SMART MIRROR USING RASPBERRY PI

A PROJECT REPORT OF CAPSTONE PROJECT 2

Submitted by

TATHAGAT SINHA

(1513101658/15SCSE101346)

In partial fulfilment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the supervision of

JAYAKUMAR VAITHIYASHANKAR

APRIL/MAY-2020

GALGOTIAS UNIVERSITY

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report **"SMART MIRROR USING RASPBERRY PI"** is the bonafide work of **"TATHAGAT SINHA(1513101658)"** who carried out the project work under my supervision.

Table of Contents

LIST OF FIGURES

ABSTRACT

This project has been developed within the context of a time where every day we see more and more connected devices. The Internet transformed our lives by connecting us more easily to information and other people in the virtual world. Mobile phones then became smartphones and since then this concept has erupted and morphed into the Internet of Things, things which connect us to everyday objects. There are no end of objects that could be made “smarter”, some being more suited to this than others. Mirrors, for example, provide a large surface ideal for displaying information and interacting with. Most people have mirrors at home so the concept of a smart mirror that you can interact with is attractive and has been fantasized in many futuristic movies. Smart mirrors, such as Magic Mirror and Home Mirror have recently started to be developed by people in the Maker community, with varying degrees of interactivity. However, so far, the features of these mirrors have been limited. This final year project describes how a smart mirror was built from scratch using a Raspberry Pi for the hardware and custom software built on top of Raspbian, a Linux distribution. The goal of the project was to create a Smart Mirror device that people could interact with but also to further develop the technology so that it would let you install and develop your own applications for it. On the whole results were good because a higher level of interactivity has been achieved by being able to use voice commands, gestures and smartphones. A few problems arose in the construction and software side of the project, such as the glass not being reflective enough and the gesture recognition being unreliable but these drawbacks can be addressed by doing more tests and trials to further develop the Smart Mirror.

INTRODUCTION

1.1 What is Smart Mirror?

Everyone knows what a mirror is. It is an object found in most people's homes. In mirrors we see our reflections. But what happens when you combine the idea of a mirror with technology? What possibilities are there and how smart could a mirror be? These are some of the questions that inspired my choice of final year project, a project which aimed to develop a smart mirror and a small operating system to power it. The device was to go beyond an ordinary mirror, to have a screen inside that you would be able to interact with by using voice commands, hand gestures and smartphones or other devices.

Multimedia is a very broad area and I like every aspect of it so it was difficult to choose a specific area and I had many ideas. However, I finally decided to build a smart mirror because it is a great combination of many of the things we have studied: web technologies, electronics, UI design, etc. The smart mirror is a popular project among DIY enthusiasts and it usually consists of a one-way mirror with a screen attached to it that displays a static web page. However, what I wanted to achieve was something you could interact with. My goal was to learn how a Raspberry Pi worked and to understand how to combine the software and the hardware components to create a multimedia project.

I started by obtaining a Raspberry Pi and creating the software. At the same time, I began documenting everything and I also searched for a suitable one--way mirror and a computer screen, as well as some sensors to physically interact with the device. I then spent a long time calibrating the sensors to work with the software. Once the software was almost finished, I started designing the frame and finally I built the smart mirror and attached all the components.

Developing this project has been a great experience. I have learned a diverse range of skills in different fields, such as DIY, Linux, electronics and web development. To obtain the final result I've had to work with many different technologies. I used Photoshop and Illustrator for the UI designs, web development tools for the software and electronics for the hardware. Not sticking to just one field has made this project a really fun one and I would recommend it to anyone who is passionate about creating things.

1.2 Internet of Things

The Internet of Things is a concept defined as a network of connected physical objects (Internet of things, 2016). It's often viewed as the next step for the internet. Recently it has gained a lot of popularity predicting that in the future most everyday objects will be connected to each other and will be able to interact in smart ways. The Smart Mirror will eventually become one of these connected objects in our households and if we think about it being able to communicate with other objects the possibilities become endless.

1.3 Maker culture

The maker culture is a contemporary culture derived from DIY culture and hacker culture(Maker culture, 2016). It focuses in the creation of new devices as well as modifying existing ones. It often supports and embraces open-source hardware and software. This culture has been growing rapidly thanks to tools and technology like the Raspberry Pi, 3D printers and other hardware that have become increasingly affordable and accessible. The Internet also plays a big part in the community as it enables people to share their ideas, blueprints and code.
Smart Mirror is a good example of a Maker culture project.

1.4 Home automation

Home automation has been around for a long time and it is all about turning the house into an intelligent unit with the goal of increasing comfort and efficiency at home. Some of the typical applications are automatic lights, intelligent thermostats, alarms, window blinds (Home automation, 2013). In my project I will not be focusing on home automation since I don't have access to any smart home devices. However it would be very easy to write an application to turn on and off the lights using voice commands or gestures on the mirror or even an application to change the temperature of the room, for example. These examples are just the tip of the iceberg as there are new connected devices emerging everyday that could interact with the mirror.

