

# PIXEL, A VIRTUAL ASSISTANT WITH FACE RECOGNITION

Research Paper

**Institiúid Teicneolaíochta Cheatharlach**



**INSTITUTE of  
TECHNOLOGY  
CARLOW**

**At the Heart of South Leinster**

Theodra Tataru

C00231174

Supervisor: Joseph Kehoe  
Institute of Technology Carlow

30th of April, 2021

# Declaration on plagiarism

<i>Student</i>	Theodora Tataru – C00231174
<i>Tutor</i>	Joseph Kehoe
<i>Institution</i>	Institute of Technology Carlow
<i>Assignment Title</i>	PIXEL, A VIRTUAL ASSISTANT WITH FACE RECOGNITION
<i>Submission Date</i>	30 <sup>th</sup> April 2021

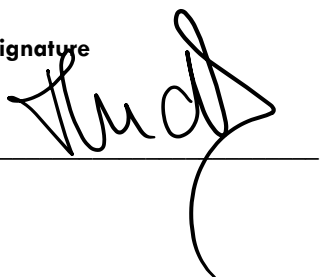
I declare that this research project titled “PIXEL, A VIRTUAL ASSISTANT WITH FACE RECOGNITION” has been written by me under the supervision of Joseph Kehoe.

This document was not presented in any previous research papers for the award of a bachelor's degree to the best of my knowledge. The work is entirely mine, and I accept full responsibility for any errors that might be found in this report. At the same time, the reference to publish materials had been duly acknowledged.

I have provided a complete table of references for all works and sources used in the preparation of this document.

I understand that failure to conform with the Institute's regulations governing plagiarism represents a serious offense.

Signature



---

Date:

30<sup>th</sup> April 2021

---

## ABSTRACT

This document is focused on the technological tools needed in the development of a virtual assistant dedicated to a broader segment of people, including people with their hearing impaired and people with poor computer skills.

The core of the project is a virtual assistant that is capable of receiving verbal instructions, process them, and return useful information. The distinguishing factor of this virtual assistant is the ability to recognize a user using facial recognition, and as an output, the returned information is displayed on a mirror surface.

The virtual assistant developed for this project does not listen continuously as similar products available on the market; the smart speaker is dormant until it detects a face, case in which it starts listening.

This is achieved by using a Raspberry Pi as the base of the project, where all the functionalities are implemented. Several other sensors and hardware modules are encapsulated among the Raspberry Pi into a wooden frame with a mirror.

The output of the virtual assistant is displayed on a screen installed behind the two-way acrylic mirror giving the user the illusion that the text appears on the surface of the mirror.

The software intended for use in this project is based on research in Artificial Intelligence, computer vision, facial recognition, servos movement, open-source virtual assistants, text-to-speech, and speech-to-text.

For this goal to be achieved, several technologies were researched, tested, and documented in this report.

## TABLE OF CONTENTS

Abstract.....	2
Table of Figures.....	5
Introduction.....	7
Motivation.....	8
Virtual assistants.....	9
Similar products.....	11
Marketing analytics.....	13
People with Hearing deficiency.....	13
Computer literacy.....	14
Research method.....	16
Hardware.....	17
Raspberry Pi 4 Model B.....	17
Pan-Tilt Hat Pimoroni.....	18
Camera with Infrared.....	18
Screen.....	19
GSM/GPRS/GNSS/Bluetooth HAT.....	21
Speakers.....	23
Software.....	24
Python.....	24
Python and Raspberry Pi.....	25
Artificial Intelligence.....	26
Machine learning.....	26
Natural language processing.....	28
Threading and multiprocessing.....	29
GIL.....	29
Threading.....	29
Multiprocessing.....	29
Open Source Computer Vision Library.....	31
Facial recognition.....	32
Dlib.....	33
Testing Raspberry Pi for Face Recognition with OpenCV and DLib.....	34
LBPH.....	36
Testing Raspberry Pi for Face Recognition with OpenCV and LBPH.....	36
OpenCV and face tracking.....	37
Proportional-Integral-Derivative controller (PID).....	37
Multiprocessing in Python.....	38
PID Controller and Multiprocessing used on Raspberry Pi Servos.....	39
Testing the Raspberry Pi Pantilt Hat using PID Controller.....	41

Testing the GSM/GPRS/GNSS HAT.....	43
Testing GPS module via serial console.....	43
Send SMS.....	46
Receive SMS.....	47
Perform a call.....	48
GPS.....	49
Open-source voice assistants.....	50
Jasper AI.....	51
Mycroft.....	54
Testing Mycroft.....	55
Jarvis AI.....	57
Text to speech.....	59
Pytttsx3.....	59
gTTS.....	60
Other TTS libraries and open-projects.....	60
Speech to text.....	62
Hidden Markov Model.....	63
Keywords.....	64
Magic/Smart mirror.....	65
TkInter.....	68
Threading.....	68
Multiprocessing.....	68
Conclusion.....	69
Bibliography.....	70

## TABLE OF FIGURES

Figure 1 „How a virtual Assistant Works”	9
Figure 2 „Inner working of Virtual Assistants”	10
Figure 3 “Voice Assistant Timeline”	11
Figure 4 „ Disability-adjusted life year for hearing loss”	13
Figure 5 „Tap to Alexa”	13
Figure 6 "Computer Literate"	14
Figure 7 "Digital skill levels among people aged between 65 and 74 (2017)"	14
Figure 8 „Facial Recognition”	15
Figure 9 "Love for Raspberry Pi"	17
Figure 10 “Raspberry Pi 4”	17
Figure 11 "Pan Tilt Hat"	18
Figure 12 "Infrared Camera"	18
Figure 14 "Night time"	19
Figure 13 "Day time"	19
Figure 15 "Raspberry Pi – 5 inches display"	19
Figure 16 "Raspberry Pi - 10.1 inches display"	19
Figure 17 "Raspberry Pi - 10.1 inches display - back"	20
Figure 18 "Display under a two-way mirror"	20
Figure 19 "Display physical installation"	20
Figure 20 "GSM, GPRS, GNSS and Bluetooth"	21
Figure 21 "Front view GSM hat"	21
Figure 22 "Back view GSM hat"	21
Figure 23 "Frequency band"	22
Figure 24 "Hi-Fi Sound Card HAT"	23
Figure 25 "The Zen of Python"	24
Figure 26 “Artificial Intelligence Branches”	26
Figure 27 “Machine Learning Techniques ”	27
Figure 28 "Multiprocessing vs. Multithreading"	30
Figure 29 "Pixel representation of an image"	31
Figure 30 „68(x,y) facial landmarks”	33
Figure 32 "Face Mapping"	34
Figure 31 "Face Recognition"	34
Figure 33 „Unknown person”	34
Figure 34 "Recognizing faces with Raspberry Pi "	35
Figure 35 "Facial recognition with LBPH Face Recognizer	36
Figure 36 "Facial recognition with LBPH Face Recognizer"	36
Figure 37 “PID Controller”	38
Figure 38 “Calculations for Proportional,-Integral-Derivative Controller”	39
Figure 39 “Video frame coordinates according to PanTilt Hat”	41
Figure 40 "PID Controller and Face Centering"	41
Figure 41 „GPS module status”	43
Figure 42 „Received messages from GPS module”	46
Figure 43 „Received messages – GPS module”	47
Figure 44 „Sent messages from phone to GPS module”	47
Figure 45 "Receiving a call from GPS module"	48
Figure 46 "GPS output"	49
Figure 47 "JasperAI requirements"	51
Figure 48 "Mycroft - Apache License"	54
Figure 49 “Picroft operating system”	55
Figure 50 “Mycroft debugging”	56
Figure 51 "Game Recording Microphone"	56
Figure 52 "USB Microphone"	56

Figure 53 "Microphone test – Jarvis AI" ..... 58  
Figure 54 "Testing TTS pyttsx3" ..... 59  
Figure 55 "Testing gTTS" ..... 60  
Figure 56 "English Phonemes" ..... 62  
Figure 57 "Evaluating Phonemes" ..... 62  
Figure 58 "Top 25 - keywords for smart speakers" ..... 64  
Figure 59 "Magic Mirror - Fully Installed" ..... 65  
Figure 60 "Magic Mirror - Languages Library" ..... 66  
Figure 61 "Magic Mirror - Home Page" ..... 67  
Figure 62 "Magic Mirror - Under the frame" ..... 67  
Figure 63 "TKinter GUI" ..... 68

## INTRODUCTION

Speech and the fact that human beings are communicating through voice is the basis of our society. According to the World Health Organization, around 466 million people suffer from a form of hearing disability (WHO, 2020). As this human feature is one of the pillars that sustain our intelligent society, the 5% of the world is suffering from a form of hearing insufficiency, always has a problem interacting with the community and with some aspects of technology in the modern days.

Also, people that do not possess strong computer skills, like the elderly population, might find it difficult to understand how to interact with a digital assistant.

Virtual assistants such as Alexa, Siri, or Google Assistants, interact with people via voice to perform actions for them. For older people or people who do not feel connected with technology, this should feel like the tech world's smoothest interaction. Unfortunately, for people with their hearing impairment and people with poor digital skills, virtual assistants are another significant impediment.

The purpose of the project is to develop a virtual assistant dedicated to a broader range of people. The term „broader range of people” refers to the general crowd plus the people with hearing disabilities and people with weak computer skills.

The idea is to create a virtual assistant that uses a camera to recognize the person that it interacts with and uses a screen to display in text format what it communicates back to the user.

This report was created to document all the researched technologies, software solutions, hardware details, and usage to facilitate the virtual assistants' development.

To create this new model of a virtual assistant, the following software and hardware technologies were researched :

- Raspberry Pi
  - Camera, microphones, and speakers
  - Pan Tilt Kit
  - Display
  - Smart Mirror
  - GSM/GPRS/GNSS/Bluetooth HAT
- Facial recognition
  - Python
  - OpenCV
  - Dlib
  - Deep learning
  - Artificial Intelligence
- Virtual assistants
  - Jasper
  - Mycroft
  - Jarvis
- Text to speech and speech to text
  - Open-source, free voices
- Backup, Version Controls and Environments
  - Git and Git Hub
  - Win32DiskImager (backup OS)
  - Virtualenv (python virtual environment)



A virtual assistant device is capable of interacting with its users by voice. The assistant is capable of following different instructions, such as :

- Identify the user by voice recognition
- Play music, podcasts, and audiobooks
- Make to-do lists
- Setting alarms and calendar appointments
- Provide weather and real-time information
- Control smart devices: light bulbs, central heating, and others (Wikipedia, 2020).

Virtual assistants are a fantastic tool to control your home and better manage your time. However, as mentioned before, it mainly interacts with its user via voice recognition and/or a mobile app. This kind of interaction excludes a segment of people from purchasing and utilize this kind of device: people with hearing disabilities and computer illiterate.

The idea of this project is to bring more functionality to a virtual assistant so that people with hearing disabilities and people lacking technological literacy can control the device, and hopefully, open windows for developers to deepen into this strategy and create functionalities for a broader range of disabilities such as deaf-mute individuals.

There is a virtual assistant on the market accessible for the people who can't hear what Alexa Echo has to say, and it is called „Tap to Alexa”. The device was designed with a touchscreen that needs to be tapped by the user to interact with the device. To instruct Alexa what to do next, the user selects an action from the screen by tapping on it (Crist, 2018).

Computer illiterates such as the elderly of society are also excluded from virtual assistants' target market because of their limited knowledge of computers. A device designed differently could positively impact their lives.

This project's approach is different from „Tap to Alexa”, having at its core face recognition equivalent to voice recognition from the traditional virtual assistant. This project aims to design a virtual assistant that activates when it detects faces and displays the requested information on a display hidden under a two-way mirror.

This report documented all the research done regarding the hardware, software, and any other tools or paths needed to develop the Virtual Assistant “Pixel”.

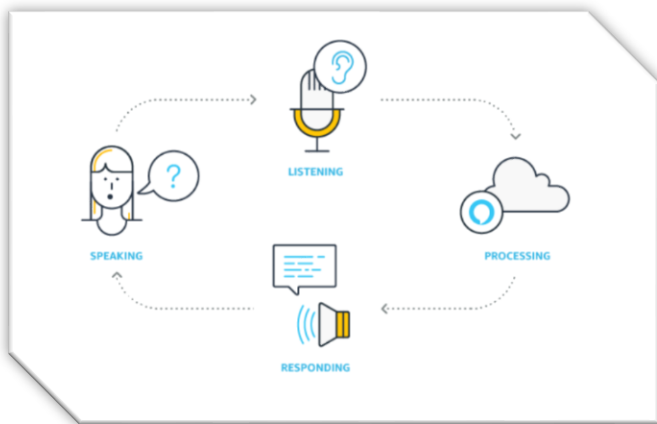
## VIRTUAL ASSISTANTS

A virtual assistant, also called a digital assistant or smart speaker, understands voice commands (natural language) and completes tasks for the user, as seen in **Figure 1** (Bridget Botelho, Margare Rouse, 2020).

The scope of the virtual assistant/smart speaker is to perform some tasks of a personal secretary such as:

- Reading emails out loud
- Looking up contacts
- Scheduling calendar appointments and made do-to lists
- Provide necessary information that would traditionally require a browser search: weather, news, and others. (Bridget Botelho, Margare Rouse, 2020).
- More modern virtual assistants can control a smart home that involves controlling lights, sockets, central heating, etc.

The technologies used to develop a virtual assistant require immense data to be fed to the Artificial Intelligence platforms, text to speech and speech to text, voice recognition, machine learning, and others (Bridget Botelho, Margare Rouse, 2020).



**Figure 1** „How a virtual Assistant Works”

Source (MIT, 2020)

Possessing a virtual assistant, people interact with it fundamentally in day-by-day life: asking questions and getting back meaningful answers. For these simple tasks to perform, the virtual assistant must understand the words, their meaning, search for an answer and then deliver it to the user. There is a need for powerful computers, artificial neural networks, machine learning, and advanced maths (Verizon, 2020).

Neural networks used to simulate brain function are based on advanced mathematics. Thus, the neurons' connections (which are numbers) in the system are interconnected into a complex network (Verizon, 2020).

As seen in **Figure 2**:

1. The virtual assistant awakens using a keyword (Hi Google, Hi Bixby, Alexa, and others)  
The virtual assistant is actually listening all the time, even when it is not interacting with the user, and when the keyword is heard, it begins to record what follows after the keyword
2. After the wake word is used, the virtual assistant records the user's command and then sends the recording to the server. The server transforms the speech into text
3. The meaning of the text is extracted with machine learning and natural language processing

4. As the meaning of the command was extracted from the text, the virtual assistant executes the user's command in different ways, depending on the meaning
5. After the command was executed, the virtual assistant delivers the answer back to the user (Verizon, 2020)

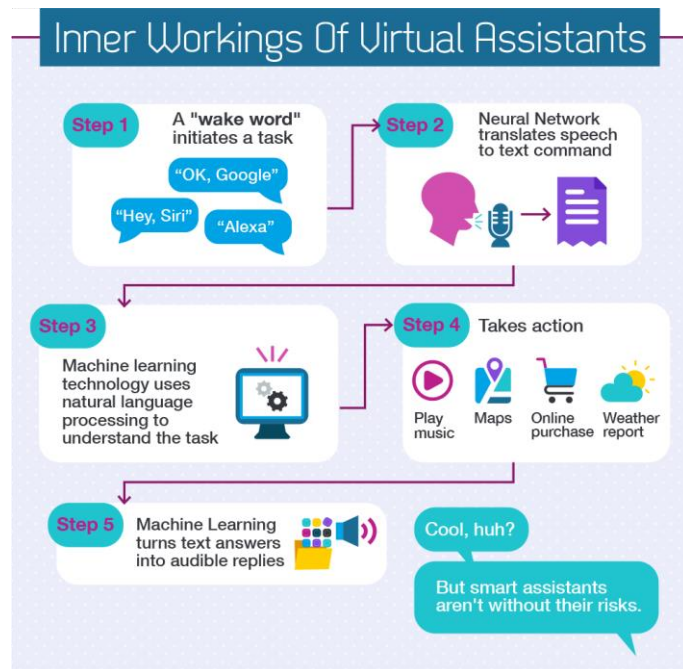
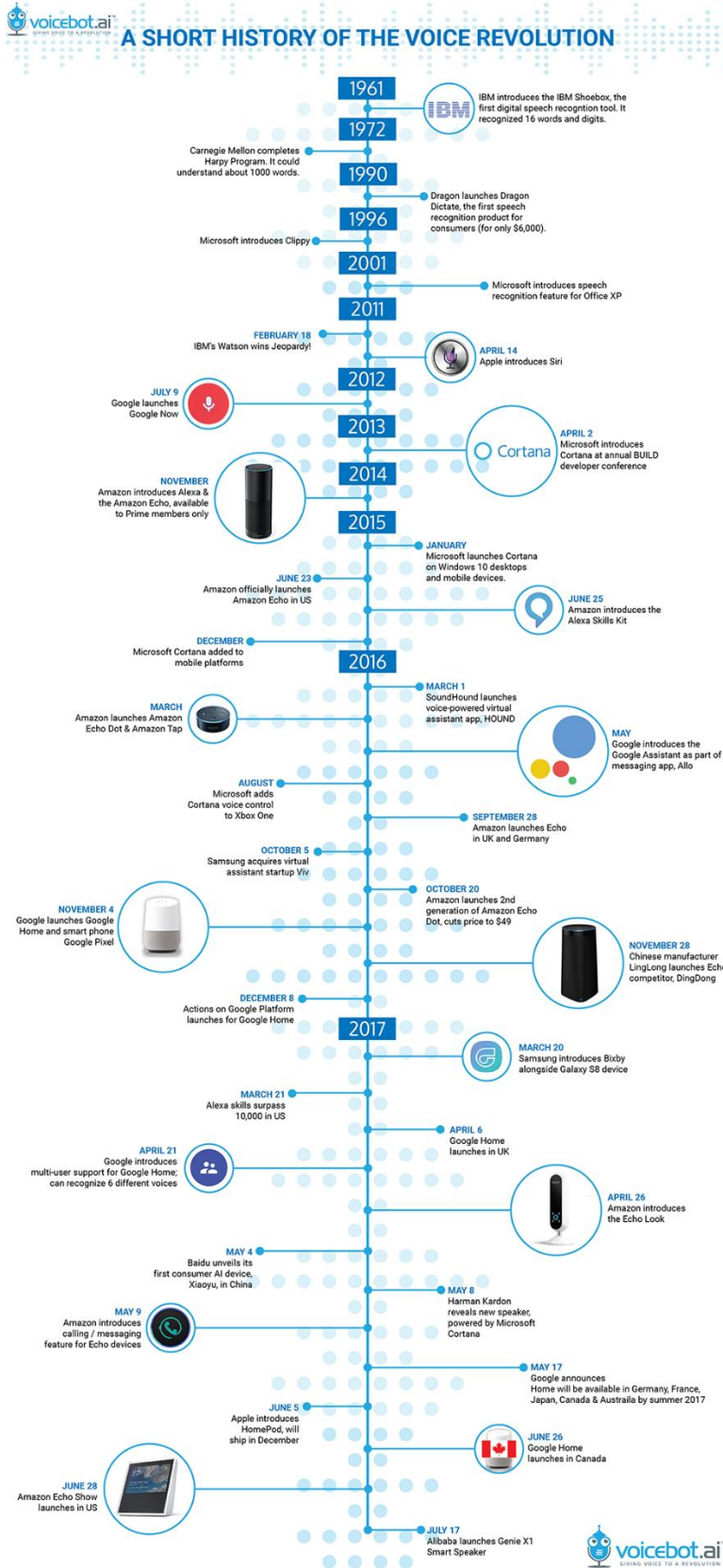


Figure 2 „Inner working of Virtual Assistants”

Source: (Verizon, 2020)

SIMILAR PRODUCTS



The era of voice assistants started in 1961 when IBM first introduced a digital assistant. The assistant was able to recognize 16 words and 9 digits (Mutchler, 2017).

Clippy was a famous and comic digital assistant present in Microsoft Word, always ready to assist you with your paperwork. Clippy is remembered by almost everybody that possessed a computer around the 2000s. Even if it was not the perfect assistant, it showed the tech world how text is interpreted, tracked, and used to improve users' experience (Mutchler, 2017).

The modern era of voice assistants was introduced by Siri in 2011 and followed by Google Now and Microsoft's Cortana (Mutchler, 2017).

Today, the market has a broad range of virtual assistants. Most popular assistants are offered by massive tech companies such as Microsoft, Samsung, Amazon, Apple, Google, and others (Mutchler, 2017).

The timeline of the Virtual Assistants can be seen in **Figure 3**.

Figure 3 "Voice Assistant Timeline"  
Source: (Mutchler, 2017)

### 1. *Google Assistant*

The virtual assistant offered by Google may be used on mobile phones, TVs, watches, smart speakers, laptops, and even in cars. It interacts with the user by voice, and it connects with products such as Netflix, Spotify, Etsy, Discord, and others. It can also read the user's emails, messages and respond to them (Google, 2020).

For people with hearing disabilities, Google offers an app called "Sound Amplifier" that turns the phone into a hearing aid. Also, "Live Transcribe" is an app that translated speech to text in real-time (Locker, 2019).

### 2. *Cortana*

Microsoft's virtual assistant is called Cortana and supports the Microsoft 365 suite's productivity and the fundamental functionality of starting timers, set alarms, check the user's calendar, read and respond to emails, texts, and SMS.

Microsoft has various features for disabled people, such as "Narrator" and Magnifier for vision impaired, "Speech Recognition," which allows the navigation of the menus, dictating documents, and surfing the web by translating speech into actions (Otachi, 2019). These features are beneficial and target a broader part of the population, but they are not necessarily linked with Cortana.

### 3. *Amazon Alexa*

Alexa, one of the most famous virtual assistants, is designed and delivered by Amazon. Besides the basic functionalities of any virtual assistant, the digital assistant can control multiple other devices that are compatible with it. This action gives Alexa the popularity gained over the years (O'Boyle, 2020).

Alexa has an Echo Dot called "Tap to Alexa" dedicated primarily to hearing and speech impaired users. When "Tap to Alexa" needs to wake and pay attention, the user taps on the screen, and the screen populates with shortcuts to the most popular Alexa features. From its display, all traditional Alexa features can be accessed, even the smart home features (Tiltman, 2018).

### 4. *Siri*

Another very popular digital assistant is Siri, introduced by Apple in 2011. The virtual assistant can perform all the functionalities as other assistants to benefit a hands-free lifestyle for its users.

Siri also was the first intelligent virtual assistant (Garcia, 2020).

### 5. *Bixby*

Bixby is the Samsung alternative of Siri, and like its competitor, it comes pre-installed on all Samsung devices.

Bixby can identify landmarks and images captured by the camera, translate texts in multiple languages, and gives price ranges of products pointed to by the camera (Garcia, 2020).

### 6. *Mycroft*

The first open-source virtual assistant, Mycroft, runs anywhere, from desktops, cars to Raspberry Pi's. Mycroft answers curiosity questions, controls the user's home, and assists with its schedule and productivity.

Virtual assistants' list does not stop here; there are other voice assistants available, some popular and some not as much, and the list is increasing every day as their popularity is growing exponentially. Today, more than 4.2 billion digital voice assistants are being used on mobile phones and smart speakers, and by 2024 it is predicted that the number of assistants will exceed the number of people in the world (H. Tankovska, 2020).

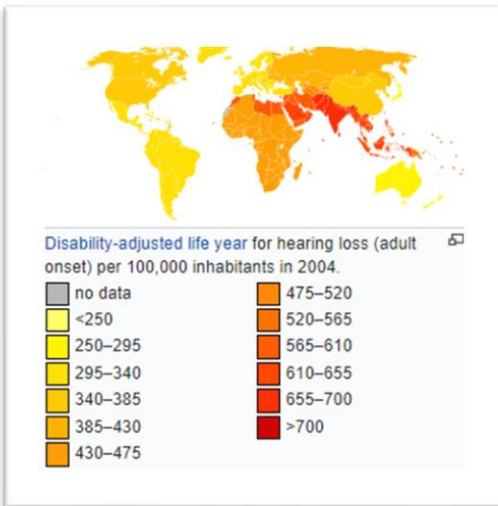
Most of the virtual assistants mentioned above come pre-installed on phones or designed into a speaker. Some speakers have an incorporated screen that displays information according to the user's request, but not all virtual assistant skills are correlated with the display; the graphical elements complement the voice interaction of the device itself (Alexa, 2020).

Globally, virtual assistants (also called smart speakers) shipments had increased by 200% in 2018. Their sales will increase to 30 billion dollars by 2024, putting the virtual assistant devices on the top-selling electronic products. 55% of households are expected to own a smart virtual assistant by 2022, as 34% of people who do not own at the moment such a device are interested in purchasing one (Andersen, 2019).

In 2020, there are about 4.2 billion virtual assistant devices used around the world. The forecast for 2024 is that the number of devices will rise to 8 billion units, which is higher than the world population at the moment, which is 7.5 billion (H. Tankovska, 2020).

PEOPLE WITH HEARING DEFICIENCY

At the moment, over 5% (466 million people) over the world suffer a form of hearing deficiency. Loss of hearing can vary at different levels: mild, moderate, severe, or profound and can affect one or both ears (WHO, 2020).



From these metrics, about 124 million people suffer from moderate to severe disability, of which 65 million are affected since childhood, and the severity increases with age (Wikipedia, 2020).

