we

Misr University for Science & Technology

Faculty of Engineering

Electronics and Communication Department

Smart Glasses For Blind And Deaf People Part A

Supervisor

Dr. Mamdouh Goda

Group Members:

Omar Tarek Mohamed	75431
Ibrahim Ahmed Hekal	75194
Esraa Hassan Ali	75285
Wesam Ahmed Mohamed	80823
Omar Mohamed Youssef	77199

Acknowledgment

First, we would like to express our heartfelt appreciation to our advisor, Dr. Mamdouh Gouda, for his unwavering support of our study and research, as well as his patience, encouragement, inspiration, and vast knowledge. His advice was invaluable throughout the research and writing of this thesis. We could not have asked for a better advisor and mentor for our project. Finally, we would like to show our special thanks to the department's head, Dr. Salama Abo El Soud.

Abstract

Being with a disability means unable to function properly and normally and that is the case for almost 15% of the world's population that is about one person in every five people lives with some sort of disability. That is why we are encouraged to help people with disability so that they are able to function more efficiently and lead more productive life.

Globally, there are 36 million blind people. The quality of life for visually impaired people is hampered by the resultant lack of independence. According to the World Health Organization, out of 9.2 billion population around 287 million people are believed to be having imperceptive of vision. It is noticed that they are still facing difficulty to perform their daily routine and it is significant to take required action with the help of arising technology to help these kinds of people to lead a normal lifestyle like others.So, we introduced a smart glass for the blind people which scan any text image and convert it into audio text, so the person will listen to the audio through a headphone that's connected to the glasses. This can help the person with visual disability to read any printed note and identify in the form vocal note.

Furthermore, according to WHO over 5% of the world's population need rehabilitation to address their hearing disability that means that over 360 million people worldwide (one in every twenty people have hearing disability). As a result, creating a way to help people with hearing disabilities to lead more productive and active live has been an aspiration that we hope to achieve. subsequently, we are looking to introduce a smart device that is able to transform voice into text for people with hearing disability to be able to communicate with others in there day to day life.

The glasses used many technologies to perform its tasks which are OCR, (PTTS). Detecting the text in the image was done using the OpenCV and Optical Character Recognition technology (OCR) with Tesseract an. In order to convert the text into speech, it used Text to Speech technology (PTTS). In order to convert the Speech to text, it used Speech to text (google API).

All the computing and processing operations were done using the Raspberry Pi 4 B+ and Raspberry pi camera.

Table of Contents

Chapter 3 Embedded processing

List of Figures

List of Tables

List of Abbreviations

Rasp: Raspberry Pi

OS: Operating System

GPIO: General-purpose input/output

CPU: central processing unit

GPU: Graphics Processing Unit

RAM: Random Access Memory

HDMI: High-Definition Multimedia Interface

CSI: Camera Serial Interface

DSI: Display Serial Interface

Open CV : Open-Source Computer Vision

RGB : Red, green, and blue

HSC : High significant channel

LSC : Low significant channel

OCR : Optical character recognition

LSTM : Long short-term memory

RNN : recurrent neural network

TTS : Text to speech conversion

ML : Machine learning

API : Application program interference

GTTS : Google Text-to-Speech

MSAPI : Microsoft speech Application program interference

Chapter 1
Introduction

Chapter 1 Introduction

Many people in our life are suffering with different diseases or disabilities. Many of them can't understand what happens without the help of sign language interpreters. Their life always depends upon their sign language interpreters, lipreading can only be used to understand about 30% of the words being spoken and can be quite difficult for them alone. The increasing number of people with disabilities in the world attracts the concern of researchers to invent various technologies, hoping that these technologies can assist the disabled people in carrying out their tasks in everyday life like normal people.

The main goal of "Smart Glasses" is to help blind people and people who have vision difficulties by introducing a new technology that makes them able to read the typed text. These glasses are provided with technology to scan any written text and convert it into audio text.

The goal of "Smart Glasses" is helping those people in different life aspects. For example, these glasses effectively helpful in the education field.

Blind people and people with vision difficulties can be able to read, study and learn everything from any printed text images. "Smart Glasses" encourage blind people or people with vision difficulties to learn and succeed in many different fields. Most Schools will be able to accept people with vision difficulties instead of open

special schools.

Chapter 2
Implementation Requirements

2.1 Introduction

This project's implementation plan primarily consists of two major fields. This chapter discusses the requirements for each part in the exact order of design and implementation.

Since this is a smart glasses system, there must be a group of interconnected devices managed by software programs to achieve the desired results autonomously. Raspberry Pi is a small single board computer. By connecting peripherals like Keyboard, mouse, display to the Raspberry Pi, it will act as a mini personal computer.

Raspberry Pi is popularly used for real time Image/Video Processing, IoT based applications and Robotics applications. Raspberry Pi is slower than laptop or desktop but is still a computer which can provide all the expected features or abilities, at a low power consumption.

Raspberry Pi Foundation officially provides Debian based Raspbian OS. Also, they provide OS for Raspberry Pi. We can install several Third-Party versions of OS like Ubuntu, RISC OS, Windows 10 IOT Core, etc.

Raspbian OS is official Operating System available for free to use. This OS is efficiently optimized to use with Raspberry Pi. Raspbian have GUI which includes tools for Browsing, Python programming, office, games, etc. We should use SD card (32 GB) to store the OS (operating System).

Raspberry Pi is more than computer as it provides access to the on-chip hardware i.e., GPIOs for developing an application. By accessing GPIO, we can connect devices like LED, motors, sensors, etc. and can control them too.