1.5 Structure for the Smart Mirror

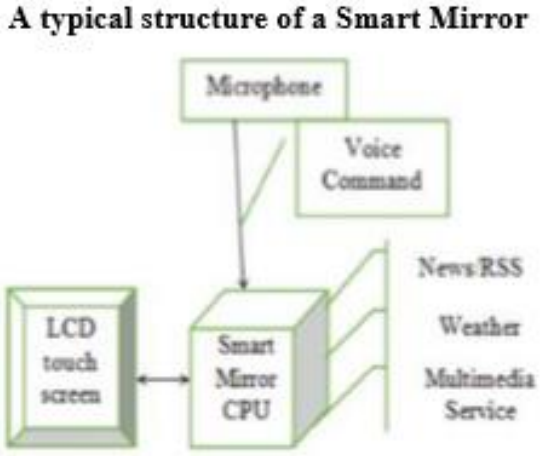

FIG. 1

LITERATURE SURVEY

Today smart homes and virtual assistant are trending among the people. Amazon, Google and Phillips are presenting their advance technology in the field of smart home or AmI. People are excited too for their amazing products and for these futuristic devices. Phillips HomeLab is a leading company for creating digital home environments. Interactive mirror is one of their project for home environments. This mirror supports playing music or videos. This mirror consists a normal mirror on a LED which performs the playback feature.

There are different kind of smart mirrors that are being proposed or available in the market. Some of them are discussed below:

2.1 Microsoft's Magic Mirror

This mirror is proposed by Microsoft in 2016. This smart mirror works on Windows 10 IoT Core on Raspberry Pi 3. This is powered by Windows Hello cognitive services. This was an open source project. Its web app was made open to GitHub repository so that anyone can build its own smart mirror. The mirror shows traffic updates, weather and supports voice recognition.

2.2 Ekko Smart Mirror

This smart mirror runs on their own linux based platform on Raspberry Pi and it required an installed app on the user's smartphone. It also has sensors which could recognize the gestures of the user. Other than highlighting news, weather and time, the user can also play videos and music.

2.3 Apple Mirror (Rafael Dymek)

This smart mirror prototype is based on iOS 10 that mirror the iPhone display. The mirror can launch all the mobile apps desired by the user. This mirror sleeps after every 45 seconds of ideal situation. This is a touchscreen smart mirror.

2.4 Nuovo Smart Mirror

This android based smart mirror required an android application on the user's smartphone. The mirror supports music and videos playback. This mirror also supports features like weather, maps and the social networking like Twitter, Facebook, etc. The auto sleep mode is also supported by the mirror.

2.5 Perseus Smart Mirror

This smart mirror runs on the separate platform on Raspberry Pi. This mirror doesn't require any application on the smartphone. This mirror is available in different sizes. This mirror supports music, videos and social networking.

2.6 Naked 3d Fitness Tracker

This mirror consists a huge number of sensors which reads a 3d scan of the body and checks for any formational abnormality. It also senses the area of the body which is prone to an injury. Its also suggests workout plans to be fit.

2.7 A Detailed Comparison of Smart Mirrors

A Detailed Comparison of Mirrors

Feature	Microsoft's Magic Mirror	Ekko Smart MIrror	Apple Mirror-Rafael Dymek	Nuovo Smart Mirror	Perseus Smart Mirror	Naked 3D Fitness Tracker
Platform	Windows 10	Customized	iOS 10	Android	Customized	Customized
App Requirement	No	Yes	Yes	Yes	No	No
Voice Recognition	Yes	No	No	No	Yes	No
Touchscreen	No	No	Yes	No	No	No
Gestures	No	Yes	No	No	No	Yes
Fitness	No	No	No	No	No	Yes
Music Support	Yes	Yes	Yes	Yes	Yes	No
Video Support	Yes	Yes	No	Yes	Yes	No
Automatic Sleep	No	No	Yes	Yes	No	No
Weather	Yes	Yes	Yes	Yes	Yes	No
Map	Yes	No	Yes	Yes	Yes	No
Social Networking	Yes	No	Yes	Yes	Yes	No

FIG. 2

PROBLEM STATEMENT

The major problem of any existing mirror is displaying just the object in front of it or just the human face without having to interact with them. This project is developed with the intention that people spend quality time in front of the mirror.

PROPOSED SYSTEM

We plan to design and develop such kind of futuristic smart mirror which provides a whole new experience to the user with the flavor of AmI. Our proposed smart mirror consists a two-way mirror, acrylic glass, monitor (LED), Raspberry Pi, Raspberry Modules, sensors.

A wooden frame will be prepared with LED attached behind the glass with all the sensors and the raspberry pi. The power supply is attached to the raspberry pi which will power the LED monitor and the sensors.

Once the mirror is activated, it will connect to the docker which contains all api and software needed to run the mirror. This will require internet access which will be provided by the wi-fi module (LAN can be also used) on the raspberry pi.

