CONTENTS

**Sl no. Title Page no.**

1. Introduction……………………………………………………………………….……..………2
2. Hardware requirements…………………………………………………….…………….3
   1. Raspberry Pi 3B Architecture…………………………………….………..…..….3
3. Software requirement………………………………………………………………..4
4. Programming language……………………………………………….……………….5
   1. Python and IDLE……………………………………………………….…………….5
5. Python program…………………………………………………………….…………….6
   1. Linux commands……………………………………………………….……………9
6. Working…………………………………………………………………………….…………9
7. Conclusion and applications…………………………………………………..…..10
   1. Future scope………………………………………………………………………….11
8. References………………………………………………………………………………….12
9. **INTRODUCTION**

A voice assistant is a digital assistant that uses voice recognition, natural language processing and speech synthesis to provide aid to users through phones and voice recognition applications. It is software that can understand and respond to commands spoken in natural language. Voice assistants are a growing technological trend, used in smartphones, tablets and computers. For example, with the help of Alexa, Cortana and Google Assistant it is possible to control devices even without lifting a finger.

Voice controlled assistants are used in help and service phone lines, smartphones and other places to assist users with tasks, including listening to an audiobook, requesting information, making reservations, adding items to a shopping list, performing mathematical calculations, playing music etc.

Here we have demonstrated a voice controlled assistant using Raspbian OS which turns on and off LED using voice commands. Google API (application program interface) is used since it is an open source which runs on top of Raspbian OS. Raspberry Pi is used as a main hardware to implement this model which works on the primary input of a user’s voice.

1. **HARDWARE REQUIREMENTS**

The following hardware equipments are required to set up the voice controlled assistant using Raspbian OS to turn on and off LED using voice commands:

1. **Raspberry pi-3 with Raspbian OS installed on SD card-8GB minimum-**Raspberry pi is the heart of the voice command system as it is involved in every step of processing data to connecting components together . The Raspbian OS is mounted onto the SD card which is then loaded in the card slot to provide a functioning operating system.
2. **Power Supply-**The Raspberry Pi needs a constant 3-5V, 2.1 mA power supply. This can either be provided through an AC supply using a micro USB charger or through a power bank.
3. **Laptop/PC-**It is needed to access command line of Pi.
4. **USB Mic-**It is needed to input the Voice command into the Raspberry Pi-3.
5. **LAN cable-**It is needed for connecting raspberry Pi using SSH connection.
6. **Speakers and Led lights-**As Input/output devices
7. **Jumper Wires-** To connect I/O devices to GPIO Pins of raspberry pi.

**2.1 Raspberry Pi 3B Architecture**

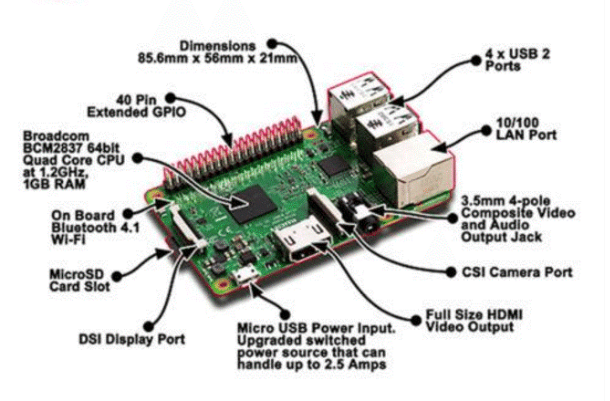
Raspberry pi 3 has a clock speed of about 1.2Ghz, Broadcom BCM2835 (CPU & GPU), 1GB SDRAM, 4 USB 2.0 Ports, Ethernet Port, HDMI, Audio, SD Card Slot, Micro USB for power, 32 bit RISC, SOC-BCM2837. Figure illustrates the architecture of raspberry Pi 3B. 

Figure 1: Raspberry Pi-3 architecture

Following figure illustrates the flowchart of the hardware connections present in the project.