2.2 Hardware requirements

To design a project that meets the required standards for a sufficient Embedded system, some requirements such as cost, timesaving, security, mobility, and space must be taken into consideration. Therefore, we narrowed down our choice of hardware devices into the following:

2.2.1 Raspberry pi

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote teaching of basic computer science in schools and in developing countries.

• Budget-Friendly

• Space-Saver

• Easy usage/maintenance

• Compatible with several devices

There are different versions of Raspberry pi available as listed below:

	Raspberry Pi Zero	Raspberry Pi 3 B+	Raspberry Pi 4 B
			
SOC Type	Broadcom BCM2835	Broadcom BCM2837B0	Broadcom BCM2711
Core Type	ARM1176JZF-S	Cortex-A53 64-bit	Cortex-A72 (ARM v8) 64-bit
No. Of Cores	1	4	4
GPU	Video Core IV	Video Core IV	Video Core VI
CPU Clock	1 GHz	1.4 GHz	1.5 GHz
RAM	512 MB	1 GB DDR2	1 GB , 2 GB, 4 GB, 8GB LPDDR4
USB	**Yes** 1x micro OTG	**Yes** 4x USB2.0	**Yes** 2x USB3.0 + 2x USB2.0 + USB-C OTG
Ethernet	**No**	**Yes** Gigabit – Over USB 2.0	**Yes** Gigabit
HDMI port	**Yes** mini HDMI	**Yes** full HDMI	**Yes** 2x micro HDMI
Analog Video Out	**-** via unpopulated pin	**Yes** shared with audio jack	**Yes** shared with audio jack
Analog Audio Out	**-** HDMI audio	**Yes** 3.5mm jack	**Yes** 3.5mm jack
LCD Panel	**No**	**Yes**	**Yes**
Wi-Fi	**Yes** 802.11n	**Yes** 2.4GHz and 5GHz 802.11 b/g/n/ac	**Yes** 2.4GHz and 5GHz 802.11 b/g/n/ac
Bluetooth®	**Yes** 4.1 BLE	**Yes** 4.2, BLE	**Yes** 5.0
Power ratings	180 mA	1.13 A @5V	1.25 A @5V
Power sources	microUSB or GPIO	microUSB, GPIO	USB-C
Power Over Ethernet	**No**	**-** with PoE Hat	**-** with PoE Hat

2.2.2 Raspberry Pi 4model B technical specifications

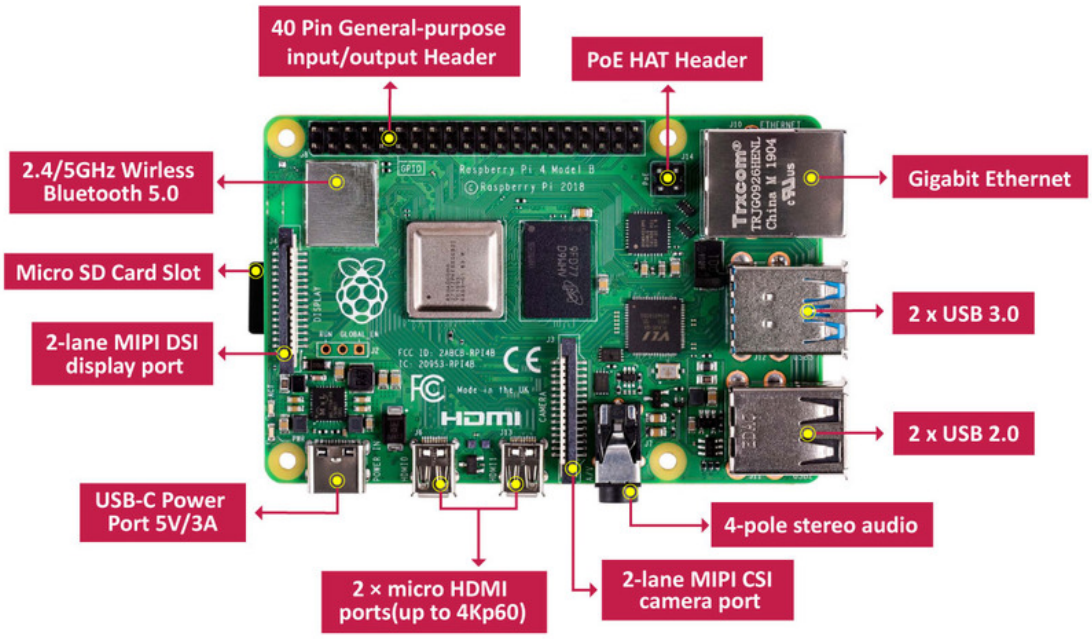


Figure 2.1 Raspberry Technical Specifications

In terms of end use, it is important to note that the performance of the new Raspberry Pi 4 Model B is equivalent to that of an entry-level x86 computer. Among the main features of this latest Raspberry Pi computer, we can note:

• A high-performance 64-bit quad-core processor

• Dual display support with resolutions up to 4K via a pair of micro-HDMI ports

• Hardware video decoding up to 4Kp60

• 4 GB of RAM

• A connection to the dual-band wireless local area network 2.4/5.0 GHz

• Bluetooth 5.0 / Gigabit Ethernet / USB 3.0 / PoE features (via a separate HAT PoE add-on module)

• SoC Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit at 1.5GHz

• SDRAM 4 GB LPDDR4-2400

• Wireless LAN 2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac, Bluetooth 5.0, BLE

• True Gigabit Ethernet

• 2 USB 3.0 ports, 2 USB 2.0 ports

• Fully backward compatible 40-pin GPIO connector

• 2 HDMI micro ports supporting video resolution up to 4K 60Hz

• 2-way MIPI DSI DSI/CSI ports for camera and display

• Stereo audio output and composite video port, 4-pole

• Slot for Micro SD card, for operating system and data storage

• Requires 5.1V, 3A power supply via USB-C or GPIO

• PoE (Power over Ethernet) enabled (requires PoE HAT)

2.2.2.1 Raspberry Pi 4 Pins (GPIO)

Raspberry Pi 4 pins overview Here's a complete overview with all the GPIOs and their primary function.