The virtual layout that will be prepared using HTML and CSS will be displayed on the mirror when it is turned on and will show calendar, weather and news headlines. The docker will contain the api of Alexa (virtual voice assistant from Amazon) that will respond to the user's voice.

The mirror will perform facial recognition which will be helpful for real time image zoom in and out. This will be one with help of OpenCV and some java programming.

The proposed smart mirror will perform these tasks:

1. A normal two-way mirror and acrylic glass will display real time image.
2. After activation the mirror will display weather, time and news.
3. The mirror can play music and videos.
4. The mirror can zoom in and out real-time images.
5. The mirror will automatically sleep if a person disappears from front with the help of sensors.
6. The mirror can be used as displaying moving images and animations in case of ideal situation with the help of sensors which will detect the presence and absence of any person in front of the mirror.
7. Through Uber api, the mirror can book a ride on Uber.
8. All the social networking websites or apps can be accessed with the voice.
9. The mirror can perform real time photo editing.
10. The mirror can be synced with other devices which leads to the home automation.
11. The mirror also supports multiple user's profile.
12. YouTube videos are also supported by the mirror.

EXISTING SYSTEM

5.1 HARDWARE

For the hardware I used a 24” LG computer monitor, a 50x90x0.5cm one-way mirror a Raspberry Pi 2, two USB microphones and two ultrasonic sensors. Everything was put together in a wooden frame.

These are the final sketches for the hardware design:

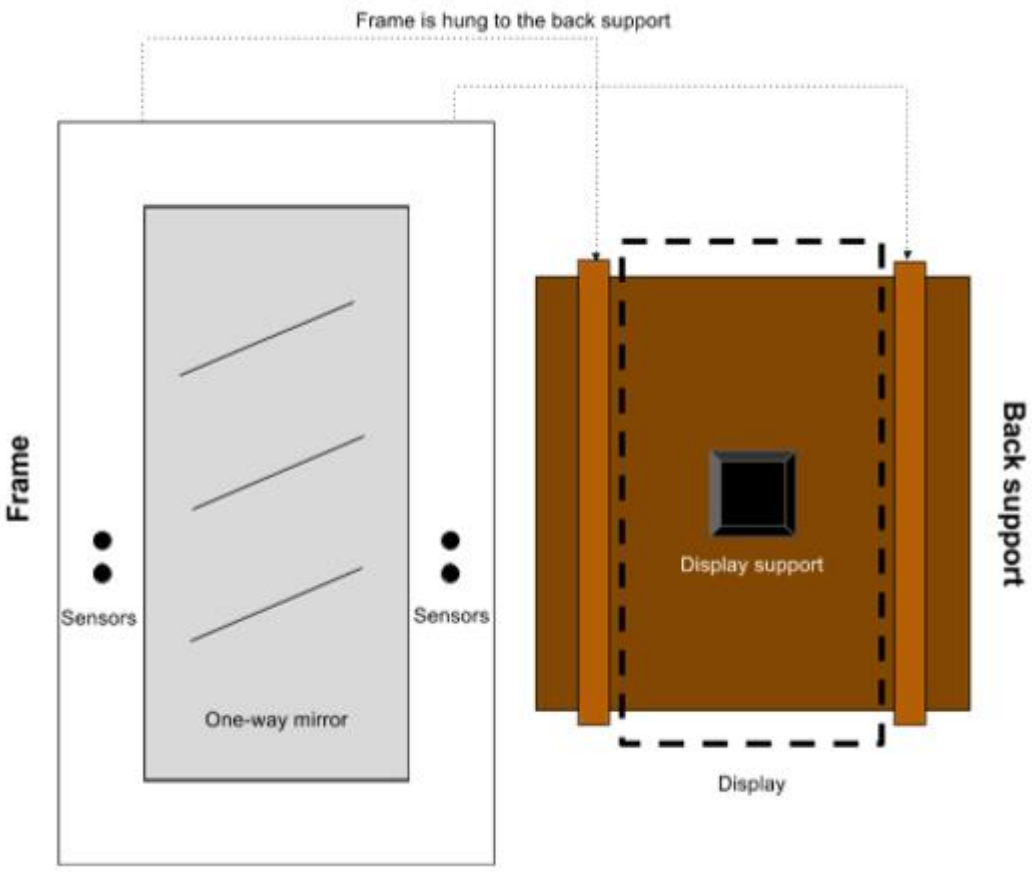

FIG. 3

The device has two wooden parts. The back part holds the display and the Raspberry Pi and is used to support the device so that it can be hung on a wall. The frame is attached to the glass by two small wooden slats and it has four holes, two on each side, that contain the ultrasound sensors. The frame can be attached and detached from the back part so it’s easy to change the

glass or even the whole frame. See appendices 1 and 2. A breakdown of each of the main parts of the smart mirror (the one-way mirror glass, display, Raspberry Pi 2, microphones, ultrasonic sensors and frame) and how they were used is described in the following sections:

5.1.1 One-way mirror

This is probably the most important part of the hardware because it's responsible for creating the futuristic effect and is the biggest part of the smart mirror. Wikipedia provides the following definition:
A one-way mirror, sometimes called two-way mirror, is a mirror that is partially reflective and partially transparent. When one side of the mirror is brightly lit and the other is dark, it allows viewing from the darkened side but not vice versa.