Figure 4 „Disability-adjusted life year for hearing loss”

Source (Wikipedia, 2020)



Figure 5 „Tap to Alexa”

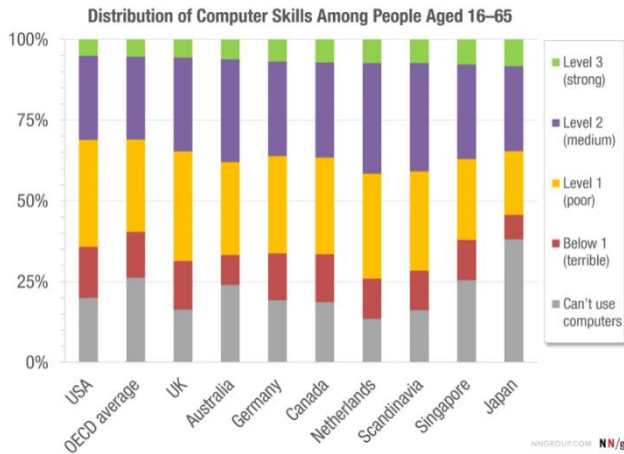
Source: (Saba, 2018)

All 466 million people of the world suffering from hearing deficiency cannot benefit from all the aspects of a virtual assistant for the fundamental reason as most virtual assistants are mainly based on voice recognition and voice commands. Of course, „Tap to Alexa” (Figure 5) is an option, but it simulates a mobile app displayed on a screen attached to a Virtual Assistant gadget, requiring the user to tap on the screen on different shortcuts to perform actions.

## COMPUTER LITERACY

In order for people to operate in today’s digital world, they need to possess foundations in computer skills that include the manipulation of the input and output devices such as mice, keyboards, screens, along with fundamental computer and internet concepts.

As determined by the “World Economic Forum”, the average metrics of computer literacy declare that 25% of individuals do not know how to use a computer, 45% rate poorly, and 30% rated as moderate to strong computer literate (Wikipedia, 2020).



As seen in **Figure 6**, around 25% of the population cannot use computers at all, while about 10% can handle computers terribly, followed by approximately 20% of the population that can utilize computers poorly. The remaining percentage is categorized by people who can use computers with medium to strong skills (Ravenscraft, 2016).

The first ~60% of the population needs assistance when purchasing a new phone, online plane tickets, installing new computer devices, or others. This segment of the population needs constant help living in today’s world as everything is strongly connected with technology.

**Figure 6 "Computer Literate"**

Source: (Ravenscraft, 2016)

**Table 3: Digital skill levels among people aged between 65 and 74 (2017)**

Percentage of people aged 65-74 who:	Ireland	Britain	Denmark	Finland	EU28
Have basic or above basic overall digital skills <sup>6</sup>	17	42	42	40	25

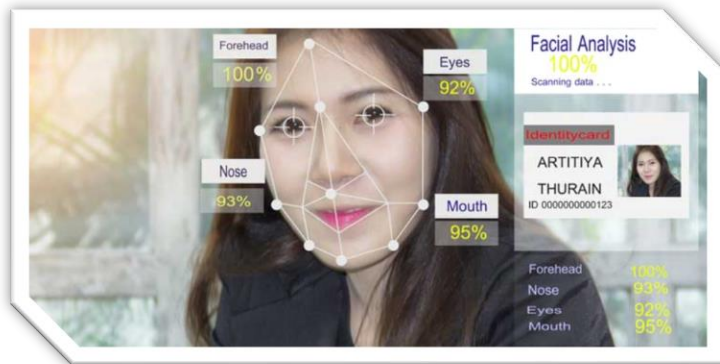
**Figure 7 "Digital skill levels among people aged between 65 and 74 (2017)"**

Source: (AgeAction, 2020)

As seen in **Figure 7**, the elderly population percentage possessing digital skills is significant, but at the same time, the remaining percentage may have never used any form of a computer.

This segment of the population is emphasized, as it is targeted as well by this project's goal. “Pixel” virtual assistant targets the elderly and computer illiterate population, as it could improve their life. “Pixel”, besides the main functionalities of a virtual assistant, would keep users engaged using the “Magic Mirror”, detailed later in this project.

Besides targeting the general population, computer illiterate and people with a hearing deficiency would benefit from “Pixel” virtual assistant. The new design implies the use of face recognition with a camera that focuses on the user’s face as it moves, as long as the face is in the camera’s range. The virtual assistant’s answer to the users’ request will be displayed on a screen hidden behind a two-way mirror, giving the magic effect of graphics appearing on a mirror’s surface.



This way, the user is not required to tap on the device to interact with it; it just needs to be within the camera range.

**Figure 8 „Facial Recognition”**

Source: (LEONARD, 2018)



## RESEARCH METHOD

The research involved to understand and choose the right technologies to achieve the goal of the project was performed using the following methodologies:

- Testing open-source projects
- Official Raspberry Pi website
- Conference Papers
- Testing the hardware
- Researching and testing IoT technologies
- Reading research papers

## RASPBERRY PI 4 MODEL B

The Raspberry Pi Model 4B is the latest product from the Raspberry Pi range of motherboards (Electronics, 2019).

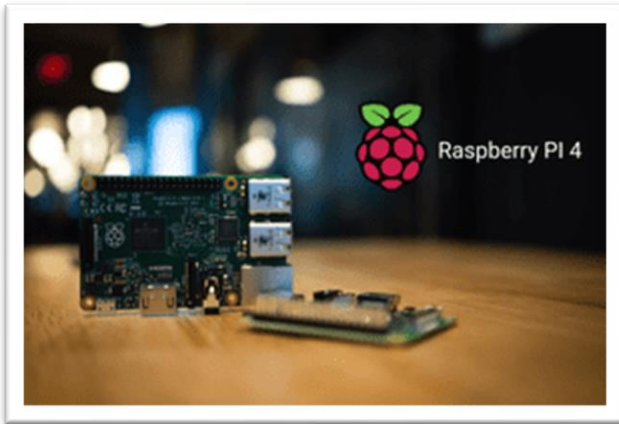


Figure 9 "Love for Raspberry Pi"

Source: (Serasinghe, 2020)

The version of Raspberry Pi chose is model 4 B+ with the following accessories (Electronics, 2019)

- 4GB of RAM
- 1.5 GHz quad-core Arm Cortex
- VideoCore VI graphics
- True Gigabit Ethernet
- 2 USBs 3.0
- 2 USBs 2.0
- 2 Micro HDMI
- 1 USB C
- 4kp60 HEVC decode

Raspberry Pies are not like the computers we are using in our life in day-by-day activities. The Raspberry Pi is a small motherboard the size of a Credit Card and is a cheaper version of a computer that comes naked, without a case (Heath, 2018).

A Raspberry Pi 4 can be seen in **Figure 10**.

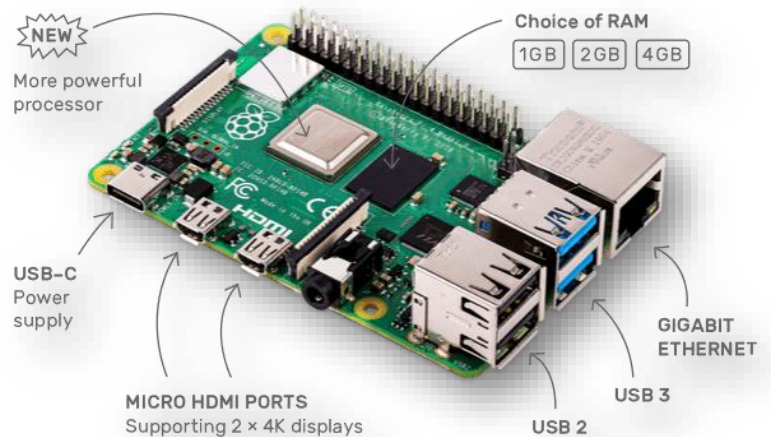


Figure 10 "Raspberry Pi 4"

Source: (Serasinghe, 2020)

The reason this model was chosen (Raspberry Pi 4 B) is that OpenCV (the package used for facial recognition) would need at least 1GB of RAM to run smoothly (Villán, 2019), and Raspberry Pi 4 is the only architecture that comes with more than 1 GB of RAM and a 64-bit processor (Electronics, 2020).

## PAN-TILT HAT PIMORONI

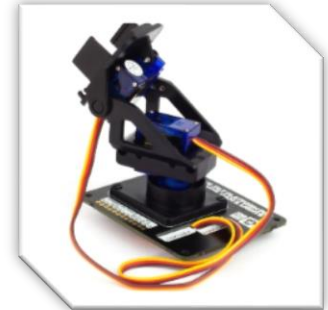
The Pan-Tilt Hat from Pimoroni is a kit dedicated to Raspberry Pi that allows the control of the camera movement, and it comes as a „hat” that mounts on top of the motherboard, as seen in **Figure 11** (Pimoroni, 2020).

The kit has two ranges of movements:

- PAN – for right and left movements with a range of 180 degrees
- TILT – for up and down movements with the same range of 180 degrees

The kit also comes with a LED drive that could serve as a light in dark environments, but it is not in the plan to use the LED as the camera used for this project has an INFRARED lens.

The initial plan was to use a Pan-Tilt HAT from WaveShare, but the experiment was unsuccessful as the camera was shaking dramatically when the Raspberry Pi was powered on. The reason for the undesired quivering was the servos. This failure led to the decision to try the Pan-Tilt Hat from Pimoroni, which performs smoothly.



**Figure 11 "Pan Tilt Hat"**  
Source: (Pimoroni, 2020)

## CAMERA WITH INFRARED

The camera intended to be used in this project is a MakerHawk Infrared Camera designed for Raspberry Pi. The reason behind this choice is the infrared lens that allows great vision during nighttime and daytime, in any light condition. The downside is that all images have a purple-ish filter-like colour when it displays images resulting from the camera's infrared lens (MakerHawk, 2020).



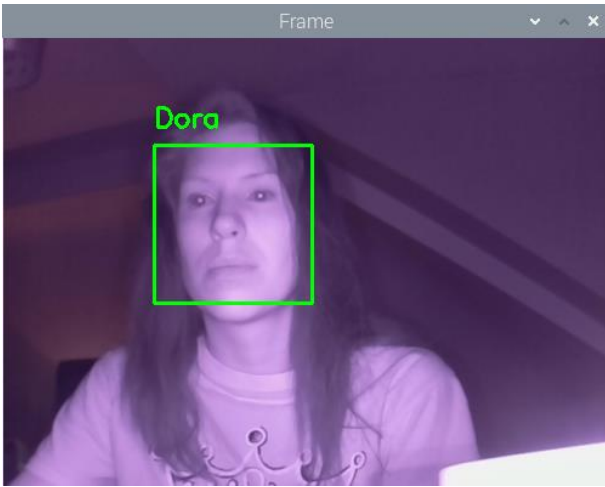
**Figure 12 "Infrared Camera"**  
Source: (MakerHawk, 2020)

The camera has the following properties

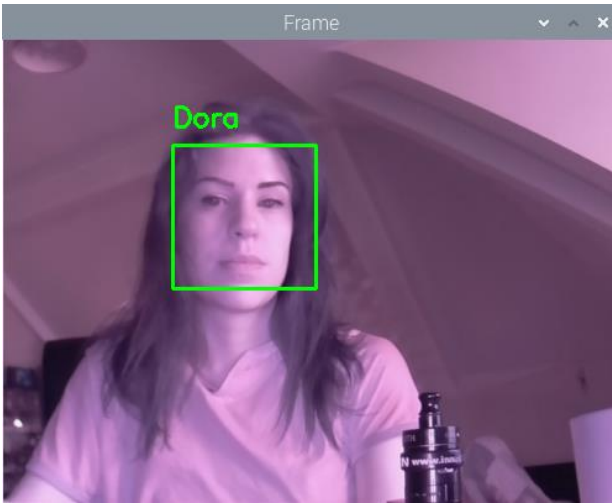
- 5 Megapixels
- Focal Length 3.6mm
- Light sensors
- Shooting angle 60 degrees
- 1080p optimum resolution
- Night vision

As can be seen in **Figure 13** and **Figure 14**, the camera is performing very well in different light conditions: night or day.

For the user experience, the purple color that covers the image is a big minus. However, for this project, the camera is used to recognize a user, and its broadcast is not displayed on the screen.



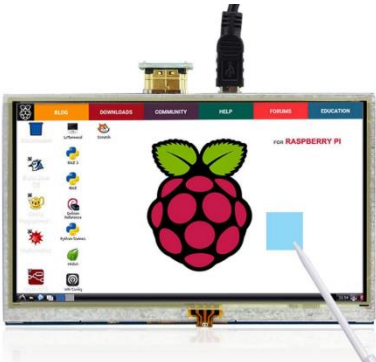
**Figure 14 "Night time"**  
Source: Theodora Tataru 2020



**Figure 13 "Day time"**  
Source: Theodora Tataru 2020

**SCREEN**

One of the screens tested for this project is a 5-inch touch screen with an HDMI input.



**Figure 15 "Raspberry Pi – 5 inches display"**  
Source: (Elecrow, 2020)

The screen will be used to display in text and graphics all the answers delivered by the virtual assistant after performing an action.

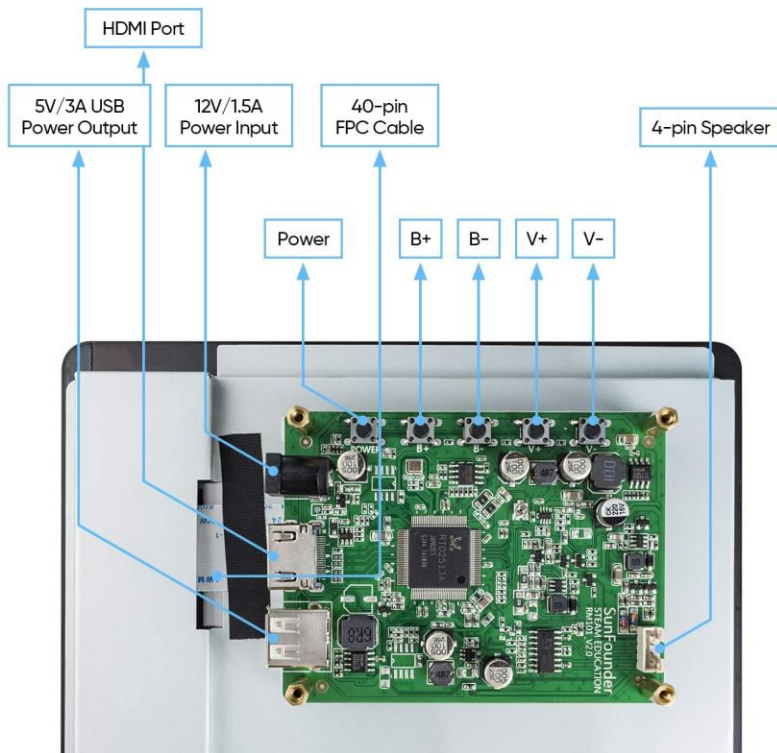
The screen has the following properties:

- 5-inch diagonal
- Resolution: 800x800
- Backlight control
- Touch screen
- Large viewing angle



**Figure 16 "Raspberry Pi - 10.1 inches display"**  
Source: (Amazon, 2020)

Another screen tested (**Figure 16**) for this project is a display with a diagonal of 10.1 inches. This choice is more suitable for the project; as the screen is wider and the text size can be increased. Besides the physical difference, this display also had a higher resolution of 1280x800.

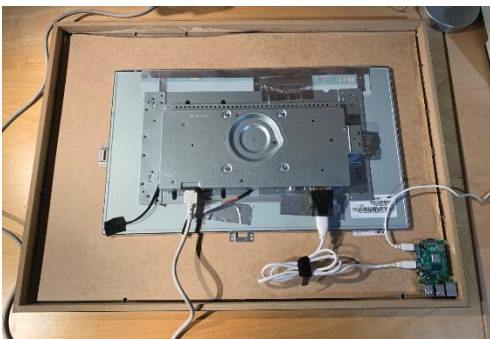


Also, at the back of the second display (**Figure 17**), the Raspberry Pi can be attached. This design benefits the project, as the Raspberry Pi will be encapsulated into a wooden frame. The hardware's resting position should be designed as tightly as possible, allowing the hardware to receive the proper ventilation.

**Figure 17 "Raspberry Pi - 10.1 inches display - back"**

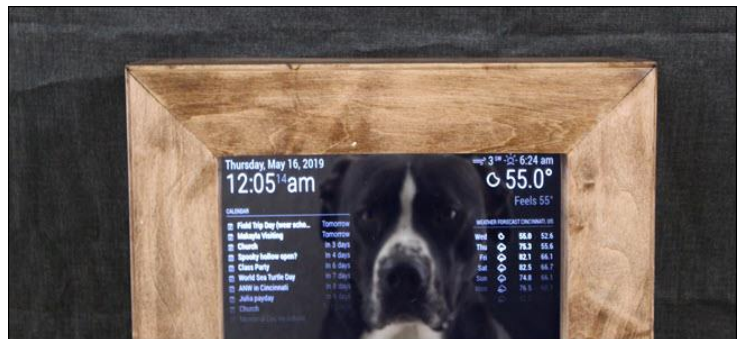
Source: (Amazon, 2020)

The screen will be positioned behind a two-way acrylic mirror, as seen in **Figure 18** and **Figure 19**.



**Figure 18 "Display under a two-way mirror"**

Source: (Hendrickson, 2019)



**Figure 19 "Display physical installation"**

Source: (TV, 2020)

## GSM/GPRS/GNSS/BLUETOOTH HAT



The Raspberry Pi equipped with GSM/GPRS/GNSS/BLUETOOTH HAT and a GSM SIM card inserted into the HAT enables multiple communication methods such as GSM, GPRS, GNSS, and Bluetooth. In simple words, it enables the Raspberry Pi to perform a phone call, send and receive text messages, connect to the internet using mobile data, get an accurate global position, and others (Waveshare, 2018).

Figure 20 "GSM, GPRS, GNSS and Bluetooth"

Source: (Amazon, 2021)

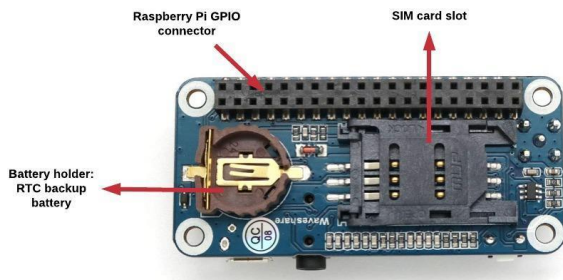


Figure 22 "Back view GSM hat"

Source: Theodora Tataru, 2021

### Figure 22:

- Battery holder**  
 Requires a CR1220 battery to keep the real clock (RTC) running when the board is switched off. This battery can keep the RTC running for up to 3 years without an external power supply.
- Raspberry Pi GPIO connector**  
 The connector enables the HAT to connect with the Raspberry Pi via GPIO pins.
- SIM card slot**  
 The Hat is network-free, giving the possibility to choose any SIM card network; therefore, the module was equipped with a SIM card that provides unlimited mobile data, unlimited phone calls, and texts.

### Figure 21:

- UART/USB selection switch** ensures ways of connection between Raspberry Pi and Hat:
  - A: control the SIM868 through USB TO UART
  - B: control the SIM868 through Raspberry Pi GPIO pins
  - C: access Raspberry Pi through USB TO UART
- USB TO UART interface**  
 Permits the GSM/GPRS/GNSS HAT to connect with the Raspberry Pi through the USB instead of the GPIO pins.

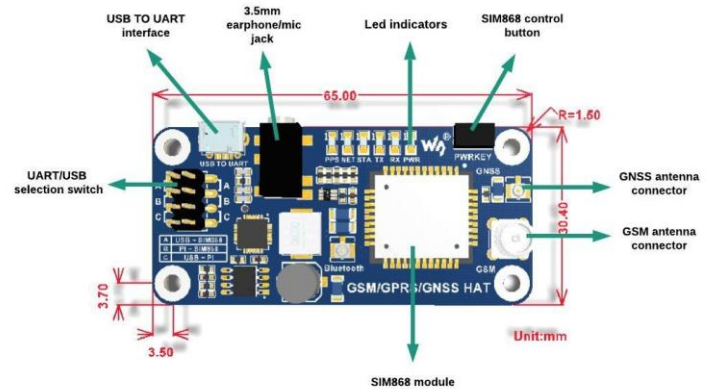


Figure 21 "Front view GSM hat"

Source Theodora Tataru. 2021

- **3.5mm earphone/mic jack**  
Allows the HAT to be equipped with a speaker/microphone via the jack port. This feature enables the system to act as a mobile phone performing calls, using the microphone as an input and the speakers as an output.
- **Led Indicators**  
The Hat is equipped with six led indicators. Each led represents a different functionality of the Hat, such as:
  1. GPS status indicator
  2. NET indicator: flashes fast when the module starts up and flashes slowly after GSM register succeed
  3. STA module working status indicator
  4. Two SIM868 UART indicators:
    - a. Tx - Transmission
    - b. Rx - Receiver
  5. Power indicator
- **SIM868 control button**  
The control button represents the start-up/shutdown of the HAT by simply pressing the button and hold for one second.
- **GNSS antenna connector**  
GNSS (Global Navigation Satellite System) enables the Hat to retrieve information from all global satellite positioning systems, receivers, and other corresponding antennas.  
This antenna receives radio signals transmitted on specific frequencies by satellites (Ltd, 2020).
- **GSM antenna connector**  
The GSM antenna transmits GSM signals at multiple frequencies such as 850, 900, 1800, 1900, and 2100MHz (Wellshow, 2021).

## Frequency band

Frequency	Application
824 ~ 849 MHz	GSM-850
872 ~ 950 MHz	E-GSM, R-GSM, TDMA, CDMA
890 ~ 950 MHz	GSM900
1710 ~ 1880 MHz	GSM 1800
1900 ~ 2200 MHz	UMTS, 3G, GPRS

Figure 23 "Frequency band"

Source: (Wellshow, 2021)

GSM is the most popular standard for mobile communication and stands for „Global System for Mobile Communication” (Wellshow, 2021).

- **SIM868 module**  
SIM868 module is described as a complete QuadBand GSM/GPRS which combines GNSS technology for satellite navigation. With an industry-standard interface and GNSS capability, it allows variable assets to be monitored in real-time at any location with signal coverage (Europe, 2021).

**Note:**

To perform a call using this hat, a set of headphones with a microphone is needed, with a Jack plug.



Figure 24 "Hi-Fi Sound Card HAT"  
Source: (Amazon, 2021)

The IBest WM8960 HI-FI Hat Audio Module compatible with the Raspberry Pi 4B+ were chosen as an audio output for this project for various reasons, including low power consumption, dual-channel speaker interface, direct drives speakers, and their size. (Amazon, 2021)

Specifications (Amazon, 2021)

- CODEC: WM8960
- Power supply: 5V
- Logic Voltage: 3.3V
- Control interface: I2C
- Audio interface: I2S
- DAC signal-noise ratio: 98dB
- ADC signal-noise ratio: 94dB
- Earphone driver: 40mW (16  $\Omega$  @3.3V)
- Speaker driver: 1W per channel (8  $\Omega$  BTL)

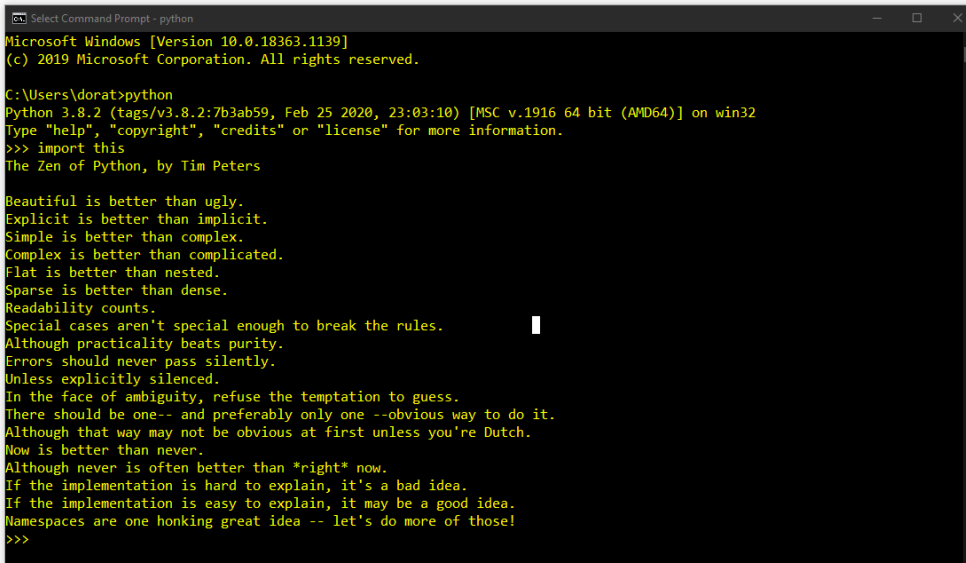
The display used for this project is equipped with an I2C. This functionality allowed the speakers to connect straight into the display without the need for the WM8960 HI-FI HAT. Therefore, the speakers were used directly through the display board.



## PYTHON

As the project implements the OpenCV library for real-time computer vision, the languages compatible with this library are Python, Java, and a few others, resulting in choosing Python as a language to implement the desired functionality as it is very popular with Raspberry Pi's for inexperienced developers with electronics.