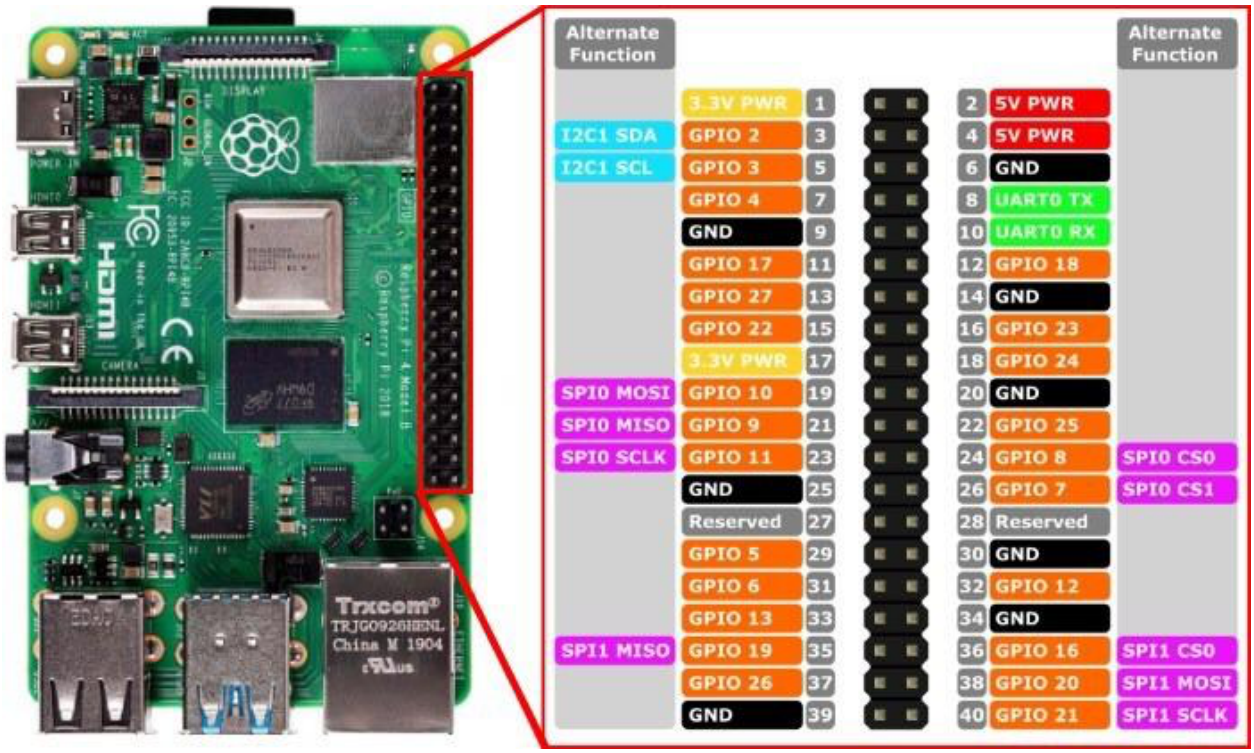


Figure 2.2 Raspberry Pi 4 Pins (GPIO)

Now, let's break down each pin or group of pins, and see what they can do.

A word of caution

Before you plug anything to a Raspberry Pi 4 pin, you must know that you can easily damage the board if you do something wrong. There is no real hardware safety when it comes to the Raspberry Pi hardware pins.

If you connect a ground (GND) pin to a 3.3V pin directly, well… You might destroy your Raspberry Pi board the second those pins are connected together.

So, be careful when you plug something or when you create a test circuit. If you have any doubt, double, triple check, and ask someone for help before you burn

your board. But if you follow some basic rules and common sense, you'll have nothing to worry about!

Ground pins

The ground is very useful for making a common reference between all components in your circuit. Always remember to connect all components to the ground.

If you connect 2 circuits together, add a wire between both grounds to make it common. If you add a new sensor/actuator to an existing circuit, connect the ground of the component to the ground of the circuit.

That's very important. Without that, you may burn some parts of the circuit, you may have components that do not function correctly, give wrong values, etc.

8 out of the 40 GPIOs are connected to the ground. You can find them with the 3 letters GND.

One additional warning: don't ever connect the ground directly to a power supply pin (3.3V or 5V)! This creates a short circuit and can definitively burn your Raspberry Pi 4 board.

Power pins

You can find 2 pins bringing 3.3V and 2 pins bringing 5V.

Those pins can be used to power components such as sensors or small actuators. Note that they are certainly not powerful enough to actuate motors such as servo or stepper motors. For that you'll need an external power source.

The power pins are used as a source to power external components, not to power the Raspberry Pi itself from an external source. (Well, there is a way to power the Raspberry Pi from the GPIO header, but you have a high probability of burning it, so just use the micro-USB port)

And just another word of caution: as previously said in the Ground pins section, don't ever connect one of the power pins directly to one of the GND of the Raspberry Pi 4!

Reserved pins

The pins 27 and 28 are reserved pins. They are usually used for I2C communication with an EEPROM.

Figure 2.3 The pins 27 and 28 of GPIO in Raspberry pi 4

If you just begin with Raspberry Pi 4 pins, just don't connect anything to those pins. There are many other available pins for you to use.