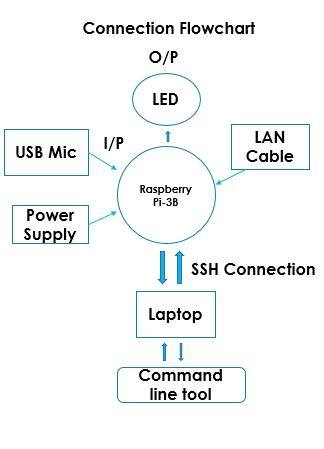


Figure 2: Flowchart of hardware connections

1. **SOFTWARE REQUIREMENT**
2. **Raspbian OS-** It is one of the official Operating systems available for free to download and use. Raspbian is based on Debian optimized for the Raspberry Pi hardware. An operating system is the set of basic programs and utilities that make your Raspberry Pi run. However, Raspbian provides more than a pure OS: it comes with over 35,000 packages, pre-compiled software bundled in a nice format for easy installation on your Raspberry Pi.
3. **LXDE-** The Raspbian desktop environment is known as the “Lightweight X11 Desktop Environment” or in short LXDE.
4. **Mobaxterm –** It is an ultimate toolbox for remote computing.

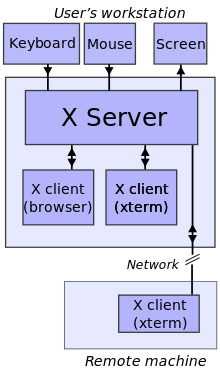


Figure 3: Block diagram of X server

1. **PROGRAMMING LANGUAGE**
   1. **Python and IDLE**

IDLE is integrated development and learning environment (IDLE) for editing and running Python 2.x or Python 3 programs. The IDLE GUI (graphical user interface) is automatically installed with the Python interpreter. IDLE was designed specifically for use with Python. IDLE has a number of features to help in developing Python programs including powerful syntax highlighting.It is packaged as an optional part of the Python packaging with many [Linux distributions](https://en.wikipedia.org/wiki/Linux_distributions). It is completely written in Python and the [Tkinter](https://en.wikipedia.org/wiki/Tkinter) GUI toolkit ([wrapper](https://en.wikipedia.org/wiki/Wrapper_function) functions for [Tcl](https://en.wikipedia.org/wiki/Tcl)/[Tk](https://en.wikipedia.org/wiki/Tk_(framework))). IDLE is intended to be a simple [IDE](https://en.wikipedia.org/wiki/Integrated_development_environment) and suitable for beginners, especially in an educational environment.

To that end, it is cross-platform, and avoids feature clutter. Its main features are:

* Multi-window text editor with [syntax highlighting](https://en.wikipedia.org/wiki/Syntax_highlighting), autocompletion, smart indent and other.
* Python shell with syntax highlighting.
* Integrated debugger with [stepping](https://en.wikipedia.org/wiki/Program_animation), persistent [breakpoints](https://en.wikipedia.org/wiki/Breakpoint), and call stack visibility.

IDLE has been criticized for various usability issues, including losing focus, lack of copying to clipboard feature, lack of line numbering options, and general user interface design; it has been called a "disposable" IDE, because users frequently move on to a more advanced IDE as they gain experience.

1. **PYTHON PROGRAM**

#!/usr/bin/env python

# Copyright (C) 2017 Google Inc.

#

# Licensed under the Apache License, Version 2.0 (the "License");

# you may not use this file except in compliance with the License.

# You may obtain a copy of the License at

#

# http://www.apache.org/licenses/LICENSE-2.0

#

# Unless required by applicable law or agreed to in writing, software

# distributed under the License is distributed on an "AS IS" BASIS,

# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

# See the License for the specific language governing permissions and

# limitations under the License.

from \_\_future\_\_ import print\_function

import RPi.GPIO as GPIO

import argparse

import json

import os.path

import pathlib2 as pathlib

import google.oauth2.credentials

from google.assistant.library import Assistant

from google.assistant.library.event import EventType

from google.assistant.library.file\_helpers import existing\_file

from google.assistant.library.device\_helpers import register\_device

try:

FileNotFoundError

except NameError:

FileNotFoundError = IOError

WARNING\_NOT\_REGISTERED = """

This device is not registered. This means you will not be able to use

Device Actions or see your device in Assistant Settings. In order to

register this device follow instructions at:

https://developers.google.com/assistant/sdk/guides/library/python/embed/register-device

"""

def process\_event(event):

"""Pretty prints events.

Prints all events that occur with two spaces between each new

conversation and a single space between turns of a conversation.

Args:

event(event.Event): The current event to process.

"""

if event.type == EventType.ON\_CONVERSATION\_TURN\_STARTED:

print()

print(event)

if (event.type == EventType.ON\_CONVERSATION\_TURN\_FINISHED and

event.args and not event.args['with\_follow\_on\_turn']):

print()

if event.type == EventType.ON\_DEVICE\_ACTION:

for command, params in event.actions:

print('Do command', command, 'with params', str(params))

if command == "action.devices.commands.OnOff":

if params['on']:

print('Turning the LED on.')