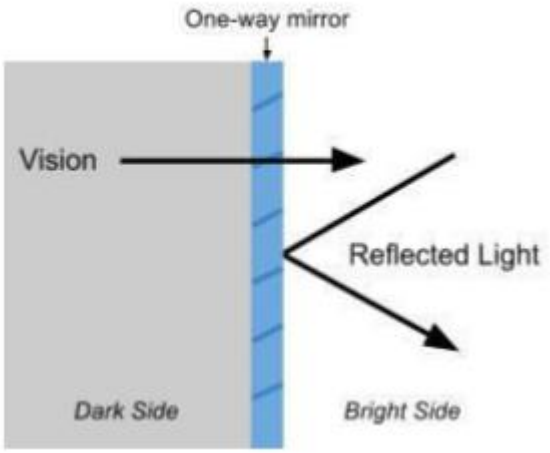

FIG.4

5.1.2 Display

The monitor is much smaller than the mirror so a black sticker was used to cover the parts of the glass which are not covered by the display. An HDMI cable was used to connect the display to the Raspberry Pi for video and audio.

5.1.3 Raspberry Pi 3

The Raspberry Pi is a single-board computer developed by the Raspberry Pi foundation in the UK. It has become the most popular computer of it's kind thanks to great support and a big community behind it as well as an inexpensive price. The Pi does not work out of the box. It lacks a hard drive and it does not come with a preinstalled operating system. To install an OS you need a microSD card prepared with an OS image. And because the software that will be running on the mirror will be coded on the same device at least a screen, a keyboard and a mouse are required.

FIG. 5

5.1.4 Microphones

One mode of interaction with the smart mirror is through microphones. Two microphones were used to power the voice recognition capabilities of the device. USB microphones had to be used because the Raspberry Pi does not have a regular microphone input. The first microphone is a
cheap simple one connected through a USB sound card to the Pi.
The voice recognition system works by listening for someone to clap with the first microphone and once that happens the second, higher quality microphone is triggered to listen for a voice command.

FIG. 6

5.1.5 Ultrasonic sensors

The ultrasonic sensors are the second way to interact with the smart mirror. An ultrasonic sensor has two main parts, a speaker and a microphone. It works by sending an ultrasound with the speaker and returning the time it takes to capture the echo with the microphone. With the time it takes and the speed of sound we can then calculate the distance of an object from the sensor.

FIG. 7

5.1.6 Frame and support

The frame is made of wood and it provides the support for the mirror and all the other components. It frames the glass and provides a way for hanging the mirror on a wall.

5.2 SOFTWARE

5.2.1 Voice Assistance Using Python Library

As we know Python is a suitable language for script writers and developers. Let’s write a script for Voice Assistant using Python. The query for the assistant can be manipulated as per the user’s need. Speech recognition is the process of converting audio into text. This is commonly used in voice assistants like Alexa, Siri, etc. Python provides an API called speech recognition to allow us to convert audio into text for further processing. In this article, we will look at converting large or long audio files into text using

the speech recognition API in python. Many others modules are also used with Speech recognition to develop the voice assistant. Some of them are described below:

- **Subprocess:** This module is used for getting system subprocess details which are used in various commands i.e Shutdown, Sleep, etc. This module comes buit-in with Python.

- **Wolframalpha:** It is used to compute expert-level answers using Wolfram's algorithms, knowledgebase and AI technology. To install this module type the below command in the terminal.

- **Pyttsx3:** This module is used for conversion of text to speech in a program it works offline. To install this module type the below command in the terminal.

- **Tkinter:** This module is used for building GUI and comes inbuit with Python. This module comes buit-in with Python.

- **Wikipedia:** As we all know Wikipedia is a great source of knowledge just like GeeksforGeeks we have used Wikipedia module to get information from Wikipedia or to perform Wikipedia search. To install this module type the below command in the terminal.

- **Speech Recognition:** Since we're building an Application of voice assistant, one of the most important things in this is that your assistant recognizes your voice (means what you want to say/ ask). To install this module type the below command in the terminal.

- **Web browser:** To perform Web Search. This module comes built-in with Python.
- **ES capture:** To capture images from your Camera. To install this module type the below command in the terminal

- **Pyjokes:** Pyjokes is used for collection Python Jokes over the Internet. To install this module type the below command in the terminal.

- **Datetime:** Date and Time is used to showing Date and Time. This module comes built-int with Python.

- **Twilio:** Twilio is used for making call and messages. To install this module type the below command in the terminal.

- **Requests:** Requests is used for making GET and POST requests. To install this module type the below command in the terminal.