Well, that's 14 slots already taken for GND, power supply and reserved pins. Now let's see how the other 26 GPIOs are used for communication.

Raspberry Pi 4 GPIOs ; GPIO means General Purpose Input/Output. Basically, that's one pin you can use to write data to external components (output), or read data from external components (input). If you embed your Raspberry Pi board with some hardware components, the GPIO header will become quite useful.

GPIOs will allow you to read some basic sensors (ex: infrared), control some actuators (those which are working with a ON/OFF mode), and communicate with other hardware boards, such as Raspberry Pi, Arduino, Beagle bone, Jetson Nano, etc. GPIOs are digital pins , The Raspberry Pi 4 GPIOs are quite like what we call "digital pins" on an Arduino board.

First you need to choose whether you want to use them as input or output. If you configure a GPIO as input, you'll be able to read a value from it: HIGH or LOW (1 or 0). And if you configure a GPIO as output, you'll be able to write a value to it, also HIGH or LOW.

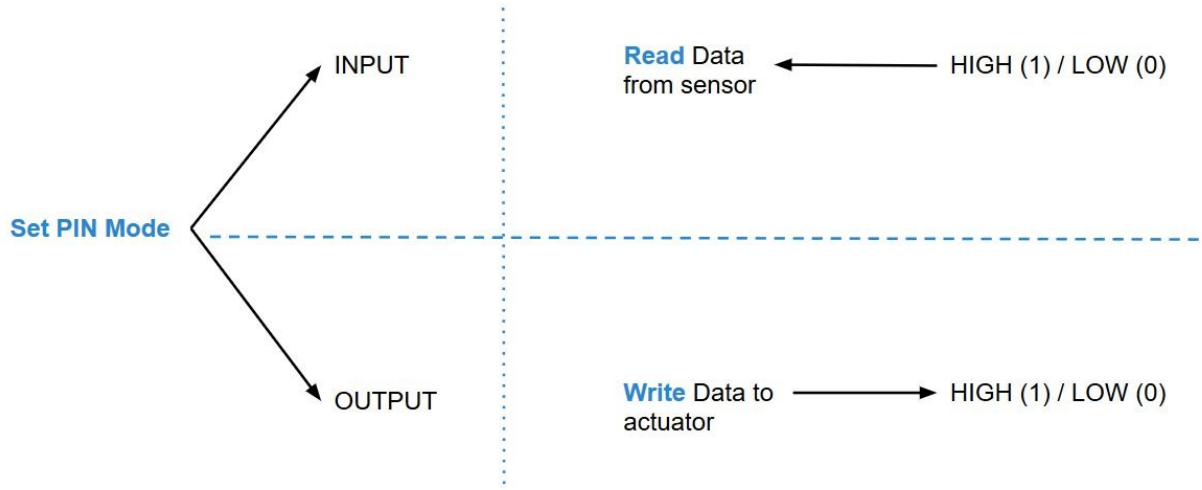


Figure 2.4 GPIOs are digital pins

A digital pin has only two states. LOW usually means 0V, and HIGH means 3.3V (with some tolerances). That's very simple, it's like a switch that you turn on and off.

Note: when you set the mode for a GPIO as output, after booting the Pi, you can expect a different default state for different GPIOs.

GPIOs voltage

All GPIOs work at 3.3V. It's important for you to know that in case you need to plug in a component with a different voltage.Sometimes, you'll find sensors that are powered with 5V, but all the communication pins are running with 3.3V. In this case, no problem: you can use the 5V power pin from the Raspberry Pi to power the component, and then use any 3.3V GPIO for the communication. If you don't mix the 5V signal with the 3.3V signals, everything should be alright.