GPIO.output(25, 1)

else:

print('Turning the LED off.')

GPIO.output(25, 0)

def main():

parser = argparse.ArgumentParser(

formatter\_class=argparse.RawTextHelpFormatter)

parser.add\_argument('--device-model-id', '--device\_model\_id', type=str,

metavar='DEVICE\_MODEL\_ID', required=False,

help='the device model ID registered with Google')

parser.add\_argument('--project-id', '--project\_id', type=str,

metavar='PROJECT\_ID', required=False,

help='the project ID used to register this device')

parser.add\_argument('--device-config', type=str,

metavar='DEVICE\_CONFIG\_FILE',

default=os.path.join(

os.path.expanduser('~/.config'),

'googlesamples-assistant',

'device\_config\_library.json'

),

help='path to store and read device configuration')

parser.add\_argument('--credentials', type=existing\_file,

metavar='OAUTH2\_CREDENTIALS\_FILE',

default=os.path.join(

os.path.expanduser('~/.config'),

'google-oauthlib-tool',

'credentials.json'

),

help='path to store and read OAuth2 credentials')

parser.add\_argument('-v', '--version', action='version',

version='%(prog)s ' + Assistant.\_\_version\_str\_\_())

args = parser.parse\_args()

with open(args.credentials, 'r') as f:

credentials = google.oauth2.credentials.Credentials(token=None,

\*\*json.load(f))

device\_model\_id = None

last\_device\_id = None

try:

with open(args.device\_config) as f:

device\_config = json.load(f)

device\_model\_id = device\_config['model\_id']

last\_device\_id = device\_config.get('last\_device\_id', None)

except FileNotFoundError:

pass

if not args.device\_model\_id and not device\_model\_id:

raise Exception('Missing --device-model-id option')

# Re-register if "device\_model\_id" is given by the user and it differs

# from what we previously registered with.

should\_register = (

args.device\_model\_id and args.device\_model\_id != device\_model\_id)

device\_model\_id = args.device\_model\_id or device\_model\_id

with Assistant(credentials, device\_model\_id) as assistant:

events = assistant.start()

device\_id = assistant.device\_id

print('device\_model\_id:', device\_model\_id)

print('device\_id:', device\_id + '\n')

GPIO.setmode(GPIO.BCM)

GPIO.setup(25, GPIO.OUT, initial=GPIO.LOW)

# Re-register if "device\_id" is different from the last "device\_id":

if should\_register or (device\_id != last\_device\_id):

if args.project\_id:

register\_device(args.project\_id, credentials,

device\_model\_id, device\_id)

pathlib.Path(os.path.dirname(args.device\_config)).mkdir(

exist\_ok=True)

with open(args.device\_config, 'w') as f:

json.dump({

'last\_device\_id': device\_id,

'model\_id': device\_model\_id,

}, f)

else:

print(WARNING\_NOT\_REGISTERED)

for event in events:

process\_event(event)

if \_\_name\_\_ == '\_\_main\_\_':

main()

**5.1 Linux commands**

The various Linux commands used in the above voice controlled assistant are

**1)arecord –l** : To Locate USB microphone in the list of capture hardware devices.

**2)aplay –l** :To locate speaker in the list of playback hardware devices.

**3).asoundrc** file contains the device hardware number .

**4)alsamixer :** Used to adjust Speaker and microphone level.

**5)source env/bin/activate:** Entering Python Virtual Environment.

**6)Python hotword.py :** Running the code file.

1. **WORKING**

The steps involved in working of the code written to turn on and off LED using voice commands are:

1. Google Cloud Speech-to-Text enables developers to convert audio to text by applying powerful neural network models in an easy-to-use API.
2. Natural Language Processing :Current approaches to NLP are based on deep learning, a type of AI that examines and uses patterns in data to improve a program's understanding
3. GPIO Pins-40: BCM25 i.e. 22pin and Gnd pin 9 are used.
4. BCM25 is an output pin activated as per Voice commands.