Python is an interpreted, high-level programming language designed for general purpose use. It focuses on simplicity and readability through its extensive use of whitespace. Python emphasizes the developer's productivity at the cost of runtime performance by offering developers multiple high-level data structures and dynamic typing. (Harris, 2020).



```
Select Command Prompt - python
Microsoft Windows [Version 10.0.18363.1139]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\dorat>python
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

The Zen of Python by Tim Peters (**Figure 24**) is a poetic description of the values of Python and its community. With 19 aphorisms designed to guide developers on how to write their source code. From advising developers to write flat code instead of nesting to encouraging unity in operations by ensuring a singular agreed-upon method of achieving tasks, it's a short yet unique discovery for Python newcomers and veterans alike.

**Figure 25 "The Zen of Python"**

Source: Theodora Tataru 2020

One of the main reasons why Python was chosen for this project is that gaining experience in Python is a plus for any developer experience as it is so prevalent in so many industries (Kumar, 2020):

- Game Development
- Machine Learning and Artificial Intelligence
- Desktop GUI
- Audio and Video applications
- Embedded Applications
- Business Applications
- Web Development

Raspberry Pi supports many programming languages such as

- Scratch
- Python
- JavaScript
- Java
- C
- C++
- Erlang
- Others (Sean McManus, 2020)

On the other hand, the official documentation of Raspberry Pi ([raspberrypi.org](http://raspberrypi.org)) focused on Python and Scratch (PI, 2020).

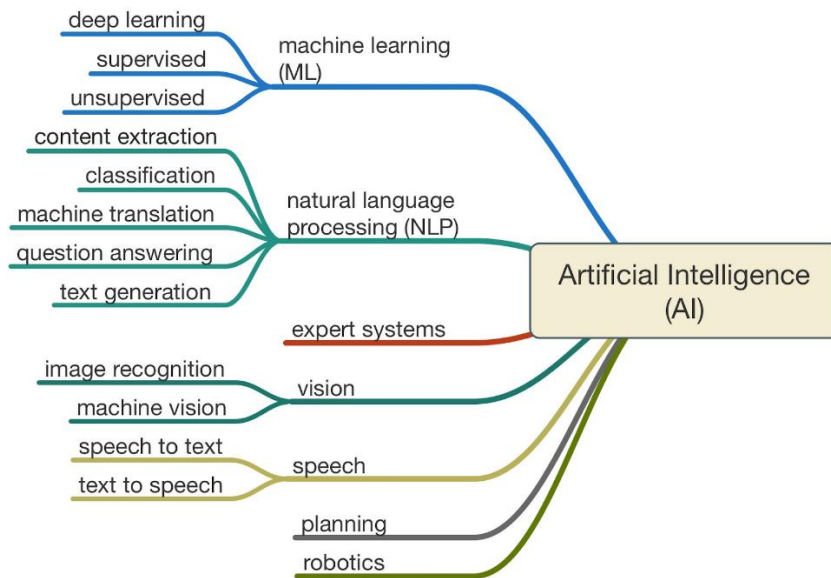
Java, C, and C++ can be implemented, but experimented developers choose these languages in electronics (Verma, 2016).

Python is commonly used with Raspberry Pi as it has a complete adoption in the community giving access to packages, tools, and frameworks, and has an easy syntax. The popularity of Python with Raspberry Pi comes from the fact that the motherboards are built for beginners to start their journey and controlling electronics in real-time (Verma, 2016)

## ARTIFICIAL INTELLIGENCE

The idea of Artificial Intelligence (AI) had been present for a long time. The ancient Greeks had mythologies about intelligent robots, but the beginning of modern AI started in Dartmouth College, Hanover, New Hampshire, in 1956. At that time, this idea had a different form from what is known today; it tried to copy human behaviour (Warwick, 2012).

Today, artificial brains are built, not designed to mimic humans, but being intelligent independently with the possibility of becoming bigger, faster, and better. Behind this artificial intelligence scene are multiple concepts and technologies: natural language processing, computer vision, algorithms, sensors, machine learning algorithms, and others (Taulli, 2019). At the core of Artificial Intelligence is the answer to Alan Turing's question: "Can a machine's ability exhibit intelligent behaviour equivalent to, or indistinguishable from, that of a human?" (Wikipedia, 2020).



These days, AI is present everywhere around us, from applications like Uber to email communications, social media, web surfing, virtual assistants, self-driving cars, and others. The AI changes the way humans interact with data and how decisions are made (Otte, 2020).

**Figure 26 "Artificial Intelligence Branches"**

Source: (Medmain, 2018)

As seen in **Figure 25**, Artificial Intelligence has various branches, and their numbers are increasing rapidly as more research is performed on this ramification of computing.

In this document, several branches mentioned above are detailed later in the report, as the "Pixel" virtual assistant's development depends on these technologies. Following, briefly, a description is given for some of the learning branches.

---

## MACHINE LEARNING

Machine learning is the programming action to optimize performance using sample data and past experience. The model learns using the training data as guidance. In other words, Machine Learning teaches computers to perform tasks, as humans: learning from experience (Alpaydin, 2020).

This learning process uses algorithms to compute methods dedicated to the process of learning. The learning process is based on two techniques: supervised and unsupervised learning:

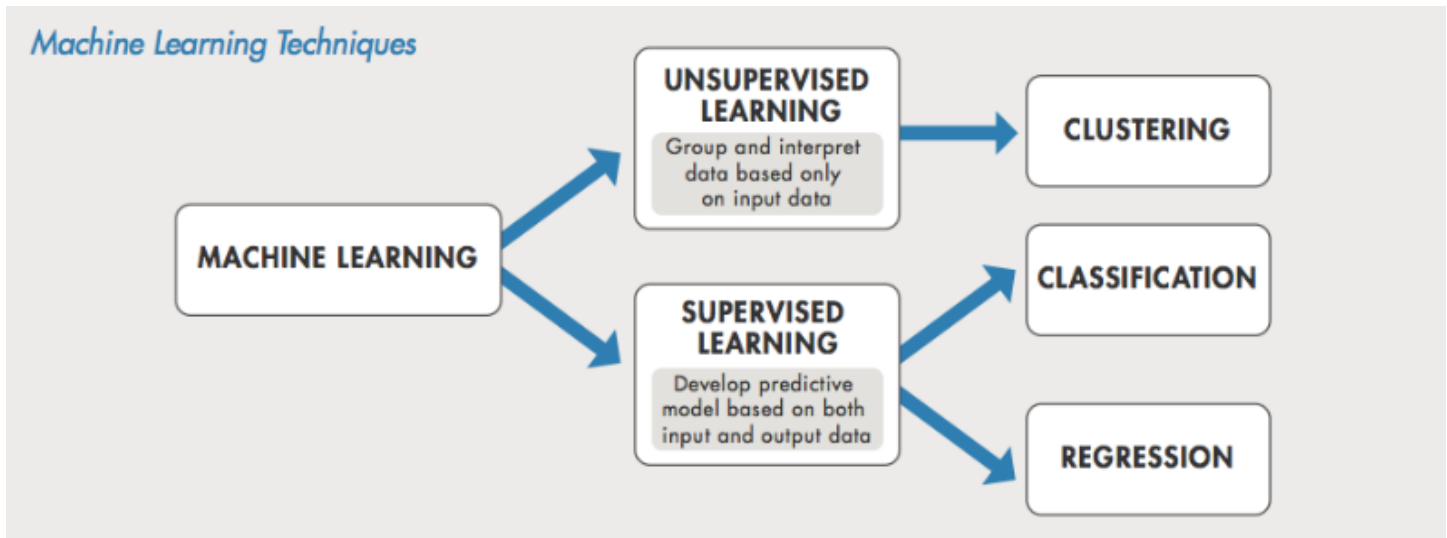


Figure 27 “Machine Learning Techniques ”

Source: (Mathlab, 2020)

### Supervised learning

The supervised learning algorithm takes a classified set of data and trains the model to produce predictions based on the sample data. The data fed to the model describes explicitly what the model should focus on. This method uses techniques to develop predictive models:

#### 1. Classification

Classification models sort input data into categories. This model is usually used in image classification and speech recognition. When working with classification problems, the first step is to determine if the problem is binary or multiple (Agarwal, 2013).

##### Algorithms:

- Logistic Regression can predict a binary response and is used when a single linear boundary can separate data
- K Nearest Neighbour generates its predictions based on the nearest neighbours’ information, assuming that those neighbours are similar. Euclidian distance metrics and others are used to find the nearest neighbour.
- The Neural Network model is inspired by the human brain, and its architecture contains neurons highly interconnected. The network is trained by modifying the strength of the connections between neurons so that when input is given, it maps to the correct response.
- The Decision Trees model predicts by following decision trees from the root node down to a leaf node. Each node consists of branching decisions, and the value of weights is determined during the training phase.
- Naïve Bayes is a classifier that, at its base, assumes that the presence of a feature is not correlated with other features in the same class. The predictions are determined by the highest probability of a feature belonging to a class (Mathlab, 2020).

Other algorithms exist at the moment, and new algorithms are developed and improved every day.

#### 2. Regression

Regression algorithms are designed to predict continuous responses such as stock prices, handwriting, and others.

##### Algorithms:

- Linear Regression is a statistical technique used to predict continuous response like a linear function.
- Nonlinear Regression is as well a statistical model technique that describes nonlinear data, that helps to describe nonlinear relations in experimental data
- Regression Trees are similar to Decision Trees in classification but are modified to predict continuous responses (Mathlab, 2020).

## Unsupervised Learning

Unsupervised learning is dedicated to finding patterns in data. The model is not fed with labelled data points; instead, the model organizes the data on its own based on the features of inserted data (Alpaydin, 2020).

Clustering is the most popular unsupervised model and is designed to find natural grouping in data. In other words, it focuses on dividing groups with similar features and assigns them into clusters (Mathlab, 2020).

The best way to choose a model for machine learning is by trial and error, keeping in mind what prediction is expected, the datasets size, and how the predictions will be used. Machine learning models are used when the task is extremely complex and needs to handle an immense amount of data (Mathlab, 2020).

---

## NATURAL LANGUAGE PROCESSING

Natural Language Processing (abbreviated NLP) is a branch of Artificial Intelligence that enables machines to read text and understand human language meaning. This AI branch aims to bring another type of interaction between humans and computers (Garbade, 2018).

The NLP is used in several common applications, such as Google Translate, Microsoft Word, Grammarly, Virtual Assistants, Interactive Voice Response (used in call centres), and many others (Garbade, 2018).

As human language is very complicated, besides the fact that humans can express one meaning in multiple ways, the tone can change a sentence's meaning. This action makes the NLP face many drawbacks. Manipulating text to understand language is extremely complex; therefore, several techniques were developed to achieve this functionality, the main two techniques being syntactic analysis and semantic analysis (Garbade, 2018).

### Syntax

This technique refers to the arrangement of words within a sentence in such a way that they make grammatical sense. The NLP assesses how the natural language lines up with grammatical rules. The syntactic analysis is implemented using various algorithms, such as (Garbade, 2018):

- **Parsing** – grammatical analysis for the input text
- **Sentence breaking** – placing sentence barriers on large pieces of text
- **Part-of-speech tagging** – identifying the part of speech for every word
- **Morphological segmentation** – dividing words into individual units called morphemes
- **Others**

### Semantic

This technique refers to the meaning that is passed by a text. Semantics is the complex and difficult part of the NLP that has not been entirely resolved yet. The semantic analysis algorithms perform different operations over text to extract meaning and interpretation from words and sentences (Garbade, 2018).

Some algorithms used in the semantic analysis are (Garbade, 2018):

- **Named entity recognition (abbreviated NER)** – identifies the parts of the text that can be categorized into preset groups.
- **Word sense disambiguation** – involves the determination of meaning to a word based on context
- **Natural language generation** – uses databases to obtain semantics and convert them into human language

The NLP techniques are evolving rapidly as more research is conducted in this field.

Some aspects of Natural Language Processing, such as Text-to-Speech and Speech-To-Text, are detailed later in the report, as they are heavily used in the implementation of this project.

### GIL

Python Global Interpreter (GIL) is a mutex, also called a lock, that allows only one thread to hold the Python Interpreter's control. This means that one thread can execute at any point in time.

GIL's impact is invisible for developers that develop single-thread programs, but for developers that are developing systems that need concurrent or parallel processes/threads, GIL can be a performance bottleneck in multi-threaded code (Ajitsaria, 2020).

Python uses reference counting for memory management, meaning that objects created in Python have a reference count variable that keeps track of an object's number of references. If the count reaches zero, the memory occupied by that object is released (Ajitsaria, 2020).

The GIL is a single lock mechanism on the interpreter, which protects the reference count variable from race conditions if multiple threads increase or decrease this value simultaneously. Other languages, such as Java, use different approaches like garbage collection (Ajitsaria, 2020).

The GIL, unfortunately, is not ideal on multi-thread Python programs, as it prevents the programs from taking full advantage of multiprocessor systems (Ajitsaria, 2020).

---

### THREADING

A thread is a separate flow of execution of a process, meaning that a program executes multiple different statements or functions simultaneously.

In Python, because of the GIL, threads do not execute simultaneously, but they appear to do so. The multithread library is light, shared memory between threads, and is used for responsive UIs and I/O bound applications (Noronha, 2017).

Multiple threads usually live in the same process, space and share most resources; Each thread performs a specific task, has its own code, has its own stack memory, and share swap memory (Noronha, 2017).

---

### MULTIPROCESSING

Multiprocessing refers to a system's ability to support more than one process running in parallel simultaneously. Programs that use multiprocessing are broken into smaller and simpler functions that run independently (Noronha, 2017).

On a single-core system, if the program is designed to run multiple processes simultaneously, the processes will run concurrently; it interrupts each process after a certain period, allowing all processes to progress with their execution.

In a multi-core system, the number of processes executing at the same time is less or equal to the number of cores the system has; multiple processes running at the same time each task execution on a core.

In contrast, threads run within the same memory space, while processes have separate memory with the main pros and cons (Brown, 2010):

(The following table refers to Python processes and threads)

<b>Multiprocessing</b>	<b>Multithreading</b>
<b>Separate memory space</b>	Lightweight, shared memory
<b>Takes advantages of multiple CPUs and cores</b>	Allows the development of more responsive UIs Great option for I/O-bound applications
<b>Avoids GIL in Python</b>	cPython C extension modules that properly release the GIL will run in parallel
<b>Child processes are interruptible/killable</b>	Not interruptible/killable
<b>Eliminates most needs for synchronization primitives unless if you use shared memory</b>	If not following a command queue/message pump model, then manual use of synchronization primitives become a necessity

Figure 28 "Multiprocessing vs. Multithreading"

Source: Theodora Tataru, 2021

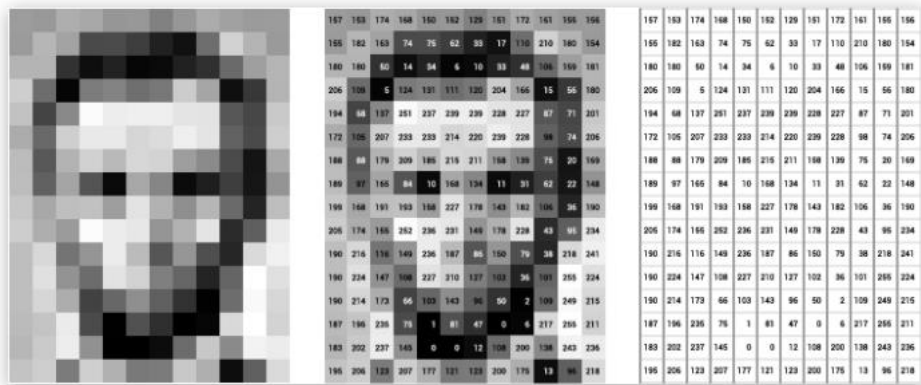
Computer vision is an AI branch designed to process and understand images and videos and how they can be manipulated to retrieve data. This branch of Artificial Intelligence is playing an essential role in self-driving cars, robotics, and photo correction applications (Kulhary, 2019).

Computer vision, as other AI's branches, is used in various applications such as (Szeliski, 2011):

- **Optical character recognition** – that reads handwritten postal codes and letters
- **Retail** - object recognition for automated checkout lanes
- **Automotive safety** - detecting road obstacles
- **Surveillance** – monitoring for intruders, analysing traffic, monitoring pools, etc
- **Others**

Today's computer vision algorithms are based on pattern recognition, achieved by training computers on a substantial amount of visual data by; processing images, label objects, and finding patterns.

This learning step is achieved by accessing, manipulating, and analysing data represented by the image's pixels (Levin, 2020).



The pixel's values are stored in a one-dimensional array. A one-dimensional array is used instead of a two-dimensional array because the computer memory consists of an increasing linear list of address spaces (Levin, 2020).

Figure 29 "Pixel representation of an image"

Source: (Levin, 2020)

Modern Computer Vision relies on Deep Learning, a branch of Machine Learning. Deep Learning uses algorithms that gather insights from data and neural networks, which are used to extract patterns from the data samples (Babich, 2020).

Open Source Computer Vision Library (OpenCV) is an open-source package dedicated to computer vision and machine learning. This library was built to provide a common base for computer vision and to speed up the use of machine perception (OpenCV, 2020).

OpenCV has over 2500 optimization and machine learning algorithms. These algorithms can be implemented to recognize images such as faces, objects, handwriting, human actions, track objects, 3d models of objects, stitch images together to produce a high-resolution picture, and others (OpenCV, 2020).

The library has different interfaces for multiple languages such as Python, C++, Java, and MATHLAB and is supported on various platforms such as Windows, Linux, Android, and macOS (OpenCV, 2020).

Raspberry Pi has its limits when it comes to power; thus, it is not the appropriate architecture for machine learning training (Heath, 2018). For Pixel, the virtual assistant, to recognize what a face is, a training model should be developed and fed with a dataset of approximately 3 million images. The model could be trained with the appropriate dataset, but the process is time-consuming, and Raspberry Pi's limited architecture would work against this project time frame. Thus, OpenCV enables us to use a pre-trained network and then use it to construct 128-d embeddings for each person's face in the dataset, resulting in the model recognizing different people (Rosebrock, 2018).



## FACIAL RECOGNITION

Facial recognition is the process of identifying or verifying a face by capturing, analysing, and comparing patterns based on a person's facial details. (S. Sharma, Karthikeyan Shanmugasundaram, Sathees Kumar Ramasamy, 2016).

Facial recognition is achieved in 4 fundamental processes (S. Sharma, Karthikeyan Shanmugasundaram, Sathees Kumar Ramasamy, 2016):

- **Face detection** is specific object detection designed to recognize the frontal faces of humans from images and videos.
- **Face alignment**
- **Facial cropping** face cropping is the process of machine learning that makes the neural network learn only the features of a face by cropping faces found in an image ignoring the other objects
- **Feature extraction** is the most crucial step in the model that extracts features from the facial images by a convolutional neural method

Once a model had been trained to recognize a particular face, it can be used to identify or verify a person's identity using their face. The process is achieved by capturing, analysing, and comparing patterns based on a person's particular features (Thales, 2020).

- **Face detection** is the first step in the process that detects human faces in a video or image
- **Face capture** transforms the analogue input (the face) into data
- **Face match** is the last step in the process; this involves the model attempting to match vector coordinates to a potential face in the image or video feed provided to it

The most common face recognition type is present in mobile devices and serves as a phone's unlock option.

Dlib is another library similar to OpenCV, designed in C++, that implements machine learning algorithms such as classification, regression, clustering, and predictions. Dlib has two main components: linear algebra and machine learning tools but no specific focus domain. Dlib provides the perfect environment for machine learning development as the core of this library is linear algebra with Basic Linear Algebra Subprograms (BLAS) (S. Sharma, Karthikeyan Shanmugasundaram, Sathees Kumar Ramasamy, 2016).

It contains various software objects for networking, computer vision, image processing, data structure, text parsing, numerical optimization, machine learning, and others (Kulhary, 2019).

The facial contour detector implemented in Dlib produces 68 coordinates for (x,y) that map a specific facial shape. These labels are called landmarks. These 68 mappings of a face are achieved by training a shape model on the labelled iBUG 300-W dataset. The landmarks are used for face predictions (Mr. Mehmet Ucar, Dr. Sheng-Jen, 2018).

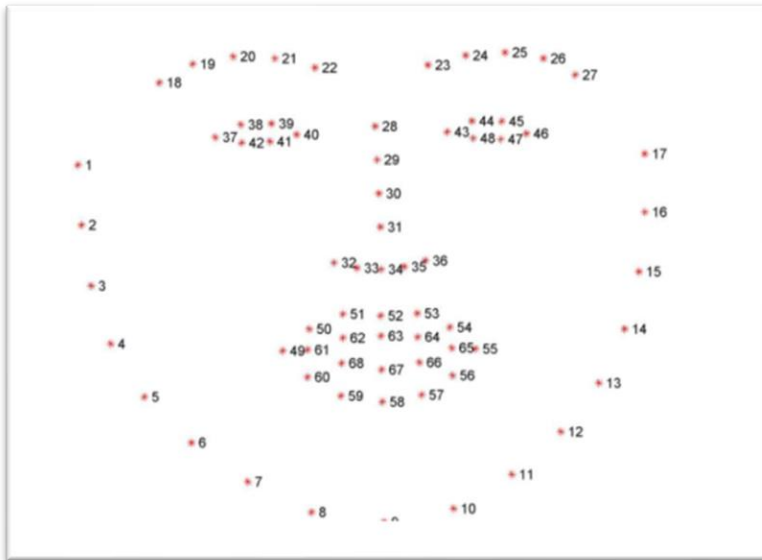


Figure 30 „68(x,y) facial landmarks”

Source: (Mr. Mehmet Ucar, Dr. Sheng-Jen, 2018)

As seen in **Figure 30**:

- The mouth can be accessed through the points between 49 -68
- The left eyebrow through the points 18-22 and the right eyebrow through the points 43-48
- The left eye through 37-42 and right eye 43-48
- The nose through the points 28-36
- The jaw from point 1 to 17 (Mr. Mehmet Ucar, Dr. Sheng-Jen, 2018)

All the data provided above are found in the „FACIAL\_LANDMARKS\_IDXS” dictionary from the „imutils” library.

## TESTING RASPBERRY PI FOR FACE RECOGNITION WITH OPENCV AND DLIB

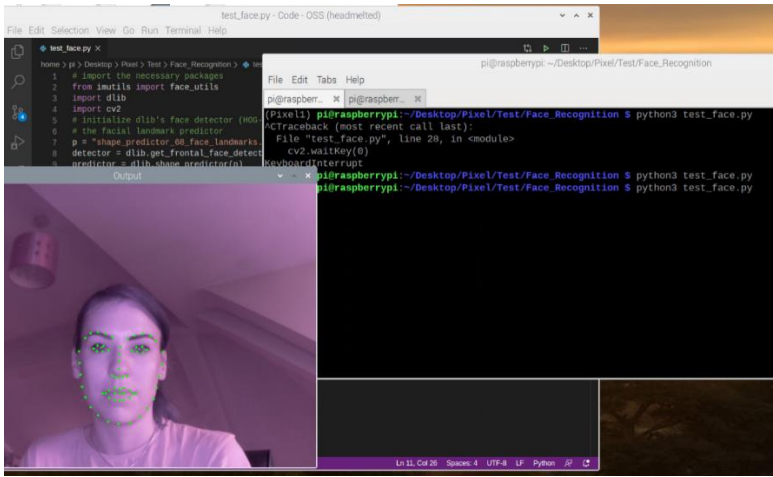


Figure 32 "Face Mapping"

Source: Theodora Tataru, Raspberry Pi 2020

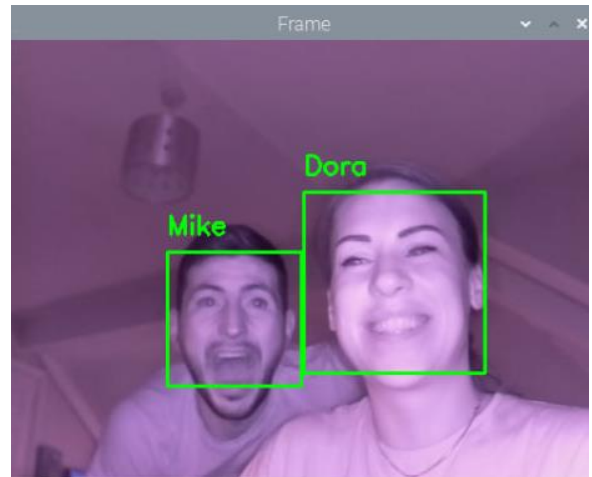


Figure 31 "Face Recognition"

Source: Theodora Tataru, Raspberry Pi 2020

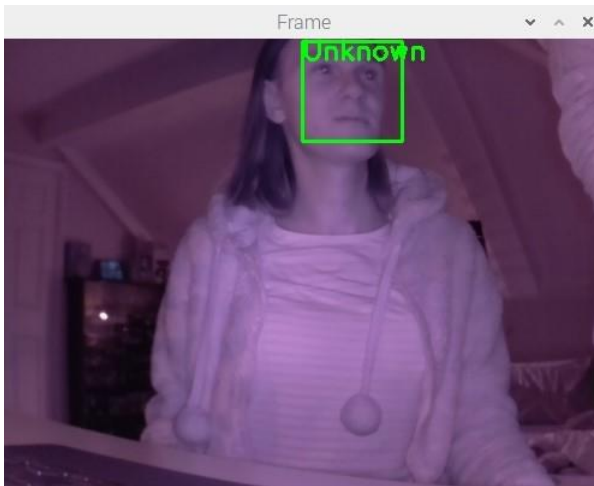


Figure 33 „Unknown person“

Source: Theodora Tataru, Raspberry Pi 2020

Using Dlib and OpenCV, testing face recognition on the Raspberry Pi was a success.

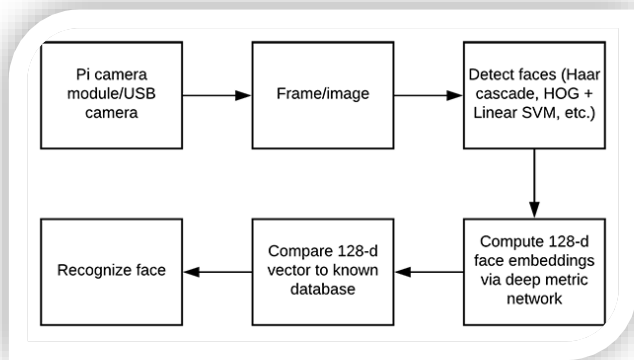
The images captured by the camera are processed and outputted as a real-valued feature vector used to recognize a face (Rosebrock, 2018). The real-valued feature vector contains all the feature values for a given data point (Agarwal, 2013).

The Dlib facial recognition network was fed a set of different pictures with the same person and some random photos with other people for comparison to train the model. The training network to quantify the face is constructing the 128-d embeddings (Rosebrock, 2018). The big picture of this training process is that the 128-d measurements of „Dora“ are very close to each other, and they are further from the measurements of „Mike“ or other persons, as seen in **Figure 31** and **Figure 33**.

Davis King created and trained the network in 2017 with a dataset of no less than 3 million pictures, predicting with 99.38% accuracy (Rosebrock, 2018), (King, 2017).

To train the model from Dlib, 24 pictures were used for each person intended to be recognized by the model and a set of images with random people to compare. The embeddings of the 128-d vectors were saved into a pickle file.

Once the model was trained to recognize specific people, it performs the following steps to recognize faces in real-time (**Figure 34**):



- the images are collected from the Raspberry Pi camera
- each frame is inspected to detect faces

If a face is detected:

- the image is processed to compute 128-d embeddings
- compared with the vector values in the pickle file
- if the values are close/match each other, the face captured by the camera is recognized.

**Figure 34 "Recognizing faces with Raspberry Pi "**

Source: (Rosebrock, 2018)

Local Binary Pattern is a simple and efficient texture operator, which labels the pixel from an image by mapping all the eight neighbours of each pixel and saving the result as a binary number. This library is used for image classification in Computer Vision (Wikipedia, 2020).

In its simplest form, the LBP vector is created as follows:

1. The image is divided into cells
2. For each pixel cell, compare the pixel with all its eight neighbours
3. If the centred pixel has a higher value than its neighbour value writes '0', and if the neighbour has a value less than the centred picture writes '1'.
4. Compute the histogram (256 d feature vector)
5. Concatenate all histograms of all cells, resulting in the feature vector for the whole image (Wikipedia, 2020).

## TESTING RASPBERRY PI FOR FACE RECOGNITION WITH OPENCV AND LBPH

In this experiment, for face recognition, LBPH was used, a library that is part of OpenCV.

500 pictures were used to train the model to recognize a face, and the mapping values of a face were stored in a yml file. This opposed to the first experiment that was saving the vector values in a pickle file.

Even the number of pictures used in this experiment is 20 times higher than the previous experiment; face recognition was a fail as if another person was in the camera range was labeled as "Dora". One of the reasons this experiment was not successful is that it uses an infrared filter for nighttime vision.

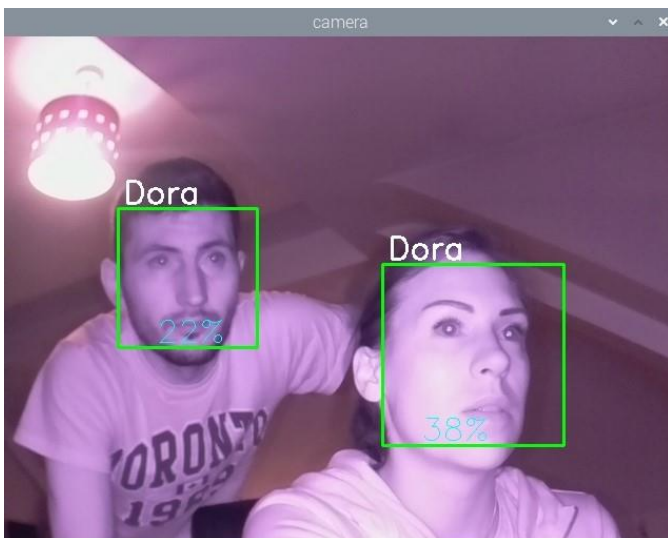


Figure 36 "Facial recognition with LBPH Face Recognizer"  
Source: Theodora Tataru (2020)

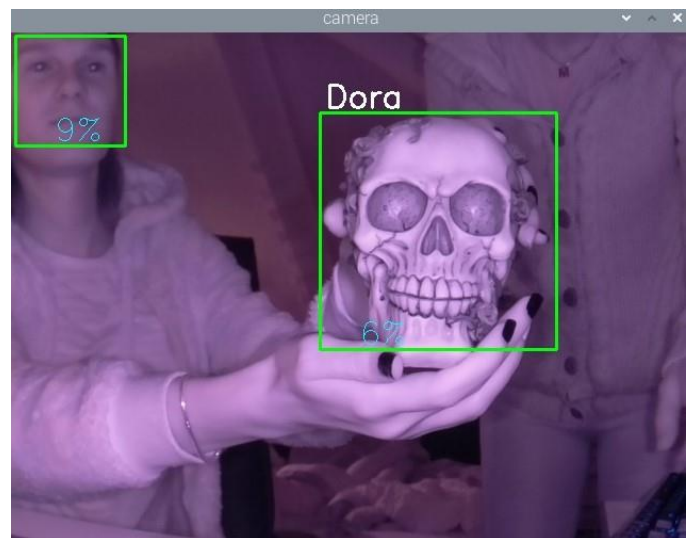


Figure 35 "Facial recognition with LBPH Face Recognizer"  
Source: Theodora Tataru (2020)

The model was trained again with a double amount of pictures (1000 images), to increase accuracy, but the same result was achieved, as seen in **Figure 35**.

In the end, the model was trained to recognize the second person from **Figure 36**, but the result was again erroneous, as different people were labelled again as the same person.

Therefore, this library will not be used for facial recognition implementation for this project.

## OPENCV AND FACE TRACKING

A Pan-Tilt Hat kit was used to track a face as it moves around the room, testing performed using Python and OpenCV.

Using two servos and python code, the camera can move up-down and left-right at the same time. This test aims to move the servos so that the image stays centred on a face. In this experiment, for simplicity, facial recognition wasn't used; instead, facial detection was implemented.

---

## PROPORTIONAL-INTEGRAL-DERIVATIVE CONTROLLER (PID)

The PID controller is a calculation performed on the input used to control the output. Automation is used to control the input and bring this value to the desired point before the output. The PID controller achieves this desired set point by monitoring the input, calculating how far this specific input is from the desired point (DC, 2018).

Some examples where the PID controller can be used are temperature control, voltage control, and cars' cruise control.

The most famous example to describe the use of PID controls is the "car on the hill" analogy.

*"You are driving down the highway and decide to turn on cruise control at 100 km/h. When your vehicle comes to a hill, the speed drops below the set point.*

*In this example, the **setpoint** is 100 km/h, the actual speed drops to 70 km/h, so the **error value** becomes  $100 - 70 = 30$ . The PID controller takes in this error value and determines how much to control the **output**, to bring the process value to the desired set point.*

*The PID controller will calculate the Error value, then press down on the gas pedal until your car reaches the 100 km/h set point (the error becomes zero)."*

(DC, 2018)

- **Setpoint value** = the desired value
- **Process value** = value that needs to be controlled
- **Error** = setpoint value – process value

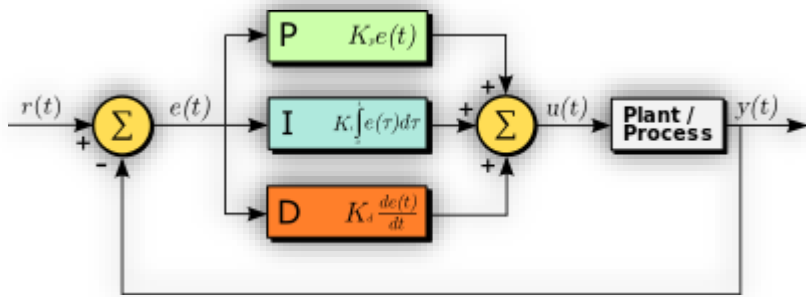
In the example quoted above, the **setpoint is the vehicle speed**, the **error value is the car's cruise control**, and the **output is the acceleration value**.

## CALCULATIONS PERFORMED ON THE ERROR VALUE TO GET THE DESIRED OUTPUT

**Proportional** value is the first calculation done to control the output. It acts immediately on the error, trying to bring the output as close as possible to the setpoint.

**Integral** is the second calculation performed on the error, which continuously accumulates over time until the setpoint was reached.

**Derivative** is monitoring the slope rate and prevents the output from exceeding the setpoint



The calculation from **Figure 37** can be written as an equation, like this:

$$u(t) = K_p e(t) + K_i \int_0^t e(t') dt' + K_d \frac{de(t)}{dt}$$

**Figure 37 “PID Controller”**

A block diagram of a PID controller in a feedback loop.  $r(t)$  is the desired process value or setpoint (SP), and  $y(t)$  is the measured process value (PV).  
Source: (Wikipedia, 2020)

The concept of the PID controller is hard to understand and even harder to be put into practice. To simplify this concept, look at the **Proportional** value as the present, large corrections that need to be made, **Integral** value as the historical corrections, and **Derivative** value as the future corrections that need to be made for the desired output (Rockbrock, 2019).

## MULTIPROCESSING IN PYTHON

The term **multiprocessing** in computer science refers to a system's capability to support more than one execution at one time. Instructions in a multiprocessing system are divided into smaller instructions that can run separately. For this action to be possible, the system must have more than one processing unit (CPUs or cores), each sharing main memory and peripherals (Wikipedia, 2020).

On a single-core system, if the CPU has allocated more than one process or thread at a time, it has to pause each task and switch fast to another to keep the queue of tasks progressing. This action is called **concurrency**, where the user has the illusion that multiple tasks are running simultaneously. However, the system swaps the task extremely fast so that all processes can progress (Kumar, 2018).

In Python, to activate the multiprocessing feature, the **multiprocessing** package needs to be imported.

This module accepts two arguments:

- target – the function that will be executed as a thread
- args – the arguments to be passed to the function