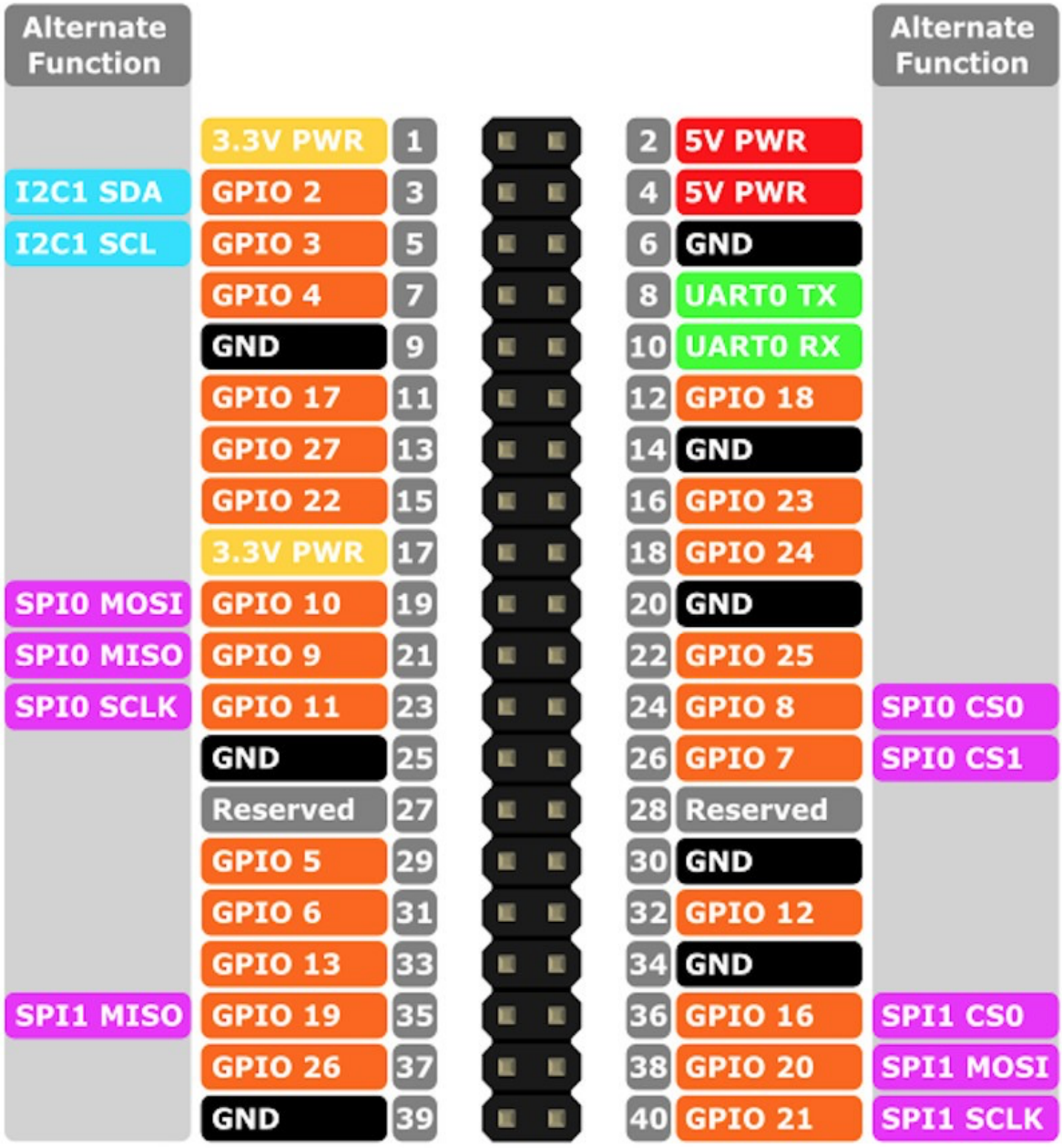


Figure 2.5 GPIO header

To Use GPIO

First you need to know its number . As you can see, the pin numbers and GPIO numbers are different. Pin numbers are in grey, and GPIO numbers in orange. Depending on the library you use to manipulate GPIOs, you'll either must use the number of the pin or the GPIO number. For example, pin 29 corresponds to GPIO 5. Any time you have a doubt, just check the pinout again and you'll know!

So, to use any of those GPIO, first you need to configure it as input or output, and after that you can write to it, or read from it.

Now, you might wonder: how can you configure and use the GPIOs from your code? Do you need to dive into complex hardware stuff to do that?

There are at least 2 libraries that will allow you to easily use those pins. For Python, you can use RPi.GPIO, and for Cpp you can use WiringPi.

Those libraries were developed so you can use the Raspberry Pi pins just like you would use Arduino pins, which means that all the complex stuff is hidden and you can use them with just a few lines of code.

For example, to set GPIO 17 (pin 11) as output/high:

```
// With WiringPi (Cpp)
pinMode (17, OUTPUT);
digitalWrite(17, HIGH);
```

```
# with RPi.GPIO (Python)
GPIO.setmode(GPIO.BCM) # Choose BCM to use GPIO numbers instead of pin numbers
GPIO.setup(17, GPIO.OUT)
GPIO.output(17, GPIO.HIGH)
```

Figure 2.6 GPIO 17 (pin 11) as output/high

To make 2 boards communicating with each other, it’s quite simple: you’ll configure a GPIO as an input on one side, and as an output on the other side. You can then use more GPIOs to transfer more pieces of information .

Communication protocols through Raspberry Pi 4 pins. You can use some hardware communication protocols directly with the Raspberry Pi 4 GPIOs. Those communication protocols are in fact the same ones that you can natively use on many Arduino boards. With those protocols you’ll be able to transfer far more information than with just a bunch of GPIOs configured as digital pins.On the Raspberry Pi 4 pinout schematics, you can see a column for alternate functions. Well, the communication protocols are all there!

In fact, saying that a GPIO is a digital pin is an overly exaggerated simplification. It’s much more than that. For each GPIO you have at least one alternate function, and sometimes many more. But let’s keep things simple here. You don’t need to know all the alternate functions to get started and develop cool applications. If you’re interested though, check out page 102 of the bmc2835 datasheet (this is the datasheet for the whole GPIO header), where you’ll see a complete table with all alternate functions for all GPIOs.

UART

UART is multi master communication protocol. This protocol is quite easy to use and very convenient for communicating between several boards: Raspberry Pi to Raspberry Pi, or Raspberry Pi to Arduino, etc.

Figure 2.7 UART pins

For using UART you need 3 pins:

1. GND that you'll connect to the global GND of your circuit.

2. RX for Reception. You'll connect this pin to the TX pin of the other component.

3. TX for Transmission. You'll connect this pin to the RX of the other component.

If the component you're communicating with is not already powered, you'll also have to use a power pin (3.3V or 5V) to power on that component.

By using a UART to USB converter, you can communicate between your laptop and Raspberry Pi with UART. Now, to use UART in your code, you can use the Serial library in Python, and WiringPi in Cpp. If you're interested in communicating between a Raspberry Pi board and an Arduino board via Serial, check out this Raspberry Pi Arduino Serial tutorial.

I2C

I2C is a master-slave bus protocol (well it can have multiple masters but you'll mostly use it with one master and multiple slaves). The most common use of I2C is to read data from sensors and actuate some components.

The master is the Raspberry Pi, and the slaves are all connected to the same bus. Each slave has a unique ID, so the Raspberry Pi knows which component it should talk to.