- **BeautifulSoup:** Beautiful Soup is a library that makes it easy to scrape information from web pages. To install this module type the below command in the terminal.

In this program I have tried to cover most of the functions that has to be performed by the smart mirror by using voice assistance this program runs after the successful face recognised by the user. Functions performed by the voice assistance are:

- Weather at your location.
- Current time and date
- News Headlines

- Business News, Technology News, World News, Sports News
- Calender
- Play music, videos, documents , etc
- Quotes
- Activates spy camera

Below is the code of the voice-assistance program which you can see in appendix 1.

5.2.2 Face Recognition for User

Recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library. Built using dlib's state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark. This also provides a simple face-recognition command line tool that lets you do face recognition on a folder of images from the command line!

This is the default program in which when the mirror senses someone near him through ultrasonic sensors, the mirror automatically execute this task and automatically ask the user whether he can take the photograph for unlock the features of the mirrors. When successfully recognised automatically

voice assistance starts to work further and we are ready to give the commands as per as our needs.

Below is the code for face-recognition program which you can see in appendix 2

5.2.3 Raspbian

Raspbian is the recommended operating system for normal use on a Raspberry Pi. Raspbian is a free operating system based on Debian, optimised for the Raspberry Pi hardware. Raspbian comes with over 35,000 packages: precompiled software bundled in a nice format for easy installation on your Raspberry Pi. Raspbian is a community project under active development, with an emphasis on improving the stability and performance of as many Debian packages as possible.

5.2.4 MirrorOS

MirrorOS is the software created for the Smart Mirror’s interface and it runs on top of Raspbian and on top of Electron. In the following figure you can see the layers of the software stack.

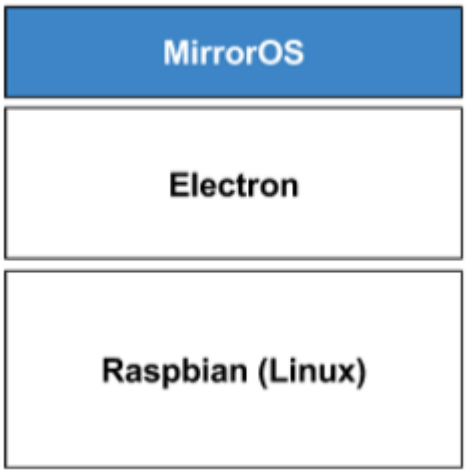

FIG. 8

MirrorOS has three main services:

1.The voice input service, used to handle all the voice recognition process usi ng2.The gesture input service, used to handle gesture recognition using the ultrasonic sensors.

3.A socket server which is in charge of communicating with smartphones or other devices.

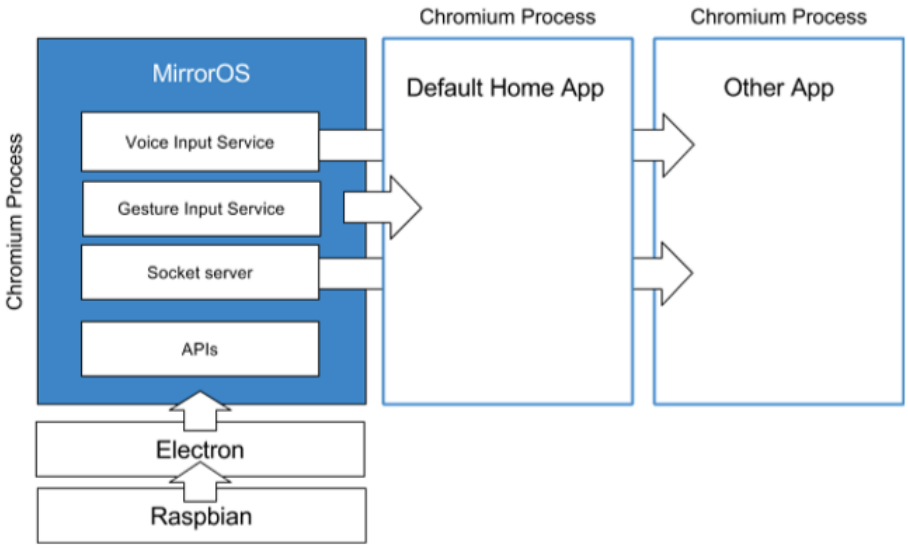

FIG. 9

5.2.5 gesture control

In this module I had tried to work with the ultrasonic sensors i.e when the user come near the smart mirror it will automatically open the face recognition program and ask the user to recognise the face. This feature is only used for on/off the smart mirror. But later on we will try to do gesture control from fingers.

Below is the gesture-control code which you can see in appendix 3

RESULT

As we had seen in the comparison table that every mirror is working on different technologies and platforms. These mirrors also differ in functionalities and users. We had proposed a mirror which works on common architecture and also had all the required functions for the user.