```
p1 = Process(target=power, args=(2,3)) #2 at the power of 3
p2 = Process(target=power, args=(1,3)) #1 at the power of 3
```

The thread is started by calling the “.start()” method.

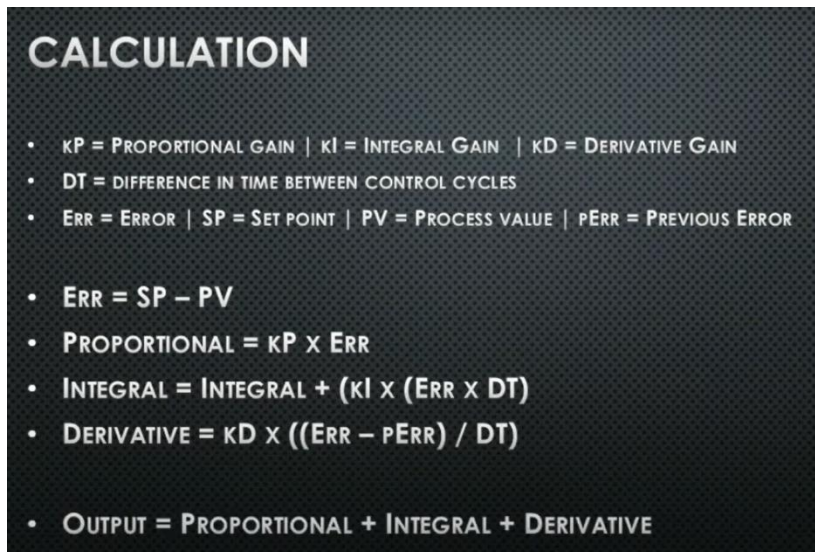
To block the thread's termination before it finishes its execution, the “.join()” method is used.

---

## PID CONTROLLER AND MULTIPROCESSING USED ON RASPBERRY PI SERVOS

As each servo is controlled individually, the multiprocessing module from Python is used to enable the manipulation and coordination of the two servos smoothly as they were one. If the multiprocessing module is not used, the servos' movement is performed abruptly, as each servo needs to wait for the other servo to finish its movement.

The PID controller class calculates the coordinates of where the servos need to be moved so that the tracked face is maintained in the middle of the video frame.



**CALCULATION**

- KP = PROPORTIONAL GAIN | KI = INTEGRAL GAIN | KD = DERIVATIVE GAIN
- DT = DIFFERENCE IN TIME BETWEEN CONTROL CYCLES
- ERR = ERROR | SP = SET POINT | PV = PROCESS VALUE | PERR = PREVIOUS ERROR
  
- $ERR = SP - PV$
- $PROPORTIONAL = KP \times ERR$
- $INTEGRAL = INTEGRAL + (KI \times (ERR \times DT))$
- $DERIVATIVE = KD \times ((ERR - PERR) / DT)$
  
- $OUTPUT = PROPORTIONAL + INTEGRAL + DERIVATIVE$

Figure 38 “Calculations for Proportional,-Integral-Derivative Controller”

Source: (DC, 2018)

To test the Raspberry Pi servos, a PID controller class was used with the following numerical gains (Rockbrock, 2019):

- Horizontal
  - Kp : 0.09
  - Ki: 0.08
  - Kd: 0.002
  
- Vertical:
  - Kp: 0.10
  - Ki: 0.11
  - Kd: 0.002

To calculate the difference between the moving cycles, the **delta time** was calculated by the difference between the **current time** minus the **previous time** the servo had moved (Rockbrock, 2019).

```
# grab the current time and calculate delta time
self.currTime = time.time()
deltaTime = self.currTime - self.prevTime
(Rockbrock, 2019)
```

The **current error** was calculated by computing the difference between the **center of the frame** (x,y position) **and the center of the face coordinates** (x,y position). The **delta error** was defined as the difference between the current error and the previous error.



```
# calculate the error
error = centerCoord.value - objCoord.value
# delta error
deltaError = error - self.prevError
(Rockbrock, 2019)
```

And finally, as the delta time and the delta error were calculated, the proportional, integral and the current derivative values can be calculated:

```
# proportional term
self.cP = error
# integral term
self.cI += error * deltaTime
# derivative term and prevent divide by zero
self.cD = (deltaError / deltaTime)
(Rockbrock, 2019)
```

Thus, the camera's new angle can be calculated by multiplying each value with its gain and sum all the multiplication together.

As the new coordinates had been calculated, the next step is to move the servos to the exact angles so that the tracked object to be centred in the middle of the video frame.

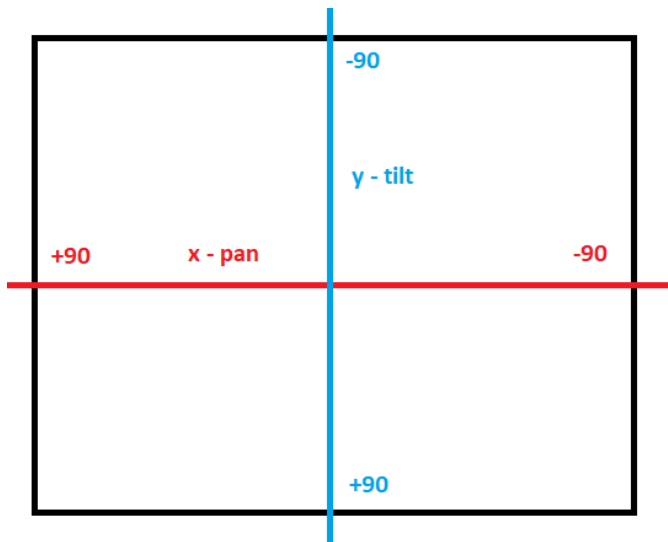
The multiprocessing package is used to coordinate the servos to move in union.

```
processPanning = Process(target=pid_process, args=(pan, panP, panI, panD, objX, centerX))
processTilting = Process(target=pid_process, args=(tlt, tiltP, tiltI, tiltD, objY, centerY))
processPanning.start()
processTilting.start()
processPanning.join()
processTilting.join()
(Rockbrock, 2019)
```

The experiment was a success, as the camera follows a face successfully, but the algorithms need to be modified and optimized, as the camera loses the tracked face often, and it recovers hard 70% of the time.

## TESTING THE RASPBERRY PI PANTILT HAT USING PID CONTROLLER

The servos from the Pan-Tilt Hat were tested using Python. The purpose of the test was to transpose into code the PID controller concept with the goal of moving the servos with the movement of a user.



As seen in **Figure 39**, the video frame's coordinates values might look unusual, but they are correct. As each servo has a 180-degree range of movement, the servos' neutral point is in the middle at 0, 0. To move the tilt servo in the up direction, it needs a value from -1 to -90; thus, to move the servo down, it needs a value from 1 to 90. The same principle applies to the pan servo.

The first module implemented is the `face_utils.py` class. This class uses OpenCV for facial detection. This class's purpose is to detect all faces in the video frame, get the center coordinates of the closest face, and return the values, as seen in **Figure 40**.

**Figure 39** "Video frame coordinates according to PanTilt Hat"

Source: Theodora Tataru 2020

The second module is the `PID` class (**Figure 40**), which implements the PID controller concept to coordinate the servos movement according to the center coordinates of the face and the center of the camera video frame.

```
self.cd = 0
self.ci = 0

def calculate_new_angle(self,error):
    time.sleep(0.1)

    self.current_time = time.time()

    # time difference between cycles
    delta_time = self.current_time - self.previous_time

    # difference between current error and prev error
    delta_error = error - self.previous_error

    # ERROR
    self.cP = error

    # the integral will accumulate overtime until the setpoint is reached
    self.ci += error * delta_time

    # is monitoring the slope to don't overshoot the error
    self.cd = (delta_error/delta_time) if delta_time > 0 else 0

    # save current values for next iteration
    self.previous_time = self.current_time
    self.previous_error = error

    # returns the new coordinates for the servo
    return sum([
        self.kP * self.cP,
        self.kI * self.ci,
        self.kD * self.cd])

class face_utils:
    def __init__(self):
        self.face_detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

    def update(self, frame, centerX, centerY):
        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # TODO: keep track of the person was there first
        # detect all faces in the video frame -> look up MIN_NEIGHBOURS
        faces_detected = self.face_detector.detectMultiScale(gray_frame, scaleFactor=1.1,
            minNeighbors=9, minSize=(30, 30), flags=cv2.CASCADE_SCALE_IMAGE)

        if len(faces_detected) > 0:
            (top_left_x, top_left_y, face_width, face_height) = faces_detected[0] #First
            # get the center of the face detected
            center_face_x = int((top_left_x + (face_width / 2) )
            center_face_y = int((top_left_y + (face_height / 2) ))

            return (center_face_x, center_face_y, faces_detected[0])
        # if no face was detected, return the initial value
        return (centerX, centerY, None)
```

**Figure 40** "PID Controller and Face Centering"

Source: Theodora Tataru 2020

As seen in **Figure 40**, the PID Controller definition is designed using the formulas detailed in the chapter above. Its core purpose is to process the coordinates of the center of the face and the coordinates of the camera video frame to control the servos' movement as much as possible the face in the middle of the camera video frame.

The result of the experiment was successful. 70% of the time, the servos were moving according to the focused face, but there were few significant errors:

- If more than one face was detected, the servo was acting sporadically, losing focus of all subjects
- The servos movement was abrupt
- Sometimes, even a subject was focused and approximately in the middle of the video frame; the servos were moving drastically to the edges of their range

As a result of this experiment, it was expected to control the servos according to an object's movement. The result can be classified as successful, but the code used needs improvement and optimization for a smooth experience.

Marcelo's (Rovai, 2018), Adrian's (Rockbrock, 2019), and Pimoroni's (Pimoroni, 2017) tutorials were helpful learning resources in the development of this servos test.

Further tests were produced by modifying different values into the PID controller class, but the servos' sudden moves were persistent. Therefore, this feature will be implemented if time permits; meanwhile, the development focuses on the device's core functionalities: virtual assistant, face recognition, skills, and the interface.

## TESTING THE GSM/GPRS/GNSS HAT

The GSM/GPRS/GNSS Hat was tested for several, but not for all functionalities that it is capable of.

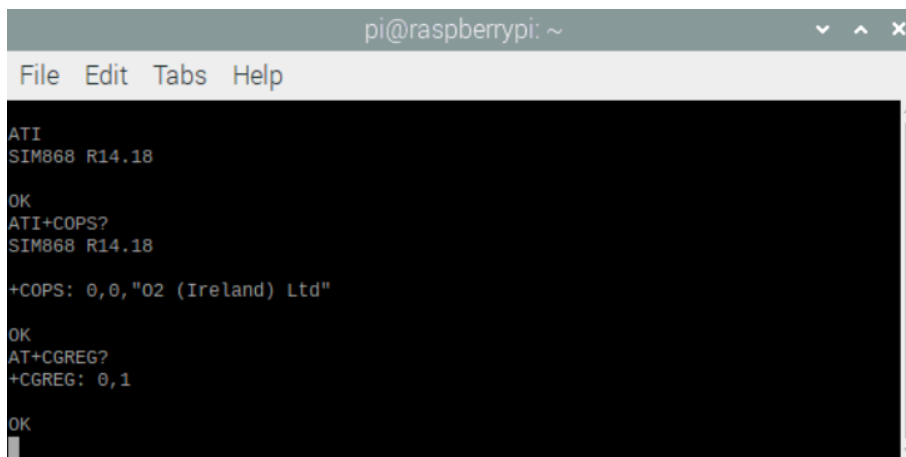
The functionalities needed for this project are:

- Send SMS
- Receive SMS
- Perform a call
- Get GPRS details such as latitude and longitude

A SIM card was inserted into the module, with the following phone number: (+353) 83 025 6686.

The purpose of this HAT is to increase the accuracy of some skills such as weather forecast, and to implement other skills from scratch such as send SOS (send an emergency text message to the user's next of kin), call a location (such as a cinema, a restaurant or others) and others.

## TESTING GPS MODULE VIA SERIAL CONSOLE



```
pi@raspberrypi: ~
File Edit Tabs Help
ATI
SIM868 R14.18
OK
ATI+COPS?
SIM868 R14.18
+COPS: 0,0,"02 (Ireland) Ltd"
OK
AT+CGREG?
+CGREG: 0,1
OK
```

In **Figure 41**, the GPS module was tested using the screen terminal, executing the `screen/dev/USB0 115200` command. 115200 is a numeric value that represents the rate at which information is transferred over a communication channel. This rate is called „Baud Rate”, and is mainly used in electronics and serial communication.

The AT+ command returns general information about the GPS module. As seen in **Figure 41**, the module's general status is marked as „ok” after the AT+ command.

**Figure 41** „GPS module status”

Source: Theodora Tataru, 2021

The AT+`COPS?` command returns the current SIM operator, in this case, 02 Ireland. This information confirms that the SIM card is correctly read by the module, as the inserted SIM card provider is 02 Ireland.

The AT+`CGREG?` command confirms the registration status of the provider with a „0,1” output.

To proceed with the GPS module's testing, by sending and receiving text messages, retrieving the GPS coordinates, and performing calls, the Python `gsmHat` module was installed. However, the `gsmHat` module did not perform as expected as the geolocation coordinates and details were all returned as zero. The `gsmHat` python module had been downloaded to resolve this issue, and its implementation had been changed to return the correct values.

Below, the changed section of the `gsmHat` python module can be observed:

```
class GPS:
    . . .

    def __init__(self):
        self.GNSS_status = ""
        self.Fix_status = ""
        self.UTC = '' # yyyyMMddhhmmss.sss
        self.Latitude = "" # ±dd.dddddd [-90.000000,90.000000]
        self.Longitude = "" # ±ddd.dddddd [-180.000000,180.000000]
        self.Altitude = "" # in meters
```

```

self.Speed = "" # km/h [0,999.99]
self.Course = "" # degrees [0,360.00]
self.HDOP = "" # [0,99.9]
self.PDOP = "" # [0,99.9]
self.VDOP = "" # [0,99.9]
self.GPS_satellites = "" # [0,99]
self.GNSS_satellites = "" # [0,99]
self.Signal = "" # max = 55 dBHz
. . .
. . .

```

```
class GSMHat:
```

```
    """GSM Hat Backend with SMS Functionality (for now)"""
```

```
    . . .
    . . .
```

```
    def __processData(self):
```

```
        if self.__serData != '':
```

```
            . . .
```

```
            # GPS Data coming here
```

```
            elif '+CGNSINF:' in self.__serData:
```

```
                self.__logger.debug('New GPS Data:')
```

```
                match = re.findall(self.regexGetAllValues, self.__serData)
```

```
                rawData = match[0][1].split(',')
```

```
                if len(rawData) == 21:
```

```
                    newGPS = GPS()
```

```
                    newGPS.GNSS_status = rawData[0]
```

```
                    #print("GNSS Status: " + rawData[0])
```

```
                    self.GPS_Data["GNSS Status"] = rawData[0]
```

```
                    newGPS.Fix_status = str(rawData[1])
```

```
                    #print("Fix Status: " + rawData[1])
```

```
                    self.GPS_Data["Fix Status"] = rawData[1]
```

```
                    #newGPS.UTC = datetime.strptime(rawData[2][:4], '%Y%m%d%H%M%S')
```

```
                    #print("UTC:" + datetime.strptime(rawData[2][:4], '%Y%m%d%H%M%S'))
```