Figure 2.8 I2C

For using I2C you'll need 3 pins:

1. GND: I guess you start to get used to that!

2. SCL: clock of the I2C. Connect all the slaves SCL to the SCL bus.

3. SDA: exchanged data. Connect all the slaves SDA to the SDA bus.

And as most of the time you'll need to power on the component, you'll also need a power pin (3.3V or 5V), linked to the Vcc pin of the component. Make sure you know which voltage is accepted by the component before you plug anything. But don't worry too much though: usually, hobby components will accept 3.3V and/or 5V.

Note that the SDA and SCL pins on the Raspberry Pi are alternate functions for GPIO 2 and 3. When you use a library (Python, Cpp, etc) for I2C, those two GPIOs will be configured so they can use their alternate function.

Some of the best and easy-to-use libraries for I2C are SMBus for Python and WiringPi for Cpp.

SPI

SPI is yet another hardware communication protocol. It is a master-slave bus protocol. It requires more wires than I2C, but can be configured to run faster.

So, when to use I2C vs SPI on your Raspberry Pi 4? Well, the answer is quite simple. Sometimes you'll find a sensor that is only I2C or SPI compatible. And sometimes, you'll just want to have a balance between those protocols, so for example you'll choose to use I2C if you already have many components using the SPI. As you progress you'll start to know the differences better, and be able to make a better choice between those two protocols. But for now, let's keep things simple.

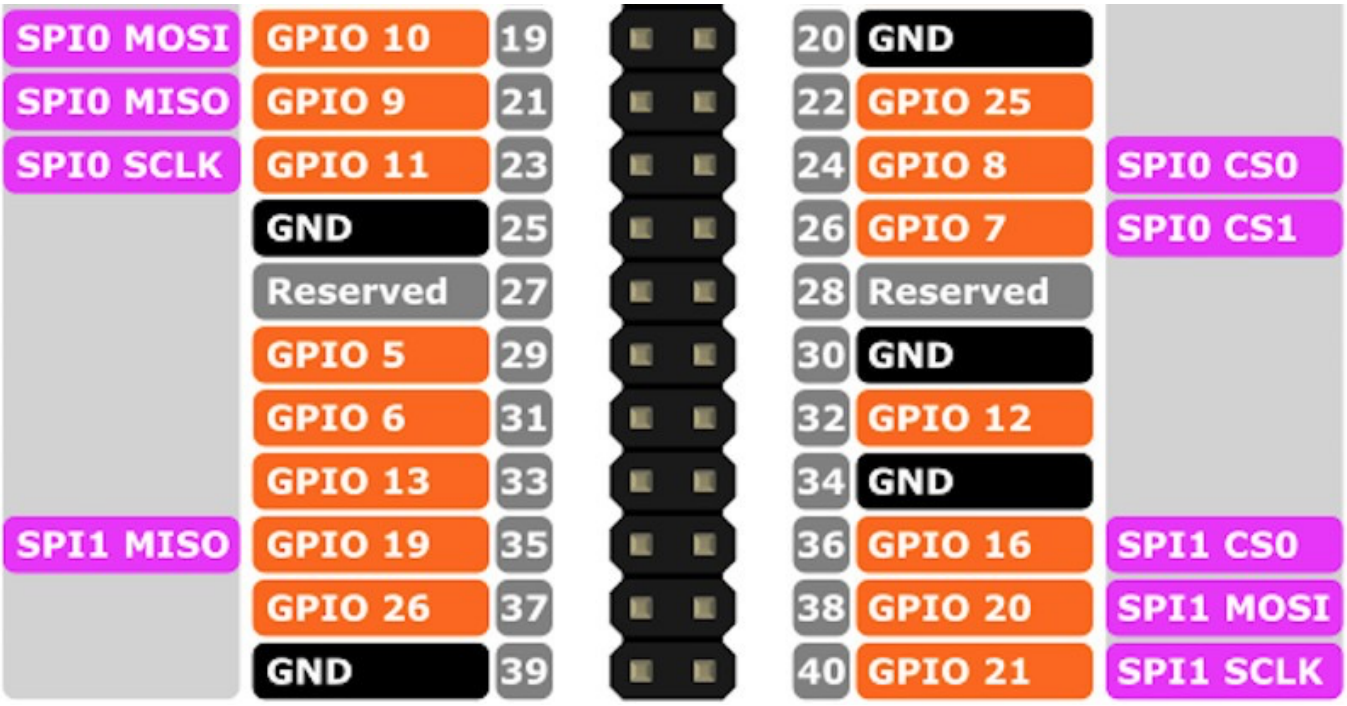


Figure 2.9 SPI

As you can see, you get 2 SPIs by default: SPI0 and SPI1. It means you can use the Raspberry Pi as a SPI master on two different SPI buses at the same time. And, as for I2C, SPI uses the alternate functions of GPIOs.

For using SPI you'll need 5 pins:

1. GND: what a surprise! Make sure you connect all GND from all your slave components and the Raspberry Pi together.

2. SCLK: clock of the SPI. Connect all SCLK pins together.

3. MOSI: means Master Out Slave In. This is the pin to send data from the master to a slave.

4. MISO: means Master In Slave Out. This is the pin to receive data from a slave to the master.

5. CS: means Chip Select. Pay attention here: you'll need one CS per slave on your circuit. By default you have two CS pins (CS0 – GPIO 8 and CS1 – GPIO 7). You can configure more CS pins from the other available GPIOs.

In your code, you can use the spidev library for Python, and WiringPi for Cpp.