CONCLUSION

We had proposed the comparative study and a design of a futuristic smart mirror which could be great device for ambient home services. Speech recognition is one of the major advantages of the mirror. Live animations will make the bathroom more fashionable. The proposed smart can be easily extended for some other frameworks like making phone calls. In future this mirror can be used to build smart home network with devices such as lights, virtual assistant, TV, music system, refrigerators, etc. can be integrated together. This would lead to real smart home.

REFERENCE

1.https://www.github.com/MichMich/MagicMirror

2. https://www.github.com/aishmittal/Smart-Mirror

3. https://www.geeksforgeeks.org/voice-assistant-using-python/?ref=rp

4. 1. D.K.Mittal – a comparative study and a new model for smart mirror- International journal of scientific research in computer science and engineering volume 5.

5. ekko smart mirror

6. Apple mirror

7. Nuovo smart mirror

8. Perseus smart mirror

9. Naked 3d fitness trackeR

10. Microsoft Smart Mirror

APPENDIX

APPENDIX 1

Voicerecognition.py

```
import subprocess
import wolframalpha
import pyttsx3
import json
import random
import speech_recognition as sr
import datetime
import wikipedia
import webbrowser
import os
import winshell
import pyjokes
import smtplib
import ctypes
import time
import requests
import shutil
from urllib.request import urlopen
engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[1].id)
def speak(audio):
    engine.say(audio)
    engine.runAndWait()
def wishMe():
    hour = int(datetime.datetime.now().hour)
    if hour>= 0 and hour<12:
        speak("Good Morning Sir !")

    elif hour>= 12 and hour<18:
        speak("Good Afternoon Sir !")

    else:
        speak("Good Evening Sir !")
```

```
    assname =("Jarvis 1 point o")
    speak("I am your Assistant")
    speak(assname)
def usrname():
    speak("What should i call you sir")
    uname = takeCommand()
    speak("Welcome Mister")
    speak(uname)
    columns = shutil.get_terminal_size().columns

    print("#####################".center(columns))
    print("Welcome Mr.", uname.center(columns))
    print("#####################".center(columns))

    speak("How can i Help you, Sir")
def takeCommand():

    r = sr.Recognizer()

    with sr.Microphone() as source:

        print("Listening...")
        r.pause_threshold = 1
        audio = r.listen(source)

    try:
        print("Recognizing...")
        query = r.recognize_google(audio, language ='en-in')
        print(f"User said: {query}\n")

    except Exception as e:
        print(e)
        print("Unable to Recognizing your voice.")
        return "None"

    return query
def sendEmail(to, content):
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.ehlo()
    server.starttls()

    # Enable low security in gmail
```

```
    server.login('tathagatsinha254@gmail.com', 'wisemanschool')
    server.sendmail('tathagatsinha254@gmail.com',
'tathagatsinha@rediffmail.com', 'content')
    server.close()
if __name__ == '__main__':
    clear = lambda: os.system('cls')

    # This Function will clean any
    # command before execution of this python file
    clear()
    wishMe()
    usrname()

    while True:

        query = takeCommand().lower()

        # All the commands said by user will be
        # stored here in 'query' and will be
        # converted to lower case for easily
        # recognition of command
        if 'wikipedia' in query:
            speak('Searching Wikipedia...')
            query = query.replace("wikipedia", "")
            results = wikipedia.summary(query, sentences = 3)
            speak("According to Wikipedia")
            print(results)
            speak(results)

        elif 'open youtube' in query:
            speak("Here you go to Youtube\n")
            webbrowser.open("youtube.com")

        elif 'open google' in query:
            speak("Here you go to Google\n")
            webbrowser.open("google.com")

        elif 'open stackoverflow' in query:
            speak("Here you go to Stack Over flow.Happy coding")
            webbrowser.open("stackoverflow.com")

        elif 'play music' in query or "play song" in query:
            speak("Here you go with music")
```

```
            # music_dir = "G:\\Song"
            music_dir = "E://Songs"
            songs = os.listdir(music_dir)
            print(songs)
            random = os.startfile(os.path.join(music_dir, songs[1]))

        elif 'the time' in query:
            strTime = datetime.datetime.now().strftime("%H:%M:%S")

            speak(f"Sir, the time is {strTime}")

        elif 'email to galgotiasuniversity' in query:
            try:
                    speak("What should I say?")
                    content = takeCommand()
                    to = "Receiver email address"
                    sendEmail(to, content)
                    speak("Email has been sent !")
            except Exception as e:
                    print(e)
                    speak("I am not able to send this email")

        elif 'send a mail' in query:
            try:
                    speak("What should I say?")
                    content = takeCommand()
                    speak("whome should i send")
                    to = takeCommand()
                    sendEmail(to, content)
                    speak("Email has been sent !")
            except Exception as e:
                    print(e)
                    speak("I am not able to send this email")

        elif 'how are you' in query:
            speak("I am fine, Thank you")
            speak("How are you, Sir")

        elif 'fine' in query or "good" in query:
            speak("It's good to know that your fine")

        elif "change my name to" in query:
            query = query.replace("change my name to", "")
```

```
        assname = query

elif "change name" in query:
        speak("What would you like to call me, Sir ")
        assname = takeCommand()
        speak("Thanks for naming me")

elif "what's your name" in query or "What is your name" in query:
        speak("My friends call me")
        speak(assname)
        print("My friends call me", assname)

elif 'exit' in query:
        speak("Thanks for giving me your time")
        exit()

elif "who made you" in query or "who created you" in query:
        speak("I have been created by Tathagat.")

elif 'joke' in query:
        speak(pyjokes.get_joke())

elif "calculate" in query:

        app_id = "K58RW2-PJY7LTPJ54"
        client = wolframalpha.Client(app_id)
        indx = query.lower().split().index()
        query = query.split()[indx + 1:]
        res = client.query(' '.join(query))
        answer = next(res.results).text
        print("The answer is " + answer)