```
                    newGPS.Latitude = str(rawData[3])
```

```
                    #print("Latitude: " + str(rawData[3]))
```

```
                    self.GPS_Data["Latitude"] = rawData[3]
```

```
                    newGPS.Longitude = str(rawData[4])
```

```
                    #print("Longitude: " + str(rawData[4]))
```

```
                    self.GPS_Data["Longitude"] = rawData[4]
```

```
                    newGPS.Altitude = str(rawData[5])
```

```
                    #print("Altitude: " + str(rawData[5]))
```

```
                    self.GPS_Data["Altitude"] = rawData[5]
```

```
                    newGPS.Speed = str(rawData[6])
```

```
                    #print("Speed: " + str(rawData[6]))
```

```
                    self.GPS_Data["Speed"] = rawData[6]
```

```
                    newGPS.Course = str(rawData[7])
```

```
                    #print("Course: " + str(rawData[7]))
```

```
                    self.GPS_Data["Course"] = rawData[7]
```

```
                    newGPS.HDOP = str(rawData[10])
```

```
                    #print("HDOP" + str(rawData[10]))
```

```
                    self.GPS_Data["HDOP"] = rawData[10]
```

```
                    newGPS.PDOP = str(rawData[11])
```

```
                    #print("PDOP: " + str(rawData[11]))
```

```
                    self.GPS_Data["PDOP"] = rawData[11]
```

```
                    newGPS.VDOP = str(rawData[12])
```

```
                    #print("VDOP: " + str(rawData[12]))
```

```
                    self.GPS_Data["VDOP"] = rawData[12]
```

```
                    newGPS.GPS_satellites = str(rawData[14])
```

```
                    #print("GPS Satellites: " + str(rawData[14]))
```

```
                    self.GPS_Data["GPS Satellites"] = rawData[14]
```

```
        newGPS.GNSS_satellites = str(rawData[15])
        #print("GNSS Satellites: " + str(rawData[15]))
        self.GPS_Data["GNSS Satellites"] = rawData[15]

        newGPS.Signal = str(rawData[18])
        #print("Signal: " + str(rawData[18]))
        self.GPS_Data["Signal"] = rawData[18]

        self.__GPSactualData = newGPS

    self.__serData = ''

def GPS_Data_List(self):
    self.__processData()
    return self.GPS_Data
```

The modified gsmHat module was saved in a file called GSM.py on the Raspberry Pi.

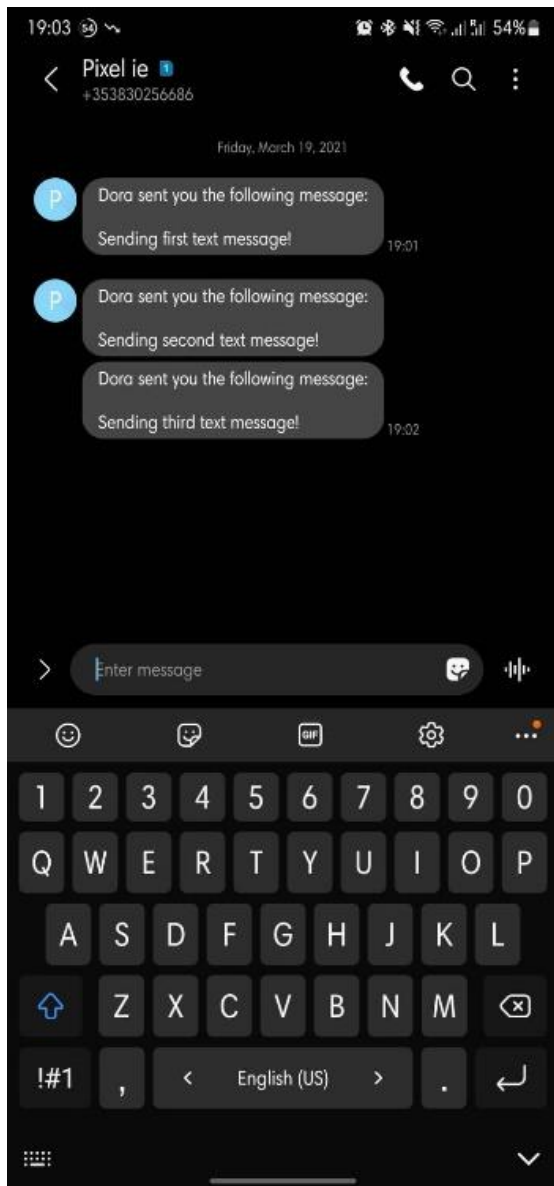
---

## SEND SMS

Using the GSM.py, the following code was produced to test the „SEND SMS” functionality:

```
import GSM
import time
def send_SMS(sender:str, receiver:str, message:str):
    # Send SMS
    gsm = GSM.GSMHat('/dev/ttyUSB0', 115200)
    Number = receiver
    Message = sender + " sent you the following message:\n\n" + message
    gsm.SMS_write(Number, Message)
    time.sleep(1)
```

The test succeeded as the receiver successfully received messages sent using the GPS module, as seen in **Figure 42**.



Three different text messages were sent to the receiver, few seconds apart.

**Figure 42** „Received messages from GPS module”

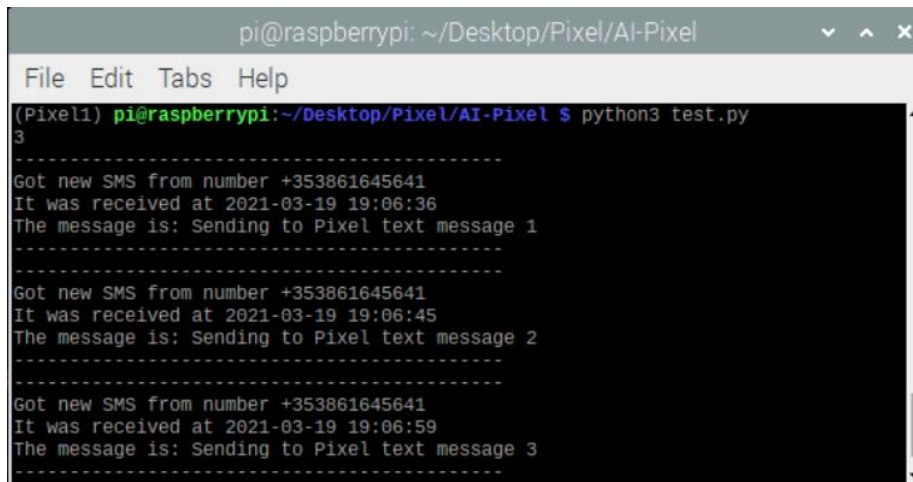
Source: Theodora Tataru, 2021

## RECEIVE SMS

Using the GSM.py, the following code was produced to test the „Receive SMS” functionality:

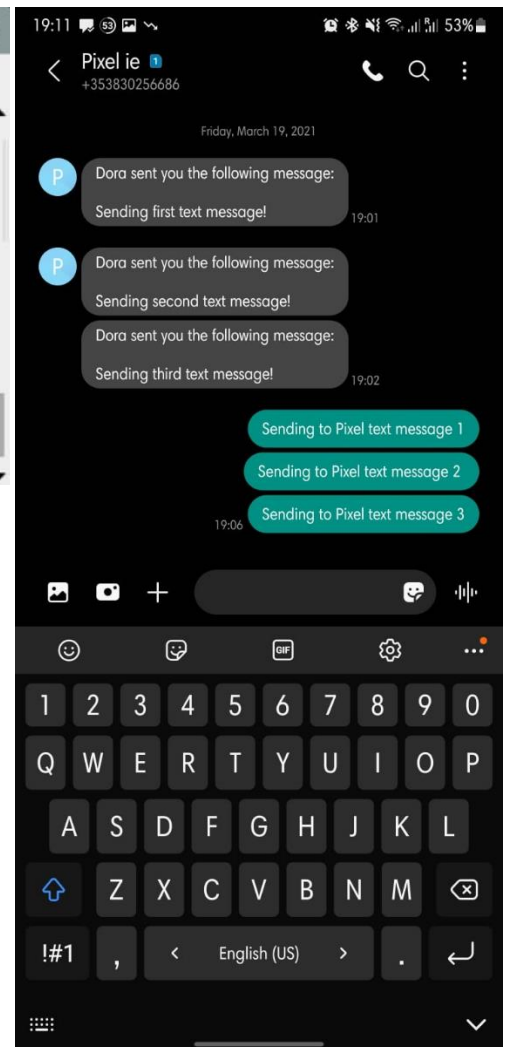
```
def check_received_SMS():
    gsm = GSM.GSMHat('/dev/ttyUSB0', 115200)
    time.sleep(1)
    print(gsm.SMS_available())
    if gsm.SMS_available() > 0:
        number_messages = gsm.SMS_available()
        while number_messages != 0:
            new_sms = gsm.SMS_read()
            print("-----")
            print('Got new SMS from number %s' % new_sms.Sender)
            print('It was received at %s' % new_sms.Date)
            print('The message is: %s' % new_sms.Message)
            print("-----")
            number_messages -= 1
        time.sleep(1)
```

This test was a success; as seen in **Figure 43**, the text messages sent to the SIM card inserted into the GPS module were received.



```
pi@raspberrypi: ~/Desktop/Pixel/AI-Pixel
File Edit Tabs Help
(Pixel1) pi@raspberrypi:~/Desktop/Pixel/AI-Pixel $ python3 test.py
3
-----
Got new SMS from number +353861645641
It was received at 2021-03-19 19:06:36
The message is: Sending to Pixel text message 1
-----
Got new SMS from number +353861645641
It was received at 2021-03-19 19:06:45
The message is: Sending to Pixel text message 2
-----
Got new SMS from number +353861645641
It was received at 2021-03-19 19:06:59
The message is: Sending to Pixel text message 3
-----
```

**Figure 43 „Received messages – GPS module”**  
Source: Theodora Tataru, 2021



**Figure 44 „Sent messages from phone to GPS module”**  
Source: Theodora Tataru, 2021



---

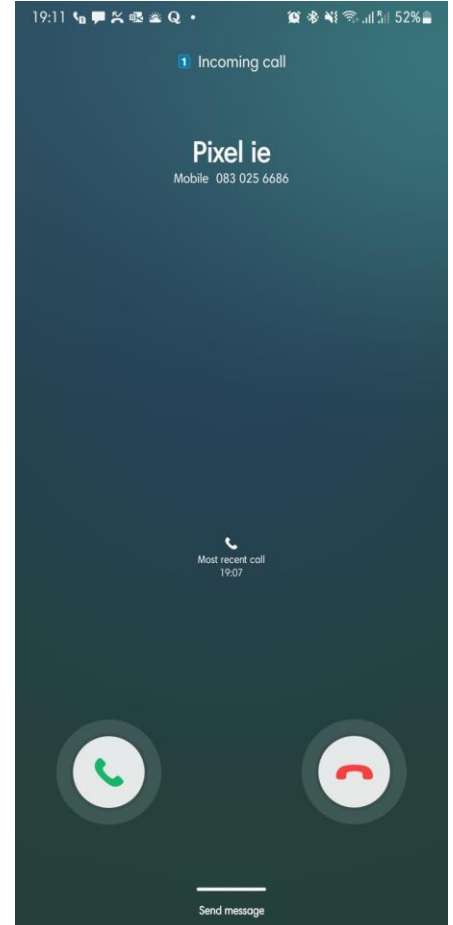
## PERFORM A CALL

Using the GSM.py, the following code was produced to test the „Perform call” functionality:

```
def call(number):  
    gsm = GSM.GSMHat('/dev/ttyUSB0', 115200)  
    print("Dialing...")  
    print("Calling...")  
    gsm.Call(number, 20)    # Or lets change the timeout to 20 seconds. This call hangs up automatically after 60 seconds  
    time.sleep(10)         # Wait 10 seconds ...  
    gsm.HangUp()
```

As can be seen in **Figure 45**, the call had been received from the 086 025 6686 SIM card, inserted in the GPS module.

The test was a success, but further implementation is required to enable users on both ends to communicate with each other.



**Figure 45 "Receiving a call from GPS module"**

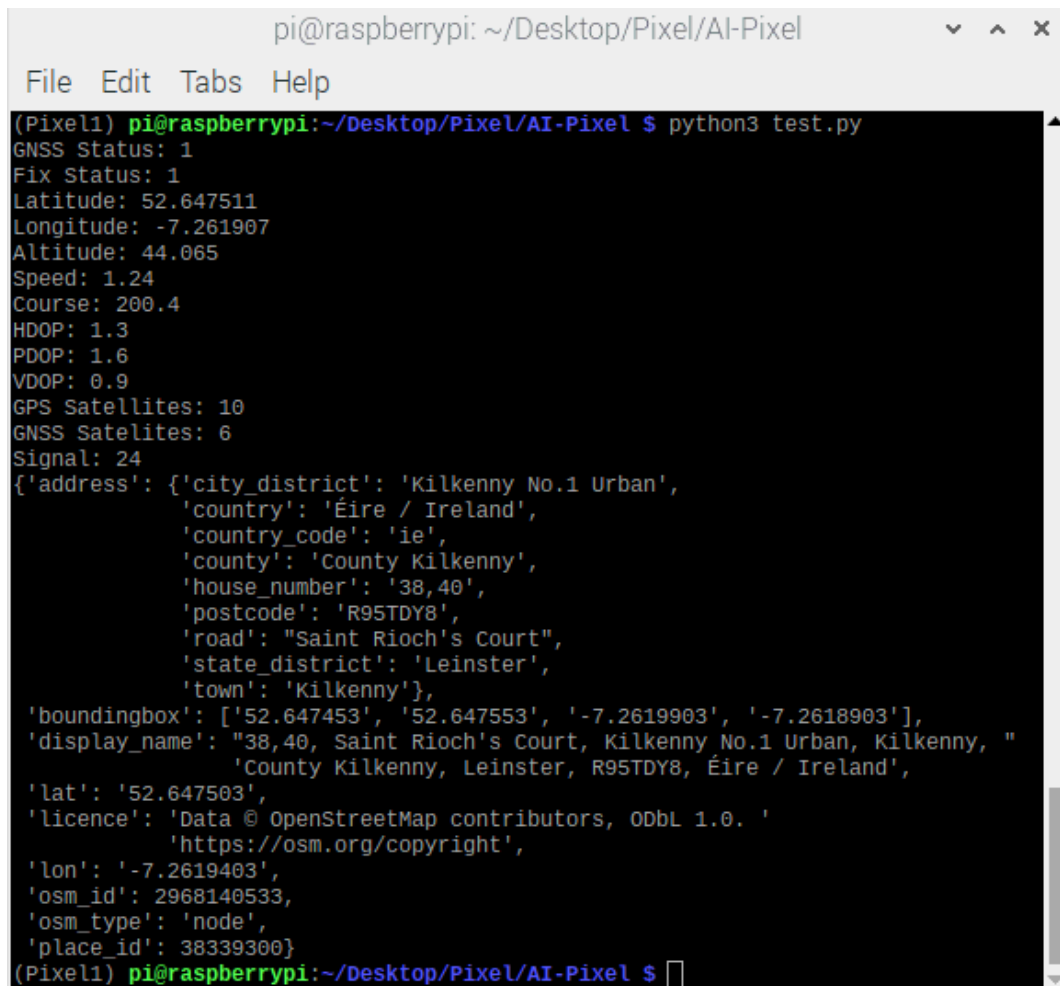
Source: Theodora Tataru, 2021

## GPS

Using the GSM.py, the following code was produced to test the „Get GPS details” functionality:

```
def get_location():
    from geopy.geocoders import Nominatim
    # Lets print some values
    gsm = GSM.GSMHat('/dev/ttyUSB0', 115200)
    time.sleep(5)
    GPSObj = gsm.GetActualGPS()
    time.sleep(5)
    GPS_Data = gsm.GPS_Data_List()
    for key in GPS_Data:
        print(key + ": " + GPS_Data[key])
    lat = GPS_Data['Latitude']
    lon = GPS_Data['Longitude']
    point = lon + "," + lat
    geolocator = Nominatim(user_agent="Pixel")
    location = geolocator.reverse(point)
    address = location.address
    print(address)
```

The test was also a success as the output seen in **Figure 46** is almost accurate. The EireCode provided by the GPS matches the number 38 on the address.



```
pi@raspberrypi: ~/Desktop/Pixel/AI-Pixel
File Edit Tabs Help
(Pixel1) pi@raspberrypi:~/Desktop/Pixel/AI-Pixel $ python3 test.py
GNSS Status: 1
Fix Status: 1
Latitude: 52.647511
Longitude: -7.261907
Altitude: 44.065
Speed: 1.24
Course: 200.4
HDOP: 1.3
PDOP: 1.6
VDOP: 0.9
GPS Satellites: 10
GNSS Satellites: 6
Signal: 24
{'address': {'city_district': 'Kilkenny No.1 Urban',
              'country': 'Éire / Ireland',
              'country_code': 'ie',
              'county': 'County Kilkenny',
              'house_number': '38,40',
              'postcode': 'R95TDY8',
              'road': "Saint Rioch's Court",
              'state_district': 'Leinster',
              'town': 'Kilkenny'},
 'boundingbox': ['52.647453', '52.647553', '-7.2619903', '-7.2618903'],
 'display_name': "38,40, Saint Rioch's Court, Kilkenny No.1 Urban, Kilkenny, "
                 "County Kilkenny, Leinster, R95TDY8, Éire / Ireland",
 'lat': '52.647503',
 'licence': 'Data © OpenStreetMap contributors, ODbL 1.0. '
            'https://osm.org/copyright',
 'lon': '-7.2619403',
 'osm_id': 2968140533,
 'osm_type': 'node',
 'place_id': 38339300}
(Pixel1) pi@raspberrypi:~/Desktop/Pixel/AI-Pixel $
```

Figure 46 "GPS output"

Source: Theodora Tataru, 2021

## OPEN-SOURCE VOICE ASSISTANTS

The project's goal is to create a virtual assistant for people with hearing impairment and people who struggle with technology. Above, multiple options were elaborated that serve for the design of the project.

Face recognition comes in use to identify individuals as they approach the virtual assistant. Servos assist the camera to keep the focus on the user's face during the interaction. But the core of the project is the virtual assistant that processes the user request and responds according to the meaning.

As detailed earlier in the document, there are multiple choices for digital assistants, but all the selections mentioned before are gadgets offered by companies as their product surcharge. These companies do not disclose their technologies, approaches, and methodologies of achieving their selling commodity.

This section of the report is focused on open-source virtual assistants, projects that are open to free use, modification, and contribution.

Training virtual assistants are expensive in time and means because the task requires large datasets of natural language. This data comes from paid workers whose job is to commit to discussions to create these samples that serve as training data for a model (Andrews, 2020).

## JASPER AI

As mentioned above, this project's whole idea was circling around the concept of a virtual assistant, and Jasper AI was the first finding at the beginning of the research.

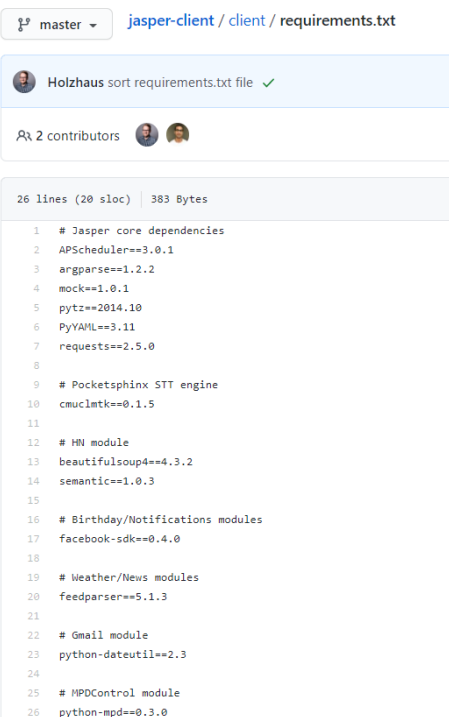
Jasper AI is an open-source virtual assistant that can be controlled by voice and that meets all the functionality of digital assistants. Jasper is active listening for the wake word to come in use and, depending on the hardware, can hear the user speak from meters away (Charlie Marsh, 2020).

Jasper is designed to use third-party modules written by developers around the world, some listed below (Charlie Marsh, 2020):

- **Google Calendar** – lets the user manipulate calendar events
- **Twitter** – allows the user to tweet and read notifications such as messages, retweets, and mentions
- **WolframAlpha** – enables the user to ask questions to the WolframAlpha search engine
- **Movies** - the user can retrieve information about movies
- **Wiki** - can be used to retrieve Wikipedia summaries
- **Find my iPhone** – forcing the user's phone to ring using the apple service
- **Stocks** -used to get live stock quotes
- **Others**

Jasper AI is designed for Raspberry Pi and demands additional Raspberry Pi hardware to have the perfect base for a virtual assistant.

The following hardware is needed to implement Jasper AI:



The screenshot shows a GitHub repository for 'jasper-client' with a file named 'requirements.txt'. The file content lists various Python dependencies for the project, including APScheduler, argparse, mock, pytz, PyYAML, requests, Pocketsphinx STT engine, cmuclmtk, HN module, BeautifulSoup4, semantic, Birthday/Notifications modules, facebook-sdk, Weather/News modules, feedparser, Gmail module, python-dateutil, and MPDControl module.

```
1 # Jasper core dependencies
2 APScheduler==3.0.1
3 argparse==1.2.2
4 mock==1.0.1
5 pytz==2014.10
6 PyYAML==3.11
7 requests==2.5.0
8
9 # Pocketsphinx STT engine
10 cmuclmtk==0.1.5
11
12 # HN module
13 beautifulsoup4==4.3.2
14 semantic==1.0.3
15
16 # Birthday/Notifications modules
17 facebook-sdk==0.4.0
18
19 # Weather/News modules
20 feedparser==5.1.3
21
22 # Gmail module
23 python-dateutil==2.3
24
25 # MPDControl module
26 python-mpd==0.3.0
```

- Minimum Raspberry Pi 3 Model B
- USB microphone
- Wifi or Ethernet cable
- Speakers
- 4GB SD card

The quickest way to install JasperAI on the Raspberry Pi is to download the pre-compiled disk image and burn it on the SD card. As a second option for ArchLinux OS, there are packages available in the Arch User Repository, but it requires Text-to-Speech and Speech-to-Text packages (Charlie Marsh, 2020).

The 3<sup>rd</sup> option requires compiling JasperAI from scratch on a Rasperrian OS. This option requires the installation of Python, pip, pyaudio, libasound, libportaudio, and other libraries.

After the libraries' successful installation, the JasperAI GitHub repository needs to be cloned on the device. The repository has an additional requirements text file that checks the system for all the dependencies required, as seen in **Figure 47** (Charlie Marsh, 2020).

Other requirements that are necessary to give voice to JasperAI are; Text-to-Speech (TTS) used by JasperAI to answer the user's commands and Speech-to-Text (STT) to understand the user's commands. The choice of these libraries is up to developer preferences.

**Figure 47 "JasperAI requirements"**

Source: (Charlie Marsh, 2020)

The developers of JasperAI have tested their digital assistant with PocketSphinx and CMUCLMTK for voice recognition, but the libraries need to be installed by the developer.

## PocketSphinx

PocketSphinx is a part of an open-source project called CMU Sphinx, and it provides a python interface to the library. Windows, Linux, and macOS support the package. The usage of this library is focused on voice recognition (pypi, 2018):

- LiveSpeech – is an iterator class that listens continuously for a specific keyword

```
from pocketsphinx import LiveSpeech

speech = LiveSpeech(lm=False, keyphrase='forward', kws_threshold=1e-20)
for phrase in speech:
    print(phrase.segments(detailed=True))
```

(pypi, 2018)

- AudioFile is another iterator class for a continual search of a keyword from a file.