2.2.2.2 Raspberry pi operating system

Raspberry Pi OS (formerly Raspbian) is a Debian-based operating system for Raspberry Pi. Since 2015, it has been officially provided by the Raspberry Pi Foundation as the primary operating system for the Raspberry Pi family of compact single-board computers.The first version of Raspbian was created by Mike Thompson and Peter Green as an independent project. The initial released build was completed on July 15, 2012

Raspberry Pi OS is highly optimized for the Raspberry Pi line of compact single-board computers with ARM CPUs. It runs on every Raspberry Pi except the Pico microcontroller. Raspberry Pi OS uses a modified LXDE as its desktop environment with the Openbox stacking window manager, along with a unique theme.

The distribution is shipped with a copy of the algebra program Wolfram Mathematica and a version of Minecraft called Minecraft: Pi Edition (note that Minecraft: Pi Edition is no longer installed as of the Debian bullseye update) as well as a lightweight version of the Chromium web browser.

User interface

Raspberry Pi OS's desktop environment, PIXEL, looks similar to many common desktops, such as macOS, Microsoft Windows, and is based on LXDE.The menu bar is positioned at the top and contains an application menu and shortcuts to Terminal, Chromium, and File Manager. On the right is a Bluetooth menu, a Wi-Fi menu, volume control, and a digital clock.

Package management

Packages can be installed via APT, the Recommended Software app, and by using the Add/Remove Software tool, a GUI wrapper for APT.

Components

PCManFM is a file browser allowing quick access to all areas of the computer, and was redesigned in the first Raspberry Pi OS Buster release (2019-06-20).

Raspberry Pi OS originally used Epiphany as the web browser, but switched to Chromium with the launch of its redesigned desktop.

Raspberry Pi OS comes with many beginner IDEs, such as Thonny Python IDE, Mu Editor, and Greenfoot. It also ships with educational software like Scratch and Bookshelf.

Version size

The Raspberry Pi documentation recommends at least a 4GB microSD card for Raspberry Pi OS Lite, an 8GB microSD card for Raspberry Pi OS, and a 16GB microSD card for Raspberry Pi OS Full.The image files themselves are 1,875MB, 3,980MB, and 8,603MB respectively.

Raspbian

Raspbian is a free operating system based on Debian optimized for the Raspberry Pi hardware. An operating system is the set of basic programs and utilities that make your Raspberry Pi run. However, Raspbian provides more than a pure OS: it comes with over 35,000 packages, pre-compiled software bundled in a nice format for easy installation on your Raspberry Pi.

The initial build of over 35,000 Raspbian packages, optimized for best performance on the Raspberry Pi, was completed in June of 2012. However, Raspbian is still under active development with an emphasis on improving the stability and performance of as many Debian packages as possible.

Raspbian is a community funded and supported free software effort. Although Raspbian is free software, the development costs associated with it are not free. As a user of Raspbian, you can support ongoing development of Raspbian by making a donation today.