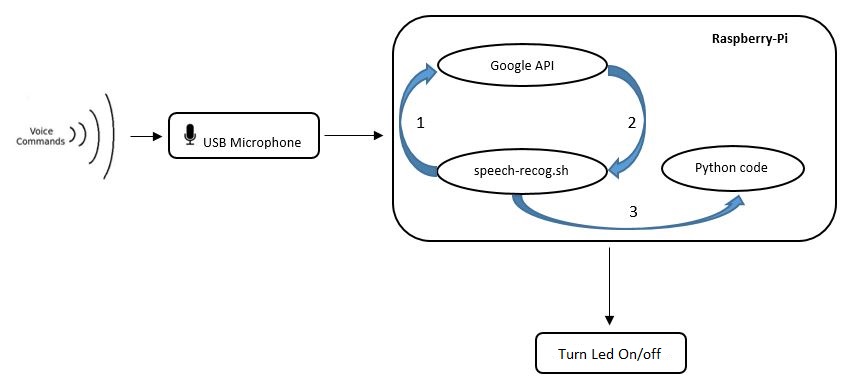


Figure 4: Illustration of working of designed voice controlled assistant

1. **CONCLUSION AND APPLICATIONS**

Many devices we use every day utilize voice assistants. They’re on our smartphones and inside smart speakers in our homes. Many mobile apps and operating systems use them. Additionally, certain technology in cars, as well as in retail, education, healthcare, and telecommunications environments, can be operated by voices.

Siri by Apple became the first digital virtual assistant to be standard on a smartphone when the iPhone 4s came out on October 4, 2011. Siri moved into the smart speaker world when the HomePod debuted in February 2018.

Google Now (which became Google Assistant) on the Android platform followed. It also works on Apple’s iOS, but has limited functionality. Then the smart speakers came along, and “Alexa” and “Hey Google” became a part of many household conversations. Alexa by Amazon is part of the Echo and the Dot. Google Assistant is part of the Google Home.

Samsung has Bixby. IBM has Watson. Microsoft has Cortana on its Windows 10, Xbox One machines, and Windows phones, and Nuance has Nina. Facebook used to have M, but its usage in the Facebook Messenger app ended in January 2018.

* 1. **Future Scope**

Technology is constantly advancing and changing, and the voice assistant market will progress along with it. In April 2015, the research firm [Gartner](https://www.gartner.com/doc/3021226/market-trends-voice-ui-consumer) predicted that by the end of 2018, 30 percent of interactions with technology would be through “conversations” with smart machines, many of them by voice.

Tractica is a market intelligence firm that focuses on human interaction with technology. Their [reports](https://www.tractica.com/newsroom/press-releases/the-virtual-digital-assistant-market-will-reach-15-8-billion-worldwide-by-2021/) say unique consumer users for virtual digital assistants (which they define as automated software applications or platforms that assist the human user through understanding natural language in written or spoken form) will grow from more than 390 million worldwide users in 2015 to 1.8 billion by the end of 2021. The growth in the business world is expected to increase from 155 million users in 2015 to 843 million by 2021. With that kind of projected growth, revenue is forecasted to grow from $1.6 billion in 2015 to $15.8 billion in 2021.

According to [Global Market Insights, Inc](https://gminsights.wordpress.com/tag/virtual-assistant-industry-statistics/)., between 2016 and 2024, the market share for the technology will grow at an annual rate of almost 35 percent. More and more sectors of the economy, like healthcare and the automotive industry, are finding uses for the speech recognition technology in addition to those found in devices like smart speakers and phones.

Future scope of our project deals with Device actions and Device Traits.

* 1. **Device Traits :** The Google Assistant needs to be able to associate a query with a command to send to your device. For this to work, we need to declare what kinds of abilities our device supports. These abilities are known as traits.
  2. **Device actions** :Device may have special abilities not covered by the current set of traits.
  3. **Actions.json :** File has to be edited to get desired response.

1. **REFERENCES**

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

**[2]** [**https://www.codeproject.com/Articles/839230/Introduction-to-Raspberry-Pi-with- Raspbian-OS**](https://www.codeproject.com/Articles/839230/Introduction-to-Raspberry-Pi-with-%20%20%20%20%20%20%20%20Raspbian-OS)

**[3]** [**https://cloud.google.com/speech-to-text/**](https://cloud.google.com/speech-to-text/)

**[4]** [**https://searchbusinessanalytics.techtarget.com/definition/natural-language-processing-NLP**](https://searchbusinessanalytics.techtarget.com/definition/natural-language-processing-NLP)

**[5]** [**https://en.wikipedia.org/wiki/IDLE**](https://en.wikipedia.org/wiki/IDLE)

**[6]** [**https://searchsecurity.techtarget.com/definition/Secure-Shell**](https://searchsecurity.techtarget.com/definition/Secure-Shell)