        speak("The answer is " + answer)

elif 'search' in query or 'play' in query:

        query = query.replace("search", "")
        query = query.replace("play", "")
        webbrowser.open(query)

elif "who i am" in query:
        speak("If you talk then definately you are human.")

elif "why you came to world" in query:
```

```
                speak("Thanks to Tathagat. further It's a secret")

            elif 'power point presentation' in query:
                speak("opening Power Point presentation")
                power = r"C:\\Users\\GAURAV\\Desktop\\Minor
Project\\Presentation\\Voice Assistant.pptx"
                os.startfile(power)

            elif 'in love' in query:
                speak("It is 7th sense that destroy all other senses")

            elif "who are you" in query:
                speak("I am your virtual assistant created by Tathagat")

            elif 'reason for you' in query:
                speak("I was created as a smart mirror project by Mister
Tathagat ")

            elif 'change background' in query:
                ctypes.windll.user32.SystemParametersInfoW(20,

    0,

    "Location of wallpaper",

    0)
                speak("Background changed succesfully")

            elif 'open bluestack' in query:
                appli =
r"C:\\ProgramData\\BlueStacks\\Client\\Bluestacks.exe"
                os.startfile(appli)

            elif 'news' in query:

                try:
                    jsonObj =
urlopen('"https://newsapi.org//v1//articles?source=the-times-of-
india&sortBy=top&apiKey=\\timesofIndiaApikey\\"')
                    data = json.load(jsonObj)
                    i = 1
```

```
                        speak('here are some top news from the times of
india')
                        print('''=============== TIMES OF INDIA
============'''+ '\n')

                        for item in data['articles']:

                                print(str(i) + '. ' + item['title'] + '\n')
                                print(item['description'] + '\n')
                                speak(str(i) + '. ' + item['title'] + '\n')
                                i += 1
                    except Exception as e:

                        print(str(e))


            elif 'lock window' in query:
                        speak("locking the device")
                        ctypes.windll.user32.LockWorkStation()

            elif 'shutdown system' in query:
                        speak("Hold On a Sec ! Your system is on its way to
shut down")
                        subprocess.call('shutdown / p /f')

            elif 'empty recycle bin' in query:
                    winshell.recycle_bin().empty(confirm = False,
show_progress = False, sound = True)
                    speak("Recycle Bin Recycled")

            elif "don't listen" in query or "stop listening" in query:
                    speak("for how much time you want to stop jarvis from
listening commands")
                    a = int(takeCommand())
                    time.sleep(a)
                    print(a)

            elif "where is" in query:
                    query = query.replace("where is", "")
                    location = query
                    speak("User asked to Locate")
                    speak(location)
```

```
                webbrowser.open("https://www.google.nl / maps / place/" +
location + "")


        elif "restart" in query:
                subprocess.call(["shutdown", "/r"])

        elif "hibernate" in query or "sleep" in query:
                speak("Hibernating")
                subprocess.call("shutdown / h")

        elif "log off" in query or "sign out" in query:
                speak("Make sure all the application are closed before sign-
out")
                time.sleep(5)
                subprocess.call(["shutdown", "/l"])

        elif "write a note" in query:
                speak("What should i write, sir")
                note = takeCommand()
                file = open('jarvis.txt', 'w')
                speak("Sir, Should i include date and time")
                snfm = takeCommand()
                if 'yes' in snfm or 'sure' in snfm:
                        strTime =
datetime.datetime.now().strftime("%H:%M:%S")
                        file.write(strTime)
                        file.write(" :- ")
                        file.write(note)
                else:
                        file.write(note)

        elif "show note" in query:
                speak("Showing Notes")
                file = open("jarvis.txt", "r")
                print(file.read())
                speak(file.read(6))


        # NPPR9-FWDCX-D2C8J-H872K-2YT43
        elif "jarvis" in query:

                wishMe()
```

```
                speak("Jarvis 1 point o in your service Mister")
                speak(assname)

        elif "weather" in query:

                # Google Open weather website
                # to get API of Open weather
                api_key = "Api key"
                base_url =
"http://api.openweathermap.org//data//2.5//weather?"
                speak(" City name ")
                print("City name : ")
                city_name = takeCommand()
                complete_url = base_url + "appid =" + api_key + "&q =" +
city_name
                response = requests.get(complete_url)
                x = response.json()

                if x["cod"] != "404":
                        y = x["main"]
                        current_temperature = y["temp"]
                        current_pressure = y["pressure"]
                        current_humidiy = y["humidity"]
                        z = x["weather"]
                        weather_description = z[0]["description"]
                        print(" Temperature (in kelvin unit) = "
+str(current_temperature)+"\n atmospheric pressure (in hPa unit)
="+str(current_pressure) +"\n humidity (in percentage) = "
+str(current_humidiy) +"\n description = " +str(weather_description))

                else:
                        speak(" City Not Found ")

        elif "wikipedia" in query:
                webbrowser.open("wikipedia.com")

        elif "Good Morning" in query:
                speak("A warm" +query)
                speak("How are you Mister")
                speak(assname)

        # most asked question from google Assistant
        elif "will you be my gf" in query or "will you be my bf" in query:
```

```
            speak("I'm not sure about, may be you should give me some
time")

        elif "how are you" in query:
            speak("I'm fine, glad you me that")

        elif "i love you" in query:
            speak("It's hard to understand")

        # elif "" in query:
            # Command go here
            # For adding more commands
```