```
from pocketsphinx import AudioFile

audio = AudioFile(lm=False, keyphrase='forward', kws threshold=1e-20)
for phrase in audio:
    print(phrase.segments(detailed=True)) # => "[('forward', -617, 63, 121)]"
```

(pypi, 2018)

- Pocketsphinx is a proxycass to decode pocketsphinx

```
from __future__ import print_function
import os
from pocketsphinx import Pocketsphinx, get_model_path, get_data_path

model_path = get_model_path()
data_path = get_data_path()

config = {
    'hmm': os.path.join(model_path, 'en-us'),
    'lm': os.path.join(model_path, 'en-us.lm.bin'),
    'dict': os.path.join(model_path, 'cmudict-en-us.dict')}
ps = Pocketsphinx(**config)
ps.decode(
    audio_file=os.path.join(data_path, 'goforward.raw'),
    buffer_size=2048,
    no_search=False,
    full_utt=False)
print(ps.segments()) # => ['<s>', '<sil>', 'go', 'forward', 'ten', 'meters', '</s>']
print('Detailed segments:', *ps.segments(detailed=True), sep='\n') # => [
#   word, prob, start_frame, end_frame
#   ('<s>', 0, 0, 24)
#   ('<sil>', -3778, 25, 45)
#   ('go', -27, 46, 63)
#   ('forward', -38, 64, 116)
#   ('ten', -14105, 117, 152)
#   ('meters', -2152, 153, 211)
#   ('</s>', 0, 212, 260)]

print(ps.hypothesis()) # => go forward ten meters
print(ps.probability()) # => -32079
print(ps.score()) # => -7066
print(ps.confidence()) # => 0.04042641466841839

print(*ps.best(count=10), sep='\n') # => [
#   ('go forward ten meters', -28034)
#   ('go for word ten meters', -28570)
#   ('go forward and majors', -28670)
#   ('go forward and meters', -28681)
#   ('go forward and readers', -28685)
```

```
# ('go forward ten readers', -28688)
# ('go forward ten leaders', -28695)
# ('go forward can meters', -28695)
# ('go forward and leaders', -28706)
# ('go for work ten meters', -28722)
# ]
```

(pypi, 2018)

- In case no parameters are passed to Pocketsphinx, AudioFile, or LiveSpeech class, the default configuration is:

```
verbose = False
logfn = /dev/null or nul
audio_file = site-packages/pocketsphinx/data/goforward.raw
audio_device = None
sampling_rate = 16000
buffer_size = 2048
no_search = False
full_utt = False
hmm = site-packages/pocketsphinx/model/en-us
lm = site-packages/pocketsphinx/model/en-us.lm.bin
dict = site-packages/pocketsphinx/model/cmudict-en-us.dict
```

(pypi, 2018)

PocketSphinx is considered a lightweight and simple engine used in speech recognition and is popular in mobile and desktop development.

### CMUCLMTK

CMUCLMTK is a wrapper library for CMU Sphinx mentioned above, which consists of a subtle layer of code that interprets an existing library interface into a compatible interface. It provides a better implementation and broader compatibility across languages and platforms (pypi, 2014).

When all requirements are met, and the Text-to-Speech and Speech-to-Text are chosen, JasperAI is ready for configuration. For the digital assistant to report weather conditions, send SMS, emails, read notifications, and others, it needs a user profile (Charlie Marsh, 2020).

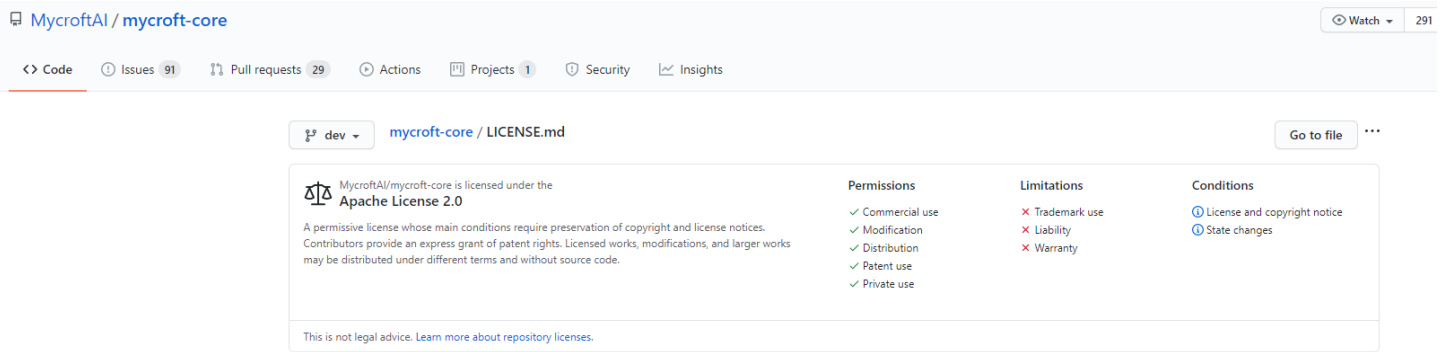
The populate.py file is used to set up a new profile. This file requires Gmail account details (users and password), details that never leave the device. This setup allows Jasper to read incoming emails and alert notifications. The Gmail account's password is not encrypted when it is stored; it is saved as plain text. Jasper also provides Facebook and Spotify configuration, again requesting login credentials that are stored in plain text on the device (Charlie Marsh, 2020).

Overall, JasperAI provides a very good foundation for a virtual assistant as it can execute the basic tasks, but on the other hand, it is an old implementation of a virtual assistant developed using Python2. Python 2 was deprecated on January 1<sup>st</sup>, 2020 (Python, 2020).

## MYCROFT

Mycroft is the world's most popular open-source voice assistant, completely customizable. The software is compatible with custom hardware, desktops, and Raspberry Pis and allows the developer to inspect, copy, modify, and contribute to the source code. The open-source project uses the opt-in privacy policy, recording the interaction with the virtual assistant only with the explicit consent of the user (Joshua Montgomery, Ryan Sipes , 2020).

Mycroft is developed using Python 3 and is protected by the Apache 2.0 license, requiring stating the changes a developer brings to the project and requires that the modified project to be released under the same license.



The screenshot shows the GitHub repository page for MycroftAI/mycroft-core. The file path is mycroft-core / LICENSE.md. The license is Apache License 2.0. The page includes a table with the following details:

Permissions	Limitations	Conditions
<ul style="list-style-type: none"><li>✓ Commercial use</li><li>✓ Modification</li><li>✓ Distribution</li><li>✓ Patent use</li><li>✓ Private use</li></ul>	<ul style="list-style-type: none"><li>✗ Trademark use</li><li>✗ Liability</li><li>✗ Warranty</li></ul>	<ul style="list-style-type: none"><li>Ⓞ License and copyright notice</li><li>Ⓞ State changes</li></ul>

Below the table, it states: "This is not legal advice. [Learn more about repository licenses.](#)"

**Figure 48 "Mycroft - Apache License"**

Source: <https://github.com/MycroftAI/mycroft-core/blob/dev/LICENSE.md>

Any process begins with the wake word, Mycroft using "Hey Mycroft", "Hey Ezra", "Hey Jarvis" and "Christopher". The assistant detected this word (who is continuously listening) and triggers Mycroft to start recording the audio input until it detects that the user stops talking. This process is implemented using a neural network trained to recognize the sound of the keyword. The Speech-to-Text package then analyzes the information through Mycroft servers from Google's Speech-to-Text. When the audio was successfully transformed into text, it is then parsed and processed to obtain the request's meaning. People can ask a question with one meaning in various ways: "What is the weather?", "How is the weather today?", "What is it like outside?", "Tell me the weather" and other ways, all with the same sense: today's weather. "Adapt" and "Padatious" are two projects that accomplish the same goal: parsing the text. "Adapt" which is implemented in Python, adapts a programmatic way of parsing, while Padatious is a neural-network parser based on AI with a broader perspective. Once a meaning had been extracted from the user's request, it is mapped with a "Skill" that knows how to execute the request. The last step is performed with the Speech-to-Text module that enables Mycroft to respond back to the user by transforming the answer (for the user's request) from text to voice (Coggeshall, 2020).

The open-source Mycroft is specifically designed to run on a Raspberry Pi 3 or 3B+ and PI's following models.

The digital assistant can be installed with Picroft OS that needs to be burned on an SD card and used as the Raspberry Pi default Operating System (Joshua Montgomery, Ryan Sipes , 2020).

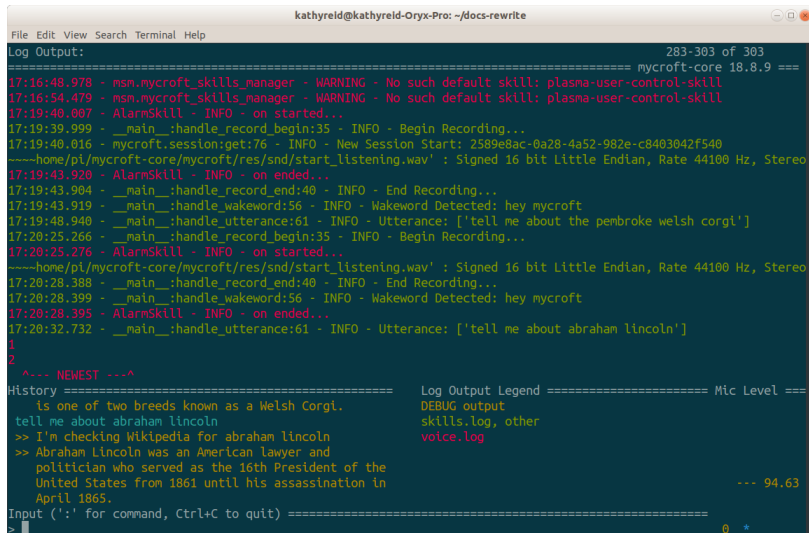
Other installation methods are available for Linux, macOS, Windows, and Android.

Raspberry Pi additional hardware requirements:

- Micro SD card 8GB minimum
- Analog Speaker
- USB microphone

Bluetooth alternatives of the speaker or microphone are not supported by default and might create difficulties in its usage.

## TESTING MYCROFT



```
kathyreid@kathyreid-Oryx-Pro: ~/docs-rewrite
File Edit View Search Terminal Help
Log Output: 283-303 of 303 mycroft-core 18.8.9 ==
17:16:48.978 - msn.mycroft_skills_manager - WARNING - No such default skill: plasma-user-control-skill
17:16:54.479 - msn.mycroft_skills_manager - WARNING - No such default skill: plasma-user-control-skill
17:19:40.007 - Alarmskill - INFO - on started...
17:19:39.999 - __main__:handle_record_begin:35 - INFO - Begin Recording...
17:19:40.016 - mycroft.session:get:76 - INFO - New Session Start: 2589e0ae-0a28-4a52-982e-cb403642f540
---home/pi/mycroft-core/mycroft/res/snd/start_listening.wav': Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
17:19:43.920 - Alarmskill - INFO - on ended...
17:19:43.904 - __main__:handle_record_end:40 - INFO - End Recording...
17:19:43.919 - __main__:handle_wakeword:56 - INFO - Wakeword Detected: hey mycroft
17:19:48.940 - __main__:handle_utterance:61 - INFO - Utterance: ['tell me about the pembroke welsh corgi']
17:20:25.266 - __main__:handle_record_begin:35 - INFO - Begin Recording...
17:20:25.276 - Alarmskill - INFO - on started...
---home/pi/mycroft-core/mycroft/res/snd/start_listening.wav': Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
17:20:28.388 - __main__:handle_record_end:40 - INFO - End Recording...
17:20:28.399 - __main__:handle_wakeword:56 - INFO - Wakeword Detected: hey mycroft
17:20:28.395 - Alarmskill - INFO - on ended...
17:20:32.732 - __main__:handle_utterance:61 - INFO - Utterance: ['tell me about abraham lincoln']
1
2
^--- NEWEST ---^
History ===== Log Output Legend ===== Mic Level ==
ls one of two breeds known as a Welsh Corgi.          DEBUG output
tell me about abraham lincoln                       skills.log, other
>> I'm checking Wikipedia for abraham lincoln        voice.log
>> Abraham Lincoln was an American lawyer and
politician who served as the 16th President of the
United States from 1861 until his assassination in
April 1865.                                          --- 94.63
Input (':' for command, Ctrl+C to quit) =====
>
```

Figure 49 “Picroft operating system”

Source: (Joshua Montgomery, Ryan Sipes , 2020)

The tests done with Mycroft for Linux is detailed below:

The Raspberry Pi tested the Mycroft version for Linux by cloning the project from Mycroft GitHub:

<https://github.com/MycroftAI/mycroft-core.git>.

Following the tutorial, the virtual environment dedicated to Mycroft was activated, and the setup was performed using their bash file from GitHub.

```
git clone https://github.com/MycroftAI/mycroft-core.git
cd mycroft-core
bash dev_setup.sh
```

The installation took about 15 minutes to finish and was followed by initiating the digital assistant with `./start-mycroft.sh all` command in the virtual environment provided by the source code. As this was an experiment to test Mycroft on the Raspberry Pi, no modifications of the code or the environment were performed.

Pairing the devices required to create an account on [www.mycroft.ai](http://www.mycroft.ai), and pair the Mycroft installed on the Raspberry Pi with the account.

Firstly, the Raspberry Pi used for this project was tested with the Picroft OS, and the interaction with the virtual assistant was a success, as seen in **Figure 49**.

Note that Picroft OS is a terminal-based operating system without a graphical user interface.

Secondly, as Raspbian OS is a Debian-based operating system, the hardware was also tested with the manual installation for Linux, which was also a success.



```

File Edit Tabs Help
Log Output: 17-29 of 29
===== mycroft-core 20.8.0 =====
----INFO | 1448 | QuestionsAnswersSkill | Searching for you can hear me now
Removing event fallback-query.mycroftai:QuestionQueryTimeout
Removing event fallback-query.mycroftai:QuestionQueryTimeout
Removing event fallback-query.mycroftai:QuestionQueryTimeout
Removing event fallback-query.mycroftai:QuestionQueryTimeout
----| INFO | 1448 | QuestionsAnswersSkill | Timeout occured check responses
----FO | 1454 | __main__:handle_wakeword:67 | Wakeword Detected: hey jarvis
----69 | INFO | 1454 | __main__:handle_record_begin:37 | Begin Recording...
----09.332 | INFO | 1454 | __main__:handle_record_end:45 | End Recording...
----INFO | 1454 | __main__:handle_utterance:72 | Utterance: ['how are you']
A--- NEWEST ---A
History ===== Log Output Legend ===== Mic Level =====
hi how are you DEBUG output 240 *
>> I am doing well, thank you skills.log, other *
you can hear me now voice.log *
>> Please rephrase your request. *
how are you *
>> I'm doing excellent -*- 35.52 *
Input (':' for command, Ctrl+C to quit) =====
>

```

Figure 50 "Mycroft debugging"

Source: Theodora Tataru, Raspberry Pi, 2020

Changing the microphone to reduce the environmental background noise, as it was thought that this could be the factor that influenced Mycroft to not respond to the wake word, did not change the Virtual Assistant's response accuracy.



Figure 52 "USB Microphone"

Source: Theodora Tataru 2020



Figure 51 "Game Recording Microphone"

Source: Theodora Tataru 2020

As seen in **Figures 51** and **52**, two microphones were used to test Mycroft on Raspberry Pi.

The microphone seen in **Figure 52** is a basic USB microphone that does not have any background noise cancellation (Amazon, 2020). Therefore, the second experiment was done using the microphone seen in **Figure 51**, a high-quality microphone using sensitivity: -47db +/- 3dB, designed for computer game recording (Sunsy, 2020).

The test's purpose was to observe how Mycroft behaves on the Raspberry Pi device, and the result could be classified as most successful.

Before interacting with Mycroft, some microphone and speaker tests were made, outside the Mycroft environment, followed by audio testing with the digital assistant using `./start-mycroft.sh audiotest` command, resulting in a partial success and failure.

The digital assistant was prompt with requests but was responding just 1 in 5 times to the wake word.

Additional microphone tests were done, and all of them passed to some extent, as the wake-up word was not always intercepted. This resulted in more confusion for not finding a reason why the wake word is not captured, even if the microphone was intercepting all the sounds.

There exist multiple digital assistance called J.A.R.V.I.S. AI (Just A Rather Very Intelligent System) that copy the functional artificial intelligence character from the Marvel Cinematic Universe from Iron Man. JARVIS is the very close personal assistant of Tony Stark (Iron Man), the movie star (Wikipedia, 2020). The required computing power for the neural networks, storage, and other resources that Jarvis (the Marvel Character) requires, at the moment, does not exist. However, this type of superior intelligence in computers will most likely be implemented in the future (Jr., 2015).

Even the standard of the functional character from the Marvel Universe standard is impossible to achieve; at least 12 open-source projects are based on the movie character (Source, 2020).

The research was focused on one open-source project from the Python libraries. The package can be found on GitHub and is under the MIT license allowing developers to freely use, modify and contribute to the project (Paul, 2020).

The project was developed in Python 3, and the set of skills of the virtual assistant are the fundamental ones:

- Date and time
- Newsfeed
- Send emails
- Quote from Wikipedia or other websites
- Weather forecast
- Open websites

Several dependencies are required to start using Jarvis AI, such as:

- **gtts** is a Python library and CLI tool to the Google-Translate text-to-speech API (pypi, 2020)
- **playsound** is another module from Python used for playing sounds (pypi, 2020)
- **SpeechRecognition** is an online/offline library for speech recognition (pypi, 2020)
- **Lxml** is a pythonic library for processing XML and HTML (Team, 2020)
- **beautifulsoup4** is a library that helps to scrape information from web pages (pypi, 2020)
- **wikipedia** is another module from the Python ecosystem that makes the parsing of the Wikipedia webpage easy to access (pypi, 2020)

Jarvis installation can be performed in two ways:

1. cloning the GitHub repository
2. install the Jarvis AI module

For the testing purposes, Jarvis AI was installed on the Raspberry PI via pip, and if the conclusion is to use Jarvis AI as the base of this project, the installation will be performed cloning the project from GitHub.

The package's installation was challenging, as Jarvis uses a different OpenCV version from the version installed on the system, resulting in `segmentation_fault` when importing the OpenCV module. A segmentation fault is a generic error with multiple possible reasons

- When a program prompts to access a memory location is not allowed to access
- Low memory
- Faulty RAM memory
- Accessing a huge data set
- Wrong query
- Buggy code
- Multiple recursion (Wikipedia, 2020)

The source of the error, as expected, was the OpenCV version, which needed to be downgraded for JasperAI to work.

The first test done was to test if speech-to-text functionality from Jarvis AI works, and the result was successful, as shown in **Figure 53**.

```
pi@raspberrypi: ~/Desktop/Jarvis_AI/JarvisAI/JarvisAI
File Edit Tabs Help
ALSA lib conf.c:5036:(snd_config_expand) Args evaluate error: No such file or di
rectory
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM bluealsa
ALSA lib pcm_oss.c:377:(snd_pcm_oss_open) Unknown field port
ALSA lib pcm_oss.c:377:(snd_pcm_oss_open) Unknown field port
ALSA lib pcm_a52.c:823:(snd_pcm_a52_open) a52 is only for playback
ALSA lib confmisc.c:1281:(snd_func_refer) Unable to find definition 'cards.bcm28
35_hdmi.pcm.iec958.0:CARD=0,AES0=6,AES1=130,AES2=0,AES3=2'
ALSA lib conf.c:4568:(snd_config_evaluate) function snd_func_refer returned err
or: No such file or directory
ALSA lib conf.c:5047:(snd_config_expand) Evaluate error: No such file or directo
ry
ALSA lib pcm.c:2565:(snd_pcm_open_noupdate) Unknown PCM iec958:{AES0 0x6 AES1 0x
32 AES2 0x0 AES3 0x2 CARD 0}
ALSA lib pcm_usb_stream.c:486:(snd_pcm_usb_stream_open) Invalid type for card
ALSA lib pcm_usb_stream.c:486:(snd_pcm_usb_stream_open) Invalid type for card
ALSA lib pcm_hw.c:1903:(snd_pcm_hw_open) card is not defined
ALSA lib pcm_hw.c:1903:(snd_pcm_hw_open) card is not defined
Say something...
You said: just checking into the microphone works

>>> print(res)
just checking into the microphone works
>>>
```

Figure 53 “Microphone test – Jarvis AI”

Source: Theodora Tataru 2020

Besides the challenges encountered when installing the module, Jarvis AI presented well, performing its tasks as a virtual assistant. Again, the project brings the developer the fundamental parts of a virtual assistant, but further additions and modifications are required.

---

After careful consideration, the decision to create a virtual assistant from scratch was taken. This decision was based on different conclusions after testing the open-source projects. All the open-source virtual assistants are accurate and could present a good base, but all projects have functionalities that would not be needed for the users this project targets.

## TEXT TO SPEECH

Speech is a distinctive feature of humans, who are the only creatures in the world that use this channel to communicate. It is no surprise that today's technologies are based on language and speech (Dutoit, 1997).

The ultimate goal of text-to-speech (abbreviated TTS) is to “read” any digital text and transform it into understandable sounds for humans: speech. The output should be smooth, coherent, and natural. Artificial speech is obtained by concatenating isolated letters, words, and sentence parts (Dutoit, 1997).

The main difference between machines and humans is that humans are self-conscious about reading rules, while machines are not. This factor makes our speech smooth and natural, while machines sound abrupt (Dutoit, 1997).

TTS is a type of assistive technology (a technology-specific that helps people work around challenges) that reads digital text aloud. This is obtained by transforming text into an audio output. The voice of TTS is computer-generated, and the user can alter its speed. The voice quality varies; some voices might sound robotic, while others sound natural (Team, 2020).

TTS technologies are used for a virtual assistant to transform the assistant's answer from text to voice, to deliver the answers to the user vocally.

TTS technologies are used by important companies such as Google (Google Assistant), Apple (Siri), Microsoft (Cortona), Amazon (Alexa), and others. All these products were described briefly in the “Virtual Assistant” section.

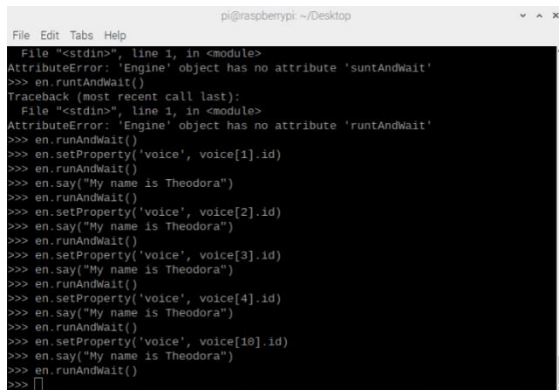
---

### PYTTSX3

Pytttsx3 is a text-to-speech library in Python.

For testing purposes, the library was installed on the Raspberry Pi using the `pip3 install pytttsx3` command. And the library was tested with the following code:

```
import pytttsx3
engine = pytttsx3.init()
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[0].id)
engine.say(text)
engine.runAndWait()
(pypi, 2020)
```



