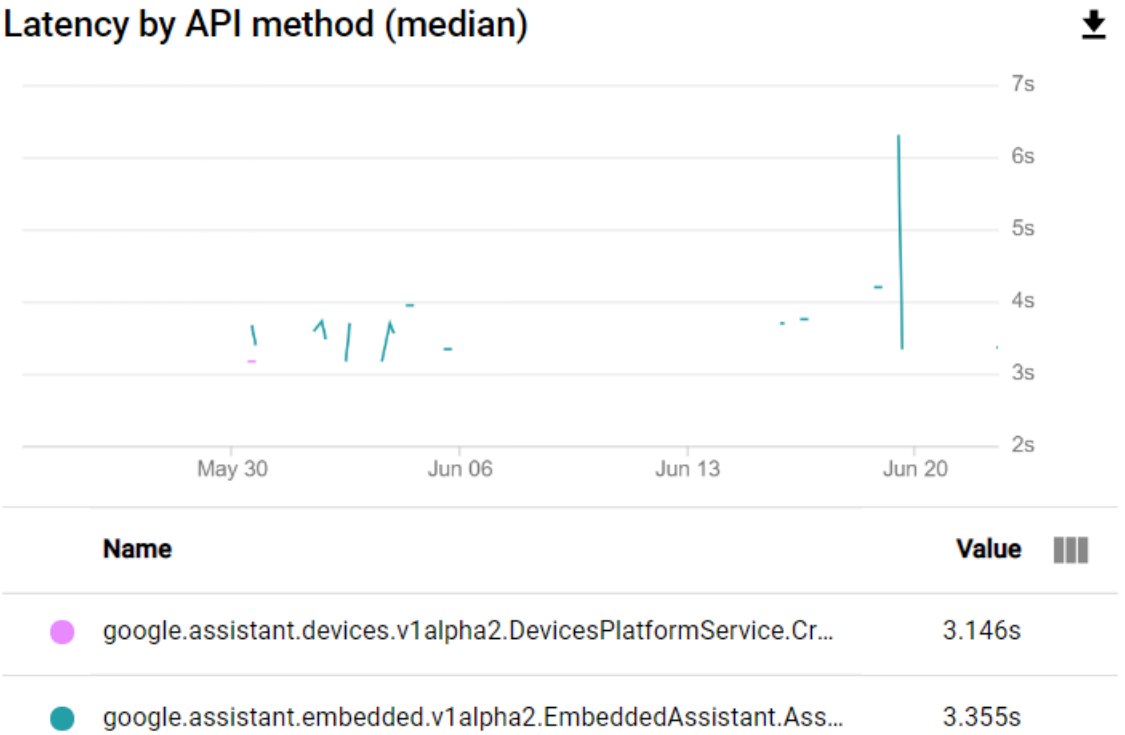


PERFORMANCE TESTING

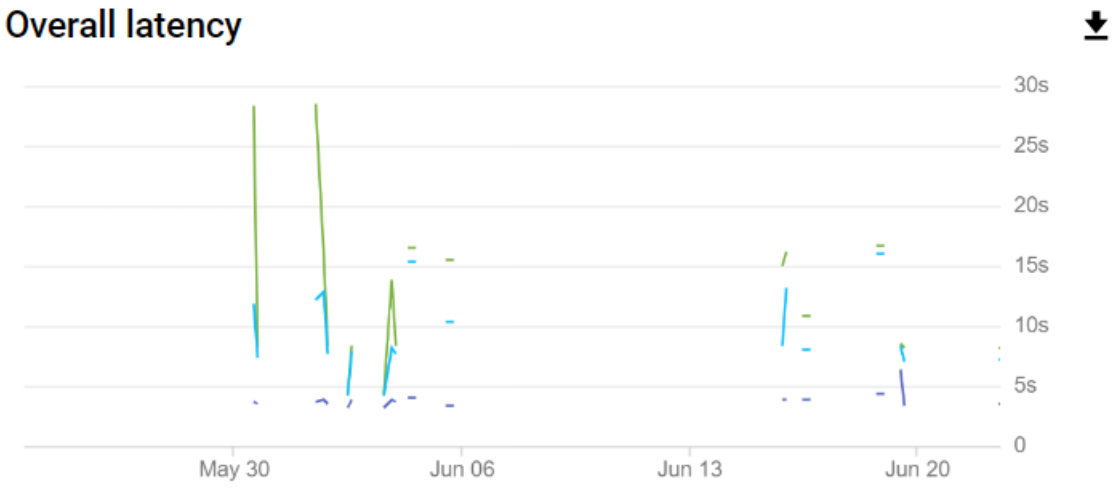
1. Errors encountered



2. Latency by API

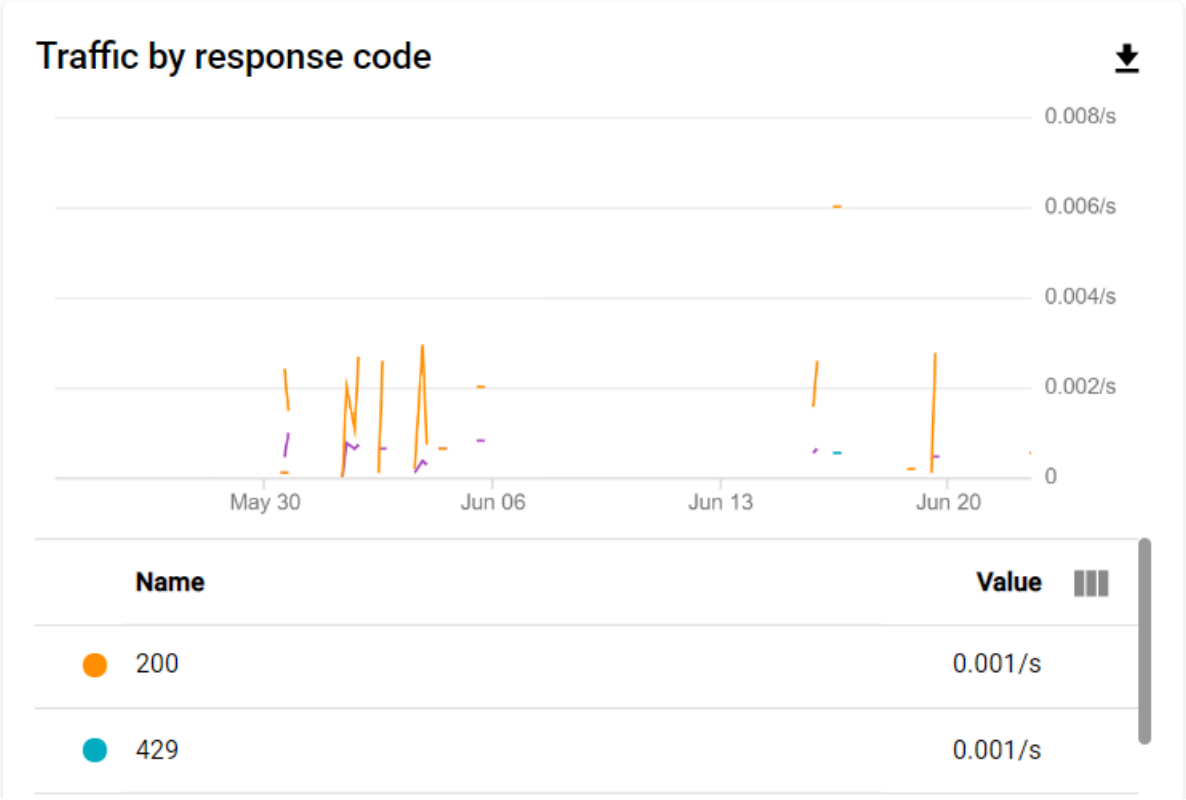


3. Overall Latency



Metric	Name	Value
REDUCE_PERCENTILE_50	p50	3.355s
REDUCE_PERCENTILE_95	p95	7.130s

4. Traffic Response



Name	Value
200	0.001/s
429	0.001/s

CONCLUSION AND FUTURE WORK

Conclusion

The project **"Edith"** has been developed as per the requirement specification. The complete functionality has been thoroughly tested, to eliminate bugs and enhance the user experience.

The design, implementation and the output reports are presented in this project report. The entire project was meticulously designed to ensure seamless user experience and easier incorporation of future modules.

Future Work

The goals of this project were purposely kept within what was believed to be attainable within the allotted timeline and resources. As such, many improvements can be made upon this initial design. That being said, it is felt that the design could be replicated to a much larger scale. The following are the features we wish to add in the future:

- Extend the project to support Home Automation like turning on lights, controlling devices with voice etc.
- Extend to support Messages, SMS and Emails.
- Sending Voice Mails, Play Radio.
- Tracking parcels/orders.
- Booking Movie/Travel Tickets.
- Make purchases in apps.

REFERENCES

[1]. https://developers.google.com/assistant/sdk/guides/library/python

[2]. https://github.com/googlesamples/assistant-sdk-python

[3]. https://pypi.org/

[4]. https://cloud.google.com/apis/docs/getting-started

[5]. https://www.raspberrypi.org/forums/

[6]. https://ubuntuforums.org/

[7]. https://stackoverflow.com/

[8]. https://www.geeksforgeeks.org/

FULL PROJECT DEMONSTRATION

https://www.youtube.com/watch?v=CJJJV1EhqXg