APPENDIX 2

Facerecognition.py

```
import os
from ecapture import ecapture as ec
import face_recognition
image = ec.delay_imcapture(0, "image", "first.jpg", 2)
image_to_be_matched = face_recognition.load_image_file('first.jpg')
image_encoded =
face_recognition.face_encodings(image_to_be_matched)[0]
current_image = face_recognition.load_image_file('user1.jpg')
current_image_encoded =
face_recognition.face_encodings(current_image)[0]
result =
face_recognition.compare_faces([image_encoded],current_image_encoded)
if result[0] == True:
   os.system('voiceassistance.py')
else:
   print ("try again")
```

APPENDIX 3

Gesturecontrol.py

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
 TRIG = 7
 ECHO = 12
 TRIG2 = 35
 ECHO2 = 38
  MIN_DISTANCE = 15
 TARGET_HOLD_COUNT = 8
 GPIO.setup(TRIG,GPIO.OUT)
 GPIO.output(TRIG,0)
 GPIO.setup(ECHO,GPIO.IN)
 GPIO.setup(TRIG2,GPIO.OUT)
 GPIO.output(TRIG2,0)
 GPIO.setup(ECHO2,GPIO.IN)
 time.sleep(0.1)
 prevSensorL = False
 prevSensorR = False
 holdCountL = 0
 holdCountR = 0
 lastInteractionTime = 0
 print ("Starting gesture recognition")
 try:   # here you put your main loop or block of code
while True:
   #print "Starting measurement"
 GPIO.output(TRIG,1)
 time.sleep(0.00001)
 GPIO.output(TRIG,0)
 while GPIO.input(ECHO) == 0:
 pass
start = time.time()
while GPIO.input(ECHO) == 1:
pass
stop = time.time()
GPIO.output(TRIG2,1)
time.sleep(0.00001)
GPIO.output(TRIG2,0)
while GPIO.input(ECHO2) == 0:
pass
start2 = time.time()
while GPIO.input(ECHO2) == 1:
pass
```

```
stop2 = time.time()
interaction = False
distance = (stop - start) * 17000
distance2 = (stop2 - start2) * 17000
sensorR = distance < MIN_DISTANCE
sensorL = distance2 < MIN_DISTANCE
#if prevSensorL and sensorR:
#print "Swipe right"
#if prevSensorR and sensorL:
#print "Swipe left"
if prevSensorL and sensorL:
holdCountL = holdCountL + 1
if holdCountL < TARGET_HOLD_COUNT:
print ("0")
if prevSensorR and sensorR:
holdCountR = holdCountR + 1
if holdCountR < TARGET_HOLD_COUNT:
print ("1")
if holdCountL >= TARGET_HOLD_COUNT:
holdCountL = 0
print ("10")
if holdCountR >= TARGET_HOLD_COUNT:
holdCountR = 0
print ("11")
interaction = sensorL or sensorR
prevSensorL = sensorL
prevSensorR = sensorR
if interaction:
lastInteractionTime = time.time()
elif time.time() - lastInteractionTime > 1:
#print "Resetting"
prevSensorL = False
prevSensorR = False
holdCountL = 0
holdCountR = 0
lastInteractionTime = time.time()
time.sleep(0.1)
except KeyboardInterrupt:
# here you put any code you want to run before the program
# exits when you press CTRL+C
print ("exiting")  except:
# this catches ALL other exceptions including errors.
# You won't get any error messages for debugging
```

```
# so only use it once your code is working
print ("Other error or exception occurred!")
finally:   GPIO.cleanup() # this ensures a clean exit
```