```
File Edit Tabs Help
pi@raspberrypi ~/Desktop
File "<stdin>", line 1, in <module>
AttributeError: 'Engine' object has no attribute 'runAndWait'
>>> en.runAndWait()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Engine' object has no attribute 'runAndWait'
>>> en.runAndWait()
>>> en.setProperty('voice', voice[1].id)
>>> en.runAndWait()
>>> en.say("My name is Theodora")
>>> en.runAndWait()
>>> en.setProperty('voice', voice[2].id)
>>> en.say("My name is Theodora")
>>> en.runAndWait()
>>> en.setProperty('voice', voice[3].id)
>>> en.say("My name is Theodora")
>>> en.runAndWait()
>>> en.setProperty('voice', voice[4].id)
>>> en.say("My name is Theodora")
>>> en.runAndWait()
>>> en.setProperty('voice', voice[10].id)
>>> en.say("My name is Theodora")
>>> en.runAndWait()
>>> []
```

As seen in **Figure 54**, the library was tested with the same input string and different voices. The result was a success as the text was transposed into audio output using a masculine voice, but the sound was very abrupt and unnatural.

**Figure 54** “Testing TTS pytttsx3”

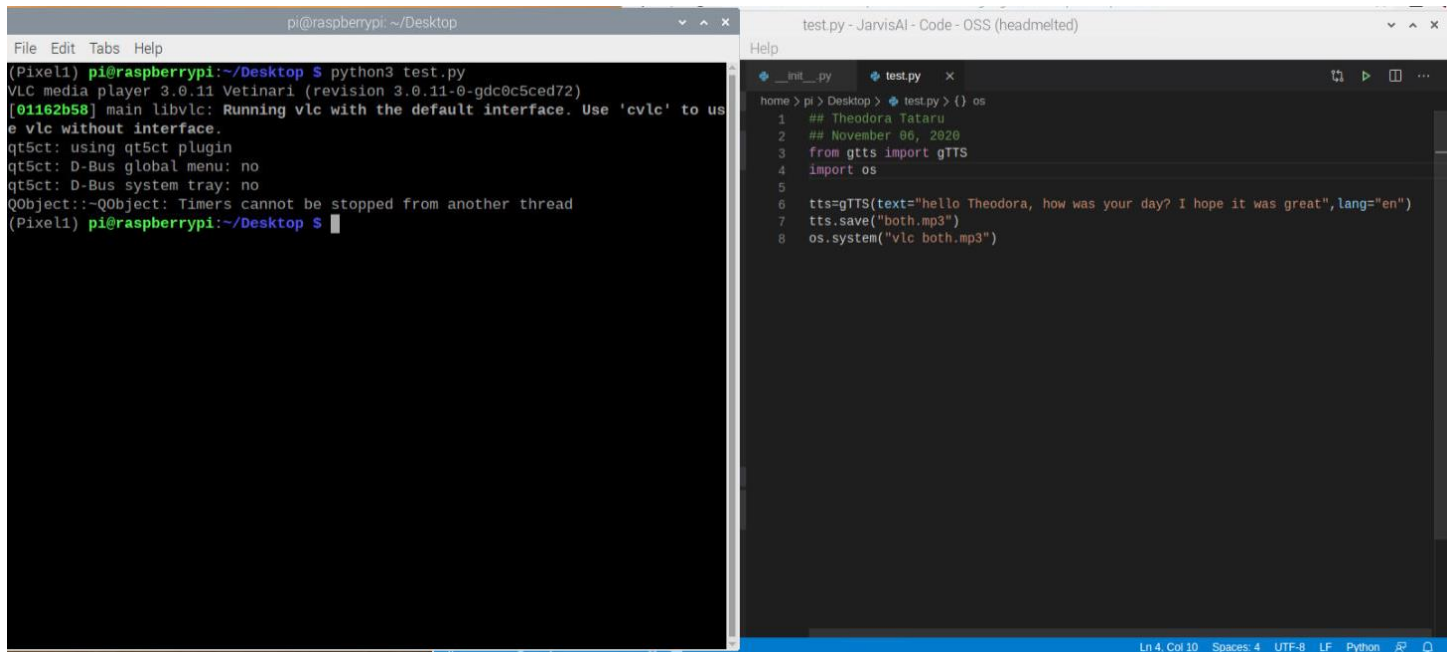
Source: Theodora Tataru, 2020

---

## GTTS

Google text-to-speech (abbreviated gTTS) is another python library that servers as an interface with Google Translate text-to-speech API (pypi, 2020).

The library was installed using the command `pip3 install gTTS` and was tested using the code seen in **Figure 55**.



The image shows two windows side-by-side. The left window is a terminal on a Raspberry Pi. It shows the command `python3 test.py` being executed. The output includes VLC media player version information, a warning about the interface, and qt5ct configuration messages. The prompt returns to `pi@raspberrypi:~/Desktop $`. The right window is a code editor showing the contents of `test.py`. The code imports `gTTS` and `os`, then uses `gTTS` to generate speech for a specific text and saves it as `both.mp3`, which is then played using `os.system`.

**Figure 55 "Testing gTTS"**  
Source: Theodora Tataru, 2020

The test was a success. The library uses the Google Translate voice, which is a feminine voice. The test resulted in a voice output that sounds more natural in contrast with pyttsx.

---

## OTHER TTS LIBRARIES AND OPEN-PROJECTS

Numerous TTS open-source projects are available for developers to explore, implement and modify. According to awesomeopensource.com, there are over 50 open-source projects available (Team, 2020).

- *Tacotron*

Tacotron provides a TTS implementation of speech synthesis using TensorFlow. The project requires multiple modules to be installed, such as TensorFlow, NumPy, falcon, matplotlib, and others (Ito, 2020).

To use their pre-trained model, the following command must be executed in the terminal `python3 demo_server.py --checkpoint /tmp/tacotron-20180906/model.ckpt`. Then a browser needs to be opened and directed to localhost:9000, where the developer can insert text that is processed to speech (Ito, 2020).

To train the model, a dataset must be used. The data must be processed and then fed to the model. Both actions are implemented in the Tacotron, and scripts are provided just to be run by the developer (Ito, 2020).

- *Transformers*

The Transformers project is based on two research papers (Springer, 2020):

- "Neural Speech Synthesis with Transformer Network", by Naihuan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, Ming Liu, Ming Zhou, Available at <https://arxiv.org/abs/1809.08895v3>
- "FastSpeech: Fast, Robust and Controllable Text to Speech", by Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, Tie-Yan Liu, Available at <https://arxiv.org/abs/1905.09263>

The project delivers a TTS that is:

- robust – no repeats in challenging sentences
- fast - with no autoregression, the predictions take a fraction of time
- controllable – the speed of speech is controllable (Springer, 2020)

An audio sample of the Transformers project can be found here: <https://as-ideas.github.io/TransformerTTS/>.

- *Other open-source projects:*
  - Lingvo, Available at <https://github.com/tensorflow/lingvo>
  - Aeneas, Available at <https://github.com/readbeyond/aeneas>
  - Deepvoice3\_pytorch, Available at [https://github.com/r9y9/deepvoice3\\_pytorch](https://github.com/r9y9/deepvoice3_pytorch)
  - Athena, Available at <https://github.com/athena-team/athena>
  - Android Speech, Available at <https://github.com/gotev/android-speech>
  - And others, Full list available at <https://awesomeopensource.com/projects/tts>

## SPEECH TO TEXT

Speech-to-Text (abbreviated STT) is the action of transforming human language into digital text.

When humans speak, a series of vibrations are generated, translated into digital language by an analogue-to-digital converter. The conversion is performed by taking very detailed measurements of these vibrations and distinguishing the relevant sounds and different frequencies. For this process to be successful, the speech's speed and volume are adjusted before processing (Note, 2020).

<b>s</b> sat	<b>t</b> tap	<b>p</b> pan	<b>n</b> nose	<b>m</b> mat	<b>a</b> ant	<b>e</b> egg	<b>i</b> ink	<b>o</b> otter
<b>g</b> goat	<b>d</b> dog	<b>ck</b> click	<b>r</b> run	<b>h</b> hat	<b>u</b> up	<b>ai</b> rain	<b>ee</b> knee	<b>igh</b> light
<b>b</b> bus	<b>f</b> farm	<b>l</b> lolly	<b>j</b> jam	<b>v</b> van	<b>oa</b> boat	<b>oo</b> cook	<b>oo</b> boot	<b>ar</b> star
<b>w</b> wish	<b>x</b> axe	<b>y</b> yell	<b>z</b> zap	<b>qu</b> quill	<b>or</b> fork	<b>ur</b> burn	<b>ow</b> now	<b>oi</b> boil
<b>ch</b> chin	<b>sh</b> ship	<b>th</b> think	<b>th</b> the	<b>ng</b> sing	<b>ear</b> near	<b>air</b> stair	<b>ure</b> sure	<b>er</b> writer

Phonemes are the smallest unit of sound, used to distinguish one word from another in any language. All the English phonemes are shown in **Figure 56**.

Figure 56 "English Phonemes"

Source: (TheSchoolRun, 2020)

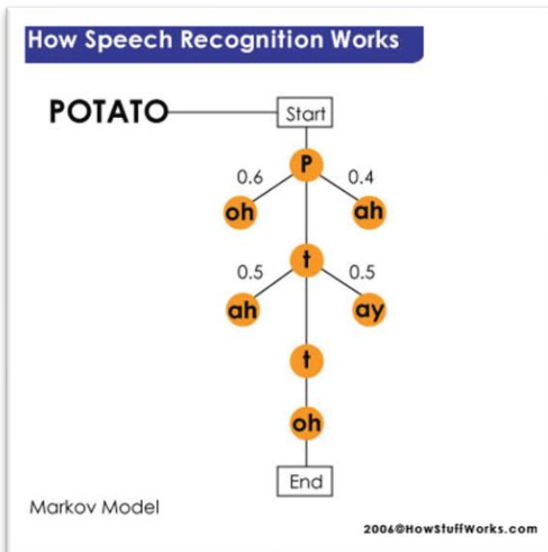


Figure 57 "Evaluating Phonemes"

Source: (Grabianowski, 2020)

The next step in processing speech involves segmenting the received signal into thousandths of seconds and match these segments to phonemes. Each match is then inspected and evaluated in relation to the phonemes around it through a complicated mathematical model to compare each match to known sentences and words.

The model returns a text on what "it believes" the users said (Note, 2020).

The most used speech recognition models are the Hidden Markov Model and Neural Networks (Grabianowski, 2020).

---

## HIDDEN MARKOV MODEL

The Hidden Markov Model is the most used in speech recognition. The model treats each phoneme as a link in a chain, where the complete chain is a word. This chain breaks in multiple directions as the model tries to match the sound with the phonemes that are most likely to come next. During this process, the model creates a probability score for each phoneme based on its training (Grabianowski, 2020).

The process becomes more complex when phrases and sentences are processed, as the system has to figure out where each sentence or word begins and ends. If the model has a vocabulary as large as 60,000 words, a sentence of 3 words could be any of 216 trillion possibilities. With all the computing power existing today, the model cannot traverse all the possibilities without help. Therefore, after developers trained the model, the training process does not end, as the user also trains the model while using the software (Grabianowski, 2020).

Usually, when a device is set up to recognize speech, the user needs to spend about 10 minutes to train the model to recognize the particular way the intended user communicates through voice.



KEYWORDS

Trigger Words	Count	% of Total
how	658,976	8.64%
what	382,224	5.01%
best	200,206	2.63%
the	75,025	0.98%
is	53,496	0.70%
where	43,178	0.57%
can	42,757	0.56%
top	42,277	0.55%
easy	31,178	0.41%
when	27,571	0.36%
why	25,980	0.34%
who	24,930	0.33%
new	24,779	0.33%
recipe	22,967	0.30%
good	22,807	0.30%
homes	21,132	0.28%
make	19,774	0.26%
does	19,449	0.26%
define	19,375	0.25%
free	18,315	0.24%
i	18,245	0.24%
list	17,136	0.22%
home	17,118	0.22%
types	16,575	0.22%
do	16,448	0.22%

The top 25 keywords used by users to interact with a virtual assistant can be observed in **Figure 58**.

These words are the most used by users to request information from a smart virtual assistant.

Figure 58 "Top 25 - keywords for smart speakers"  
Source: (Andersen, 2019)

## MAGIC/SMART MIRROR

The technology has advanced so much that it had come to the point where nearly every single device used on daily basics has become a smart device, such as Smart Phones, TVs, watches, speakers, lights, sockets, and others.

One of today's most elegant technologies is the Smart Mirror; A classical mirror combined with technology to bring more functionality to people's homes.

Most households' common mirror is a one-way mirror with the inner side completely covered with a dark mate color that reflects 100% of the light, reflecting back any objects in front of it. The mirror required for this project is a mirror that reflects light from one direction and allows light to pass from the other direction, called a two-way acrylic mirror (Rabideau, 2020).

A Smart Mirror looks like a classical mirror but has a display installed at its back. When the display is active, it can project on the mirror surface information such as time, newsfeed, weather, appointments, and many more.



Figure 59 "Magic Mirror - Fully Installed"

Source: (Horsey, 2019)

This technology uses a display positioned behind the two-way acrylic mirror and connects to other devices such as phones, Alexa, Raspberry Pi, and others. It is often called a Magic Mirror, being associated with the famous mirror from the "Snow White" tale (Rabideau, 2020).

The Smart Mirror can be personalized on what information to display and can be decorated with smart lights on the frame, fulfilling users' needs on choosing the color and warmth.

The smart mirror has three main components: a display, a two-way acrylic mirror, and a computer.

The light pixels from the display installed behind the acrylic mirror are permitted to pass through and give the magic effect of text appearing on the mirror's surface. In accordance with this project, the display used by the Raspberry Pi to show in text the result of the user requirements can be encapsulated in a Magic Mirror.

The hardest part of building the frame that holds all the hardware is the actual handy work of building the structure, which can be made from scratch, or it can be directly bought once the size of the display is known (Gartenberg, 2017).

On the Smart Mirror software side, the base project is an open-source project that developers can use or modified for free will. The project is protected by the MIT license, which allows full use of the project for any individual (Teeuw, 2020).

Also, creative people and developers can create their own software and use a two-way acrylic mirror to create a fairy tale piece of furniture.

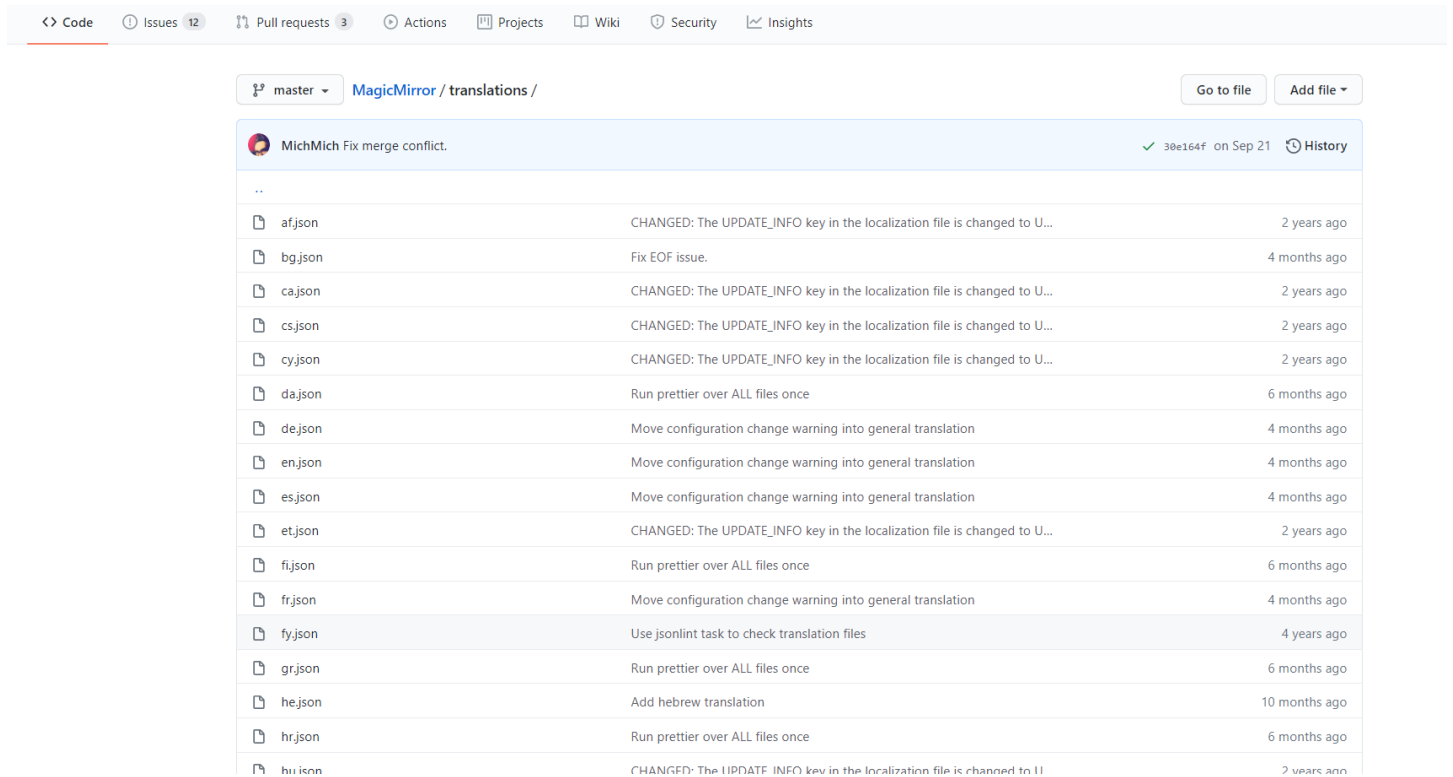
The Magic Mirror was designed to run its software on a Raspberry Pi (from model 2 upwards), but it can run on other devices as well.

The open-source project testing starts with cloning the project from the GitHub page and installing NodeJS module on the Operating System. Once the repository was cloned, the software is installed using the `npm install` command from the terminal. After a successful installation, the application is started using the `npm run service`, which starts the mirror on the Raspberry Pi's output display (Mirror, 2020).

The open-source project comes with multiple functionalities such as:

- Alerts
- Calendar
- Clock
- Compliments
- Weather
- Newsfeed (Teeuw, 2020)

All the Magic Mirror functionalities are available in over 30 languages such as English, German, French, Romanian, and many others, as seen in **Figure 60**.

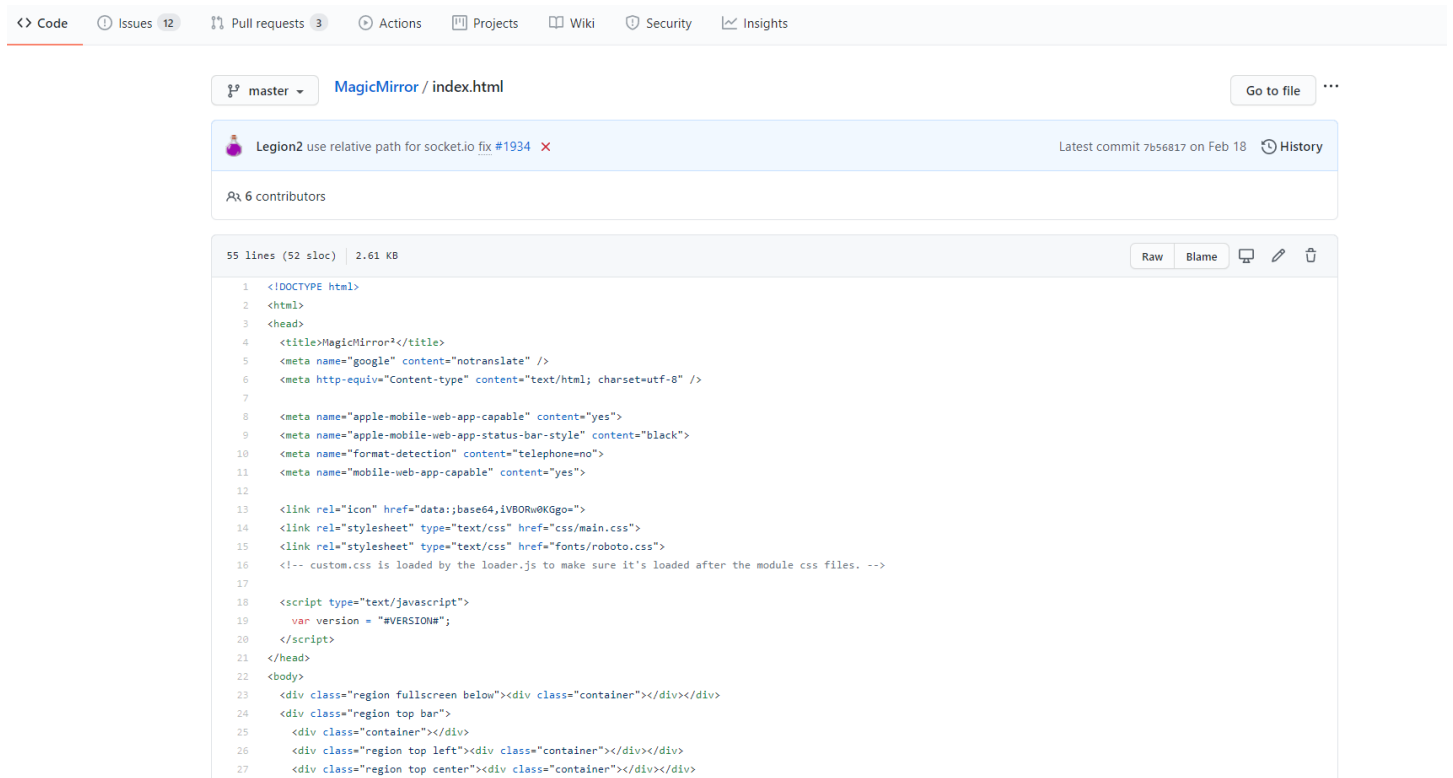


**Figure 60 "Magic Mirror - Languages Library"**

Source: (Teeuw, 2020)

The base of the Magic Mirror is HTML, CSS, JSON, and JavaScript.

The main page displayed to the user is an HTML page processed by Java Script and decorated with CSS, as seen in **Figure 61**.



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>MagicMirror</title>
5 <meta name="google" content="notranslate" />
6 <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
7
8 <meta name="apple-mobile-web-app-capable" content="yes">
9 <meta name="apple-mobile-web-app-status-bar-style" content="black">
10 <meta name="format-detection" content="telephone=no">
11 <meta name="mobile-web-app-capable" content="yes">
12
13 <link rel="icon" href="data:base64,iVBORw0KGgo">
14 <link rel="stylesheet" type="text/css" href="css/main.css">
15 <link rel="stylesheet" type="text/css" href="fonts/roboto.css">
16 <!-- custom.css is loaded by the loader.js to make sure it's loaded after the module css files. -->
17
18 <script type="text/javascript">
19   var version = "#VERSION#";
20 </script>
21 </head>
22 <body>
23 <div class="region fullscreen below"><div class="container"></div></div>
24 <div class="region top bar">
25   <div class="container"></div>
26 <div class="region top left"><div class="container"></div></div>
27 <div class="region top center"><div class="container"></div></div>
```

**Figure 61 "Magic Mirror - Home Page"**

Source: (Teeuw, 2020)

To use the Magic Mirror for this project, the source code of the software needs to be modified in such a way to display generic information such as weather, news, time, and date at the same time with the output of the virtual assistant.



As seen in **Figure 62**, on the back of the acrylic mirror is the hardware required to run the software and the display, and on the front, the light pixels of the display passes through the two-way acrylic mirror, giving the effect that the text displayed is shown on the surface of the mirror.

**Figure 62 "Magic Mirror - Under the frame"**

Source: (Hobson, 2016)

Magic Mirror's concept impact over "Pixel" the virtual assistant would be significant, as the users would naturally engage with the digital assistant while standing in front of the mirror. However, the open-source project will not be used for this project, but it presents the base source of inspiration.

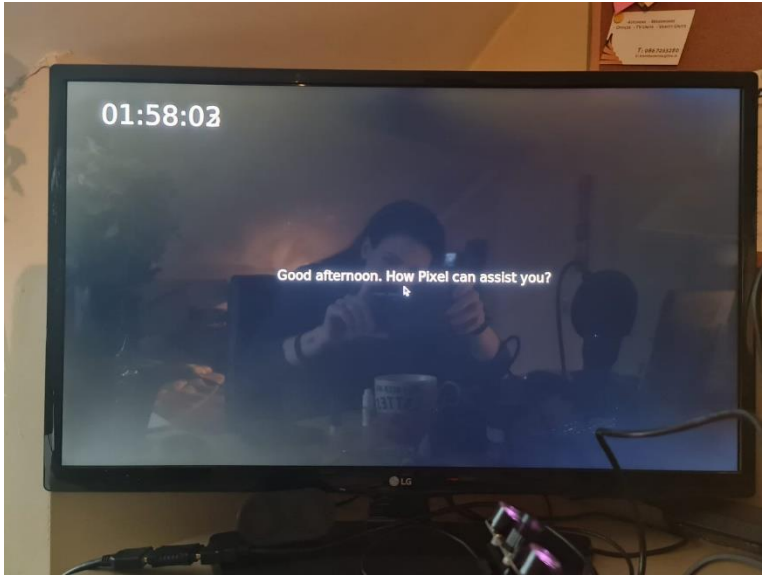
---

## TKINTER

As mentioned in the section above, the open-source Magic Mirror project will be used as an inspiration, but another interface will be created using Tkinter.

Tkinter is a Python standard GUI package.

One basic implementation of the interface was designed using this package, as seen in **Figure 63**.



**Figure 63 "Tkinter GUI"**

Source: Theodora Tataru, 2020

Once the interface was initialized, the AI assistance was going on standby, waiting for the interface process to finish its execution before proceeding. This action needed to be rectified, as the interface needs to be always on at the same time as the virtual assistant.

---

## THREADING

The virtual assistant was instantiated as a thread using the “threading” Python package. The test was a success, as the vocal assistance was starting at the same time with the interface, but a slight delay could be felt in the virtual assistant actions: listening, processing, and returning the answer to the user’s query.

In this case, the threads started were using the same memory space, sharing the resources.

---

## MULTIPROCESSING

The same action was repeated using the multiprocessing Python package.

In this case, each process was using its own memory addresses and was not sharing resources. Therefore, the processes were able to run in parallel, and the virtual assistant's delay was significantly reduced.

Tkinter was chosen from other various libraries that could be used to design the interface because the library is very basic. As the interface needed for “Pixel” virtual assistant is as well as simple as possible, using a black background and white text.

As the black color does not reflect light and the white color reflects the most amount of light from the color spectrum, only the white text is reflected on the two-way mirror surface. This action gives the user the illusion that text appears on the actual mirror.

## CONCLUSION

In this research document, several hardware and software technologies were researched with the purpose of developing a virtual assistant that will impact a greater segment of the market, including people with hearing deficiency and people with poor computer knowledge.

For this project's hardware side, research was conducted for Raspberry Pi and additional compatible components such as servos, different Raspberry Pi hats, camera, speakers, microphone, temperature/humidity sensor, two-way-acrylic mirrors, wooden frame, GSM/GPS module and others.

Various technologies were researched for the software side of the project, such as artificial intelligence, machine learning, facial detection, facial recognition, natural language processing, magic mirror, and others.

The software and hardware researched serve to develop a different kind of virtual assistant: an assistant that will recognize the user by its face and display the information required on a mirror surface.

### *Hardware*

Several tests were done on the Raspberry Pi to determine which technologies would serve the project the best, and the following conclusions were determined:

- The best Raspberry Pi for this project is the model 4B, as it is the only version that comes with more than 1GB of RAM, and the project requires at least 4GB of RAM to implement facial recognition
- The camera that would suit the best this project is the infrared camera from MakerHawk, as it can operate in low light conditions
- With different tests performed, the servos' sudden moves were persistent. Therefore, the feature that moves the camera to track the user's face will be implemented if the time permits; meanwhile, the development focuses on the device's core functionalities: virtual assistant, face recognition, skills, and the interface.
- The intended screen for the project is a 10-inch screen that will be mounted behind a two-way mirror, giving the user the illusion that the graphics appear on the mirror's surface. This trick will keep users of any age focused and engaged, as they will perceive the display as a standard mirror with "magic" properties.

### *Software*

This side of the project is most likely to change, but at this moment in time, the following conclusion had been terminated:

- Python is the language that will be used to program the virtual assistant, as it is a powerful language with excellent functionality for Raspberry Pi and Computer Vision
- OpenCV library would suit the project perfectly, as it provides pre-trained models for facial detection and recognition. The use of such a library is imperative, as a facial recognition model cannot be trained on Raspberry Pi in the time frame required for this project
- gTTS will be used to give voice to the device, as the voice is more natural compare with other text-to-speech libraries
- speech\_recognition will be used to transform speech-to-text
- Dlib module is a perfect fit for the project, as several tests were performed on the Raspberry Pi, and the Dlib library passed the test significantly better than other libraries
- The virtual assistant that processes the user requirements and returns meaningful information will be designed from scratch and is inspired by several open-source assistants
- The interface will be created from scratch, as the open-source project "Magic Mirror" is using some libraries irrelevant for this project. The target audience needs a clear, intuitive, and clean interface.

The development of "Pixel" virtual assistant is a complex process, and during its development evolution, the project might shift from these conclusions, and further research might be necessary.

## BIBLIOGRAPHY

Agarwal, S., 2013. *Machine Learning: A very quick introduction*, s.l.: Ubuntu.

AgeAction, 2020. *Supporting digital Literacy among older people*, Dublin: s.n.

Ajitsaria, A., 2020. *What Is the Python Global Interpreter Lock (GIL)?*. [Online]

Available at: <https://realpython.com/python-gil/#:~:text=The%20Python%20Global%20Interpreter%20Lock%20or%20GIL%2C%20in%20simple%20words,at%20any%20point%20in%20time.>

[Accessed 04 January 2021].

Alexa, A., 2020. *Display and Behavior Specifications for Alexa-Enabled Devices with a Screen*. [Online]

Available at: <https://developer.amazon.com/en-US/docs/alexa/custom-skills/display-and-behavior-specifications-for-alexa-enabled-devices-with-a-screen.html>

[Accessed 13 November 2020].

Alpaydin, E., 2020. *Introduction to Machine Learning*. Fourth Edition ed. Massachusetts: Library of Congress Cataloging-in-Publication Data.

Amazon, 2020. *Raspberry Pi 10 inch Touch Screen - SunFounder 10.1" HDMI 1280x800 IPS LCD Touchscreen*. [Online]

Available at: <https://www.amazon.co.uk/dp/B0776VNW9C/?coliid=IY90HMKUKF7Q5&colid=24EX12HEGL4VT&th=1>

[Accessed 07 December 2020].

Amazon, 2020. *Richera USB 2.0 Mini Microphone Makio Mic for Laptop,Desktop PCs Notebook,MSN,Skype,VOIP,Voice Recognition Software*. [Online]

Available at: [https://www.amazon.co.uk/gp/product/B01FJWO5K4/ref=ppx\\_yo\\_dt\\_b\\_asin\\_image\\_o04\\_s03?ie=UTF8&psc=1](https://www.amazon.co.uk/gp/product/B01FJWO5K4/ref=ppx_yo_dt_b_asin_image_o04_s03?ie=UTF8&psc=1)

[Accessed 04 November 2020].

Amazon, 2021. *IBest WM8960 Hi-Fi Sound Card HAT Audio Module for Raspberry Pi Supports Stereo Encoding/Decoding Hi-Fi Playing/Recording directly Drive Speakers to Play Music,I2S I2C Interface*. [Online]

Available at: [https://www.amazon.co.uk/gp/product/B07R8M3XFQ/ref=ppx\\_yo\\_dt\\_b\\_asin\\_title\\_o09\\_s00?ie=UTF8&psc=1](https://www.amazon.co.uk/gp/product/B07R8M3XFQ/ref=ppx_yo_dt_b_asin_title_o09_s00?ie=UTF8&psc=1)

[Accessed 29 March 2021].

Amazon, 2021. *Waveshare Raspberry Pi GSM/GPRS/GNSS Bluetooth HAT Expansion Board GPS Module with Low-Power Consumption Based on SIM868 Compatible With Raspberry Pi 2B 3B 4B Zero*. [Online]

Available at: [https://www.amazon.co.uk/gp/product/B076CPX4NN/ref=ppx\\_yo\\_dt\\_b\\_asin\\_image\\_o01\\_s00?ie=UTF8&psc=1](https://www.amazon.co.uk/gp/product/B076CPX4NN/ref=ppx_yo_dt_b_asin_image_o01_s00?ie=UTF8&psc=1)

[Accessed 17 March 2021].

Andersen, D., 2019. *26 Voice Search Stats Marketers Need to Know in 2020*. [Online]

Available at: <https://www.dialogtech.com/blog/voice-search-statistics/>

[Accessed 04 January 2021].

Andrews, E. L., 2020. *An Open-Source Challenger to Popular Virtual Assistants*. [Online]

Available at: <https://hai.stanford.edu/blog/open-source-challenger-popular-virtual-assistants>

[Accessed 4 November 2020].

Babich, N., 2020. *What Is Computer Vision & How Does it Work? An Introduction*. [Online]

Available at: <https://xd.adobe.com/ideas/principles/emerging-technology/what-is-computer-vision-how-does-it-work/#:~:text=Computer%20vision%20is%20the%20field,pixel%20level%20and%20understand%20it.>

[Accessed 13 November 2020].

Bridget Botelho, Margare Rouse, 2020. *Virtual Assistant (AI Assistant)*. [Online]

Available at: <https://searchcustomerexperience.techtarget.com/definition/virtual-assistant-AI-assistant#:~:text=A%20virtual%20assistant%2C%20also%20called,completes%20tasks%20for%20the%20user>

[Accessed 14 October 2020].

- Brown, J., 2010. *Multiprocessing vs Threading Python [duplicate]*. [Online]  
Available at: <https://stackoverflow.com/questions/3044580/multiprocessing-vs-threading-python>  
[Accessed 4 January 2021].
- Charlie Marsh, J. H. S. S., 2020. *Jasper AI*. [Online]  
Available at: <https://jasperproject.github.io/>  
[Accessed 2020 October 29].
- Coggeshall, J., 2020. *Mycroft: an open-source voice assistant*. [Online]  
Available at: <https://lwn.net/Articles/826625/>  
[Accessed 31 October 2020].
- Crist, R., 2018. *Amazon's Echo Show makes Alexa more accessible to the deaf and speech-impaired*. [Online]  
Available at: <https://www.cnet.com/news/amazon-tap-to-alexa-accessibility-feature/>  
[Accessed 13 October 2020].
- DC, 2018. *PID Made Simple*. [Online]  
Available at: <https://pidexplained.com/what-is-pid/>  
[Accessed 24 October 2020].
- DLib, 2018-2020. *Software/Dlib*. [Online]  
Available at: <https://computervisiononline.com/software/1105138543>  
[Accessed 17 October 2020].
- Dutoit, T., 1997. *An Introduction to Text-to-Speech Synthesis*. 1st ed. s.l.:Kluwer Academic Publishers.
- Elecrow, 2020. *ELECROW 5 Inch Touch Screen HDMI Monitor HD 800x480 TFT LCD Display for Raspberry Pi 2B B+ Raspberry Pi 3B*. [Online]  
Available at: [https://www.amazon.co.uk/gp/product/B013JECYF2/ref=ppx\\_yo\\_dt\\_b\\_asin\\_image\\_o00\\_s02?ie=UTF8&psc=1](https://www.amazon.co.uk/gp/product/B013JECYF2/ref=ppx_yo_dt_b_asin_image_o00_s02?ie=UTF8&psc=1)  
[Accessed 03 October 2020].
- Electronics, D.-K., 2019. *Raspberry Pi Model 4 B*. [Online]  
Available at: [https://www.digikey.ie/en/product-highlight/r/raspberry-pi/raspberry-pi-4-model-b?utm\\_adgroup=General&utm\\_source=google&utm\\_medium=cpc&utm\\_campaign=Dynamic%20Search EN RLSA Product Purchaser &utm\\_term=&productid=&gclid=CjwKCAjw5p\\_8BRBUeIwAPpJO6\\_R0E1j9VUEsm](https://www.digikey.ie/en/product-highlight/r/raspberry-pi/raspberry-pi-4-model-b?utm_adgroup=General&utm_source=google&utm_medium=cpc&utm_campaign=Dynamic%20Search%20EN%20RLSA%20Product%20Purchaser&utm_term=&productid=&gclid=CjwKCAjw5p_8BRBUeIwAPpJO6_R0E1j9VUEsm)  
[Accessed 15 October 2020].
- Electronics, D.-K., 2020. *RASPBERRY PI 3*. [Online]  
Available at: <https://www.digikey.ie/product-detail/en/raspberry-pi/RASPBERRY-PI-3/1690-1000-ND/6152799>  
[Accessed 15 October 2020].
- Europe, T., 2021. *GSM + GPS/Glonass Compound SMT Module*. [Online]  
Available at: <https://www.texim-europe.com/product/SIM868-SMM>  
[Accessed 17 March 2021].
- Garbade, D. M. J., 2018. *A Simple Introduction to Natural Language Processing*. [Online]  
Available at: <https://becominghuman.ai/a-simple-introduction-to-natural-language-processing-ea66a1747b32>  
[Accessed 13 November 2020].
- Garcia, L., 2020. *The 13 best personal assistant apps*. [Online]  
Available at: <https://www.care.com/c/stories/15112/personal-assistant-app/>  
[Accessed 27 October 2020].
- Gartenberg, C., 2017. *Building your own smart mirror is surprisingly easy*. [Online]  
Available at: <https://www.theverge.com/circuitbreaker/2017/8/17/16158104/smart-mirror-diy-raspberry-pi-commute-weather-time-gadget>  
[Accessed 07 November 2020].



Google, 2020. *Hey Google*. [Online]  
Available at: <https://assistant.google.com/>  
[Accessed 27 October 2020].

Grabianowski, E., 2020. *How Speech Recognition Works*. [Online]  
Available at: <https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/speech-recognition.htm>  
[Accessed 13 November 2020].

H. Tankovska, 2020. *Number of digital voice assistants in use worldwide 2019-2024*. [Online]  
Available at: <https://www.statista.com/statistics/973815/worldwide-digital-voice-assistant-in-use/#:~:text=Number%20of%20digital%20voice%20assistants%20in%20use%20worldwide%202019%2D2024&text=In%202020%2C%20there%20will%20be,higher%20than%20the%20world%27s%20population>  
[Accessed 19 October 2020].

Harris, N., 2020. *What is Python? Executive Summary*. [Online]  
Available at: <https://www.python.org/doc/essays/blurb/>  
[Accessed 15 October 2020].

Heath, N., 2018. *Raspberry Pi and machine learning: How to get started*. [Online]  
Available at: <https://www.techrepublic.com/article/raspberry-pi-and-machine-learning-how-to-get-started/>  
[Accessed 17 October 2020].

Heath, N., 2019. *What is the Raspberry Pi 4? Everything you need to know about the tiny, low-cost computer*. [Online]  
Available at: <https://www.zdnet.com/article/what-is-the-raspberry-pi-4-everything-you-need-to-know-about-the-tiny-low-cost-computer/>  
[Accessed 15 October 2020].

Hendrickson, J., 2019. *How to Build Your Own Futuristic Smart Mirror*. [Online]  
Available at: <https://www.howtogeek.com/414647/how-to-build-a-smart-mirror/>  
[Accessed 13 November 2020].

Hobson, J., 2016. *SMART MIRROR NOTICES YOU AND TURNS ON*. [Online]  
Available at: <https://hackaday.com/2016/01/14/smart-mirror-notices-you-and-turns-on/>  
[Accessed 7 November 2020].

Horse, J., 2019. *DIY smart mirror*. [Online]  
Available at: <https://www.geeky-gadgets.com/diy-smart-mirror-29-05-2019/>  
[Accessed 07 November 2020].

Ito, K., 2020. *Tacotron*. [Online]  
Available at: <https://github.com/keithito/tacotron>  
[Accessed 09 November 2020].

Joshua Montgomery, Ryan Sipes, 2020. *Mycroft AI*. [Online]  
Available at: <https://mycroft.ai/>  
[Accessed 31 October 2020].

Jr., D. O., 2015. *Does JARVIS actually exist?*. [Online]  
Available at: <https://www.quora.com/Does-JARVIS-actually-exist>  
[Accessed 06 November 2020].

King, D., 2017. *High Quality Face Recognition with Deep Metric Learning*. [Online]  
Available at: <http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html>  
[Accessed 17 October 2020].

Kulhary, R., 2019. *OpenCV – Overview*. [Online]

Available at: <https://www.geeksforgeeks.org/opencv-overview/>

[Accessed 20 October 2020].

Kumar, A., 2020. *Top 10 Python Applications in the Real World You Need to Know*. [Online]

Available at: <https://www.edureka.co/blog/python-applications/>

[Accessed 20 October 2020].

Kumar, N., 2018. *Multiprocessing in Python*. [Online]

Available at: <https://www.geeksforgeeks.org/multiprocessing-python-set-1/>

[Accessed 26 October 2020].

LEONARD, M., 2018. *Why confidence matters in facial recognition systems*. [Online]

Available at: <https://gcn.com/articles/2018/08/06/trust-facial-recognition.aspx>

[Accessed 20 October 2020].

Levin, G., 2020. *Image Processing and Computer Vision*. [Online]

Available at: [https://openframeworks.cc/ofBook/chapters/image\\_processing\\_computer\\_vision.html](https://openframeworks.cc/ofBook/chapters/image_processing_computer_vision.html)

[Accessed 13 November 2020].

Locker, M., 2019. *Google has two cool, new Android apps to help people with hearing loss*. [Online]

Available at: <https://www.fastcompany.com/90302345/google-aims-to-help-people-with-hearing-loss-with-two-cool-inclusive-android-apps>

[Accessed 27 October 2020].

Ltd, T. C., 2020. *FREQUENTY ASKED QUESTIONS WHEN DESIGNING-IN A GNSS ANTENNA*. [Online]

Available at: <https://www.te.com/content/dam/te-com/documents/about-te/marketing/global/select-campaign/gnss-antennas-faqs.pdf>

[Accessed 17 March 2021].

MakerHawk, 2020. *MakerHawk Raspberry Pi Camera Night Vision Camera Module 5MP OV5647*. [Online]

Available at: <https://www.amazon.co.uk/MakerHawk-Raspberry-Camera-Vision-Adjustable/dp/B071718FDK>

[Accessed 01 October 2020].

MakerHawk, 2020. *MakerHawk Raspberry Pi Camera Night Vision Camera Module 5MP OV5647*. [Online]

Available at: <https://www.amazon.co.uk/MakerHawk-Raspberry-Camera-Vision-Adjustable/dp/B071718FDK>

[Accessed 10 October 2020].

Mathlab, 2020. *Machine Learning with MATLAB*. [Online]

Available at: [https://uk.mathworks.com/campaigns/offers/machine-learning-with-matlab.confirmation.html?elqsid=1603991809679&potential\\_use=Student](https://uk.mathworks.com/campaigns/offers/machine-learning-with-matlab.confirmation.html?elqsid=1603991809679&potential_use=Student)

[Accessed 29 October 2020].

Medmain, 2018. *My first steps into the world of A.I.*. [Online]

Available at: <https://medium.com/@Medmain/my-first-steps-into-the-world-of-ai-d7d591b2fe22>

[Accessed 03 November 2020].

Mirror, M., 2020. *Magic Mirror Team*. [Online]

Available at: <https://docs.magicmirror.builders/getting-started/installation.html#usage>

[Accessed 07 November 2020].

MIT, 2020. *Human User Experience, Amazon Echo*. [Online]

Available at: <http://web.mit.edu/2.744/studentSubmissions/humanUseAnalysis/nikimr97/UserExperience.html>

[Accessed 16 October 2020].

Mr. Mehmet Ucar, Dr. Sheng-Jen, 2018. *MAKER: Face Detection Library to Teach*. s.l., s.n., p. 5.

Mutchler, A., 2017. *Voice Assistant Timeline: A Short History of the Voice Revolution*. [Online]  
Available at: <https://voicebot.ai/2017/07/14/timeline-voice-assistants-short-history-voice-revolution/>  
[Accessed 27 October 2020].

Noronha, B., 2017. *Multithreading vs Multiprocessing in Python*. [Online]  
Available at: <https://dev.to/nbosco/multithreading-vs-multiprocessing-in-python--63j#:~:text=The%20Python%20threading%20module%20uses,the%20same%20unique%20memory%20heap.&text=The%20multiprocessing%20library%20uses%20separate,is%20much%20easier%20to%20use.>  
[Accessed 04 January 2021].

Note, T., 2020. *What is Speech to Text Software?*. [Online]  
Available at: <https://takenote.co/what-is-speech-to-text-software/>  
[Accessed 09 November 2020].

O'Boyle, B., 2020. *The 13 best personal assistant apps*. [Online]  
Available at: <https://www.pocket-lint.com/smart-home/news/amazon/138846-what-is-alexa-how-does-it-work-and-what-can-amazons-alexa-do>  
[Accessed 27 October 2020].

OpenCV, T., 2020. *About OpenCV*. [Online]  
Available at: <https://opencv.org/about/>  
[Accessed 17 October 2020].

Otachi, E., 2019. *Windows 10 Accessibility Features For Disabled People*. [Online]  
Available at: <https://helpdeskgeek.com/windows-10/windows-10-accessibility-features-for-disabled-people/>  
[Accessed 27 October 2020].

Otte, S., 2020. *How does Artificial Intelligence Work?*. [Online]  
Available at: <https://www.innoplexus.com/blog/how-artificial-intelligence-works/>  
[Accessed 28 October 2020].

Paul, D., 2020. *Jarvis AI*. [Online]  
Available at: [https://github.com/Dipeshpal/Jarvis\\_AI](https://github.com/Dipeshpal/Jarvis_AI)  
[Accessed 4 November 2020].

Pimoroni, 2017. *Pan Tilt Face Tracker*. [Online]  
Available at: <https://github.com/pimoroni/PanTiltFacetracker>  
[Accessed 20 October 2020].

Pimoroni, 2020. *Pan-Tilt-Hat - Full Kit*. [Online]  
Available at: <https://shop.pimoroni.com/products/pan-tilt-hat?variant=22408353287>  
[Accessed 15 October 2020].

PI, R., 2020. *DOCUMENTATION > USAGE*. [Online]  
Available at: <https://www.raspberrypi.org/documentation/usage/>  
[Accessed 16 October 2020].

pypi, 2014. *CMUCLMTK*. [Online]  
Available at: <https://pypi.org/project/cmuclmtk/>  
[Accessed 31 October 2020].

pypi, 2018. *Pocketsphinx Python*. [Online]  
Available at: <https://pypi.org/project/pocketsphinx/>  
[Accessed 31 October 2020].

pypi, 2020. *beautifulsoup4* 4.9.3. [Online]

Available at: <https://pypi.org/project/beautifulsoup4/>

[Accessed 6 November 2020].

pypi, 2020. *gTTS* 2.1.1. [Online]

Available at: <https://pypi.org/project/gTTS/>

[Accessed 6 November 2020].

pypi, 2020. *gTTS* 2.1.1. [Online]

Available at: <https://pypi.org/project/gTTS/>

[Accessed 6 November 2020].

pypi, 2020. *playsound* 1.2.2. [Online]

Available at: <https://pypi.org/project/playsound/>

[Accessed 6 November 2020].

pypi, 2020. *pytsx3* 2.90. [Online]

Available at: <https://pypi.org/project/pytsx3/>

[Accessed 6 November 2020].

pypi, 2020. *SpeechRecognition* 3.8.1. [Online]

Available at: <https://pypi.org/project/SpeechRecognition/>

[Accessed 6 November 2020].

pypi, 2020. *wikipedia* 1.4.0. [Online]

Available at: <https://pypi.org/project/wikipedia/>

[Accessed 6 November 2020].

Python, 2020. *Sunsetting Python* 2. [Online]

Available at: [https://www.python.org/doc/sunset-python-](https://www.python.org/doc/sunset-python-2/#:~:text=We%20have%20decided%20that%20January,as%20soon%20as%20you%20can.)

[2/#:~:text=We%20have%20decided%20that%20January,as%20soon%20as%20you%20can.](https://www.python.org/doc/sunset-python-2/#:~:text=We%20have%20decided%20that%20January,as%20soon%20as%20you%20can.)

[Accessed 31 October 2020].

Rabideau, C., 2020. *What to Know About Smart Mirrors*. [Online]

Available at: <https://www.familyhandyman.com/article/what-to-know-about-smart-mirrors/>

[Accessed 07 November 2020].

Ravenscraft, E., 2016. *This Chart Shows How Computer Literate Most People Are*. [Online]

Available at: <https://lifehacker.com/this-chart-shows-how-computer-literate-most-people-are-1789761598>

[Accessed 13 November 2020].

Rockbrock, A., 2019. *Pan/tilt face tracking with a Raspberry Pi and OpenCV*. [Online]

Available at: <https://www.pyimagesearch.com/2019/04/01/pan-tilt-face-tracking-with-a-raspberry-pi-and-opencv/>

[Accessed 21 October 2020].

Rockbrock, A., 2019. *Pan/tilt face tracking with a Raspberry Pi and OpenCV*. [Online]

Available at: <https://www.pyimagesearch.com/2019/04/01/pan-tilt-face-tracking-with-a-raspberry-pi-and-opencv/>

Rosebrock, A., 2018. *Raspberry Pi Face Recognition*. [Online]

Available at: <https://www.pyimagesearch.com/2018/06/25/raspberry-pi-face-recognition/>

[Accessed 17 October 2020].

Rovai, M., 2018. *Pan-Tilt Multi Servo Control*. [Online]

Available at: <https://www.hackster.io/mjrobot/pan-tilt-multi-servo-control-b67791>

[Accessed 15 October 2020].

S. Sharma, Karthikeyan Shanmugasundaram, Sathees Kumar Ramasamy, 2016. *FAREC — CNN based efficient face recognition technique using Dlib*. Ramanathapuram, IEEE.

Saba, E., 2018. *Amazon Echo Show's new 'Tap to Alexa' accessibility feature is handy enough for anyone to use*. [Online] Available at: <https://www.aftvnews.com/amazon-echo-shows-new-tap-to-alexa-accessibility-feature-is-handy-enough-for-anyone-to-use/> [Accessed 15 October 2020].

Sean McManus, M. C., 2020. *Top 10 Programming Languages Ported to the Raspberry Pi*. [Online] Available at: <https://www.dummies.com/computers/raspberry-pi/top-10-programming-languages-ported-to-the-raspberry-pi/> [Accessed 16 October 2020].

Serasinghe, S., 2020. *Setting up your Raspberry Pi 4 - Headless mode*. [Online] Available at: <https://sahansera.dev/setting-up-raspberry-pi-4-headless-mode/> [Accessed 20 October 2020].

Source, A. O., 2020. *The Top 12 Jarvis Open Source Projects*. [Online] Available at: <https://awesomeopensource.com/projects/jarvis> [Accessed 06 November 2020].

Springer, A., 2020. *Transformer TTS*. [Online] Available at: <https://github.com/as-ideas/TransformerTTS> [Accessed 09 November 2020].

Sunsky, 2020. *Yanmai SF-777 1.4m Computer Game Recording Condenser Microphone with Pop Filter & Tripod Stand*. [Online] Available at: <https://www.sunsky-online.com/product/default!view.do?subject.itemNo=MCP0082L> [Accessed 04 November 2020].

Szeliski, R., 2011. *Computer Vision Algorithms and Applications*. 1st ed. New-York: Springer London Dordrecht Heidelberg.

Taulli, T., 2019. *Artificial Intelligence Basics*. Monrovia: Apress.

Team, L., 2020. *lxml - XML and HTML with Python*. [Online] Available at: <https://lxml.de/> [Accessed 6 November 2020].

Team, O. S., 2020. *The Top 65 Tts Open Source Projects*. [Online] Available at: <https://awesomeopensource.com/projects/tts> [Accessed 09 November 2020].

Team, T. U., 2020. *Text-to-Speech Technology: What It Is and How It Works*. [Online] Available at: <https://www.understood.org/en/school-learning/assistive-technology/assistive-technologies-basics/text-to-speech-technology-what-it-is-and-how-it-works> [Accessed 6 November 2020].

Teeuw, M., 2020. *Magic Mirror*. [Online] Available at: <https://github.com/MichMich/MagicMirror> [Accessed 07 November 2020].

Thales, 2020. *Facial recognition: top 7 trends (tech, vendors, markets, use cases and latest news)*. [Online] Available at: <https://www.thalesgroup.com/en/markets/digital-identity-and-security/government/biometrics/facial-recognition> [Accessed 13 November 2020].

TheSchoolRun, 2020. *What is a phoneme?*. [Online] Available at: <https://www.theschoolrun.com/what-is-a-phoneme> [Accessed 09 November 2020].

- Tiltman, M., 2018. *Amazon's Tap to Alexa feature: How it works and how to turn it on*. [Online]  
Available at: <https://www.pocket-lint.com/smart-home/news/amazon/145176-amazon-s-tap-to-alexa-feature-how-it-works-and-how-to-turn-it-on>  
[Accessed 27 October 2020].
- TV, B., 2020. *Customise your Raspberry Pi magic mirror with modules*. [Online]  
Available at: <https://www.blogdot.tv/customise-your-raspberry-pi-magic-mirror-with-modules/>  
[Accessed 13 November 2020].
- Verizon, I., 2020. *Understanding Virtual Assistants*. [Online]  
Available at: <https://fios.verizon.com/beacon/virtual-assistants/>  
[Accessed 17 October 2020].
- Verma, A., 2016. *What is the advantage of Python code in Raspberry Pi?*. [Online]  
Available at: <https://www.quora.com/What-is-the-advantage-of-Python-code-in-Raspberry-Pi>  
[Accessed 16 October 2020].
- Villán, A. F., 2019. *Mastering OpenCV 4 with Python*. [Online]  
Available at: <https://subscription.packtpub.com/book/data/9781789344912/1/ch01lvl1sec10/technical-requirements>  
[Accessed 15 October 2020].
- Warwick, K., 2012. *Artificial Intelligence the basics*. Second ed. New York: Routledge.
- Waveshare, 2018. *GSM/GPRS/GNSS HAT User Manual*. [Online]  
Available at: [https://www.waveshare.com/w/upload/4/4a/GSM\\_GPRS\\_GNSS\\_HAT\\_User\\_Manual\\_EN.pdf](https://www.waveshare.com/w/upload/4/4a/GSM_GPRS_GNSS_HAT_User_Manual_EN.pdf)  
[Accessed 16 March 2021].
- Wellshow, 2021. *GSM Antennas*. [Online]  
Available at: <http://www.wellshow.com/products/antennas/gsm-antennas/>  
[Accessed 17 March 2021].
- WHO, 2020. *Deafness and hearing loss*. [Online]  
Available at: <https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss#:~:text=Over%205%25%20of%20the%20world%27s,will%20have%20disabling%20hearing%20loss>  
[Accessed 14 October 2020].
- Wikipedia, 2020. *Amazon Alexa*. [Online]  
Available at: [https://en.wikipedia.org/wiki/Amazon\\_Alexa](https://en.wikipedia.org/wiki/Amazon_Alexa)  
[Accessed 13 October 2020].
- Wikipedia, 2020. *Computer literacy*. [Online]  
Available at: [https://en.wikipedia.org/wiki/Computer\\_literacy](https://en.wikipedia.org/wiki/Computer_literacy)  
[Accessed 13 November 2020].
- Wikipedia, 2020. *Hearing Loss*. [Online]  
Available at: [https://en.wikipedia.org/wiki/Hearing\\_loss](https://en.wikipedia.org/wiki/Hearing_loss)  
[Accessed 19 October 2020].
- Wikipedia, 2020. *J.A.R.V.I.S.*. [Online]  
Available at: <https://en.wikipedia.org/wiki/J.A.R.V.I.S.>  
[Accessed 06 November 2020].
- Wikipedia, 2020. *Local Binary Patterns*. [Online]  
Available at: [https://en.wikipedia.org/wiki/Local\\_binary\\_patterns](https://en.wikipedia.org/wiki/Local_binary_patterns)  
[Accessed 24 October 2020].

Wikipedia, 2020. *Multiprocessing*. [Online]  
Available at: <https://en.wikipedia.org/wiki/Multiprocessing>  
[Accessed 26 October 2020].

Wikipedia, 2020. *PID controller*. [Online]  
Available at: [https://en.wikipedia.org/wiki/PID\\_controller](https://en.wikipedia.org/wiki/PID_controller)  
[Accessed 25 October 2020].

Wikipedia, 2020. *Segmentation fault*. [Online]  
Available at: [https://en.wikipedia.org/wiki/Segmentation\\_fault](https://en.wikipedia.org/wiki/Segmentation_fault)  
[Accessed 06 November 2020].

Wikipedia, 2020. *Turing test*. [Online]  
Available at: [https://en.wikipedia.org/wiki/Turing\\_test](https://en.wikipedia.org/wiki/Turing_test)  
[Accessed 28 October 2020].