2.2.2.3 Raspbian Setup And Installation

Step 1. Download Raspbian Image

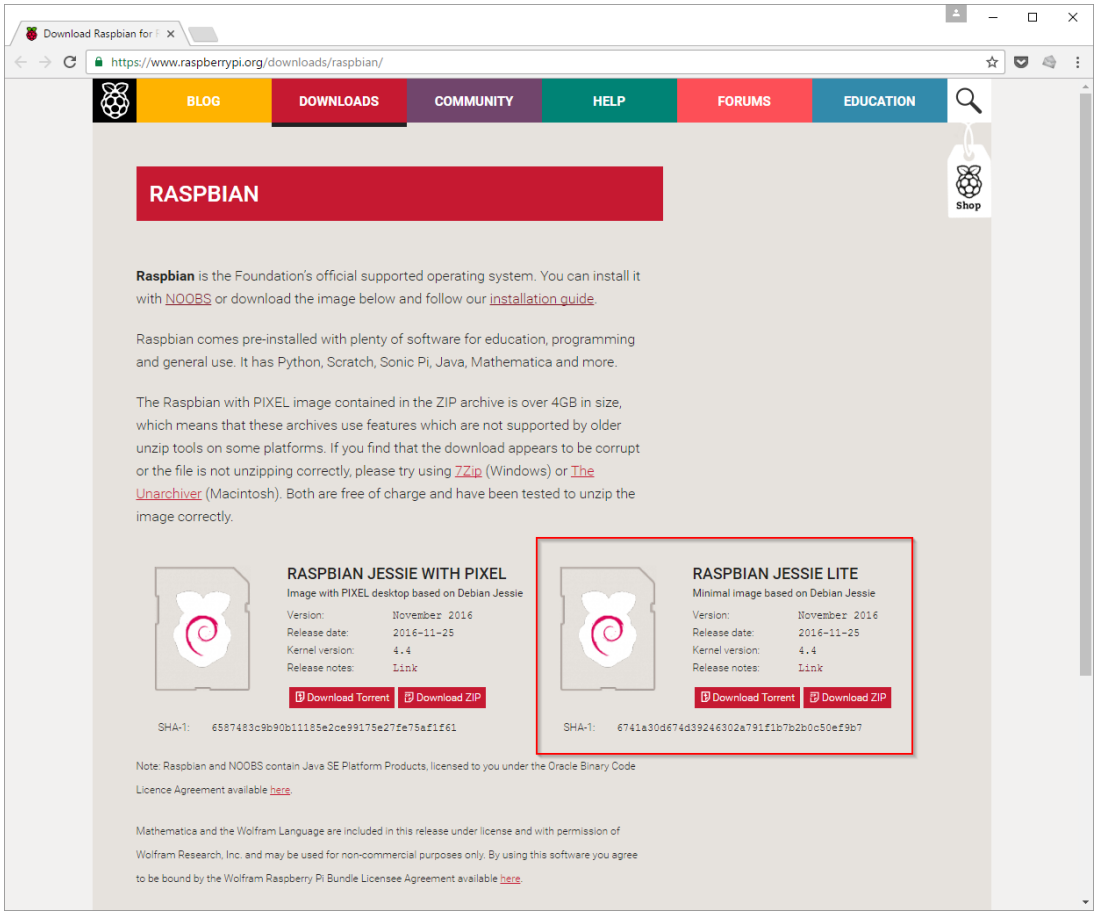


Figure 2.10 Step 1 Of Raspbian Setup

Step 2. Write Image to SD Card

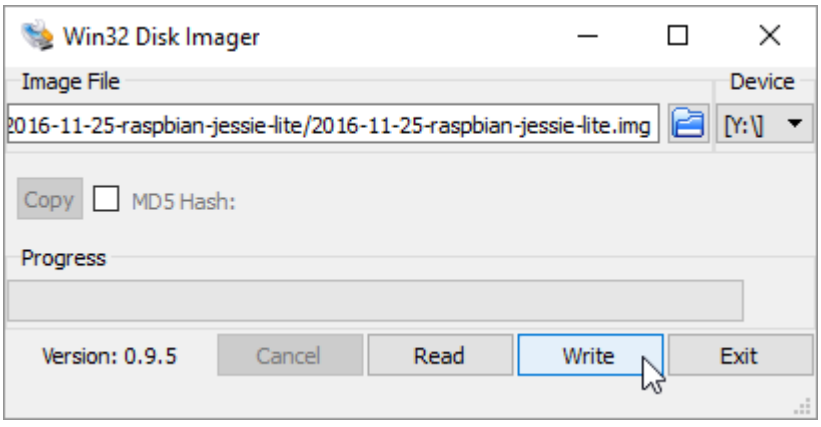


Figure 2.11 Step 2 Of Raspbian Setup

Step 3. Add “SSH” File to the SD Card Root

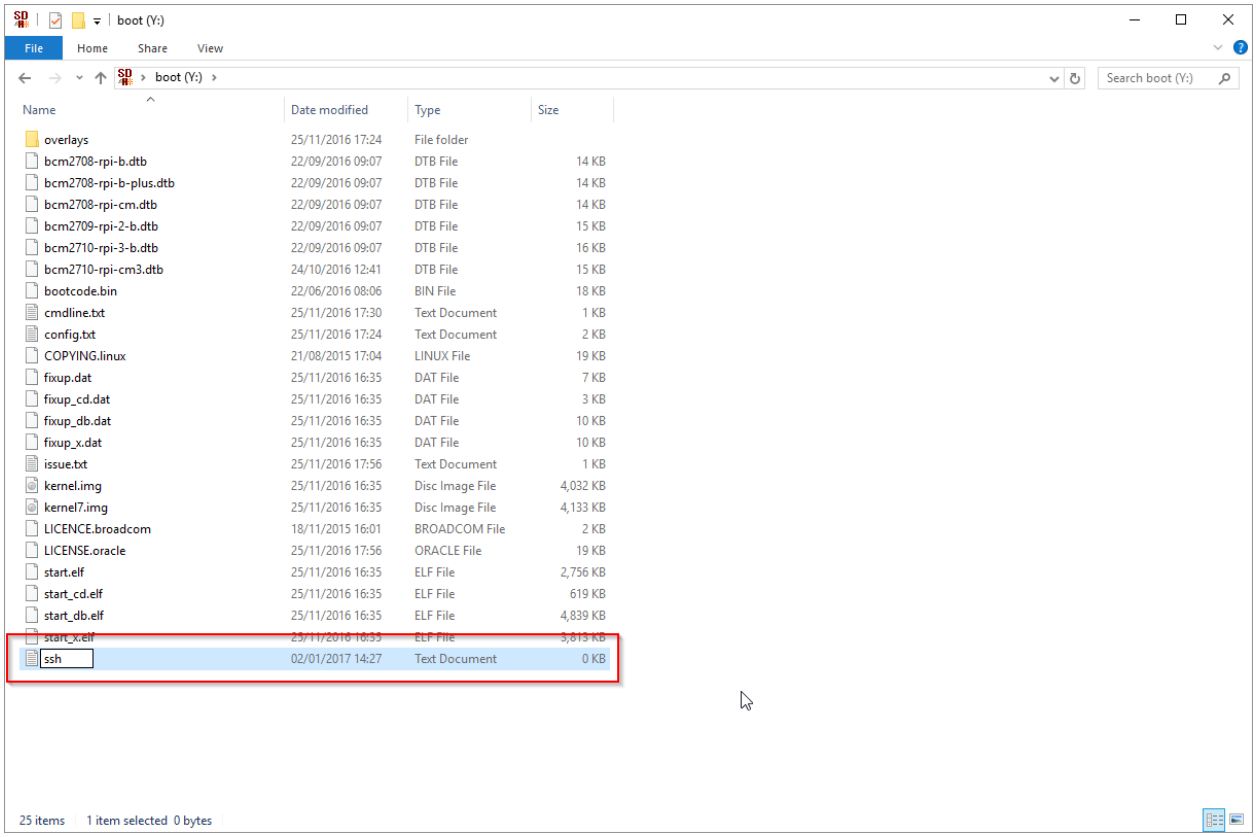


Figure 2.12 Step 3 Of Raspbian Setup

Step 4. Boot your Pi

Pop your prepared SD card, power and a network cable into the Pi.

Figure 2.13 Step 4 Of Raspbian Setup

Step 5. Find your Pi's IP Address

To configure your Pi, you need the IP address. You can find this in your Router's DHCP lease allocation table:

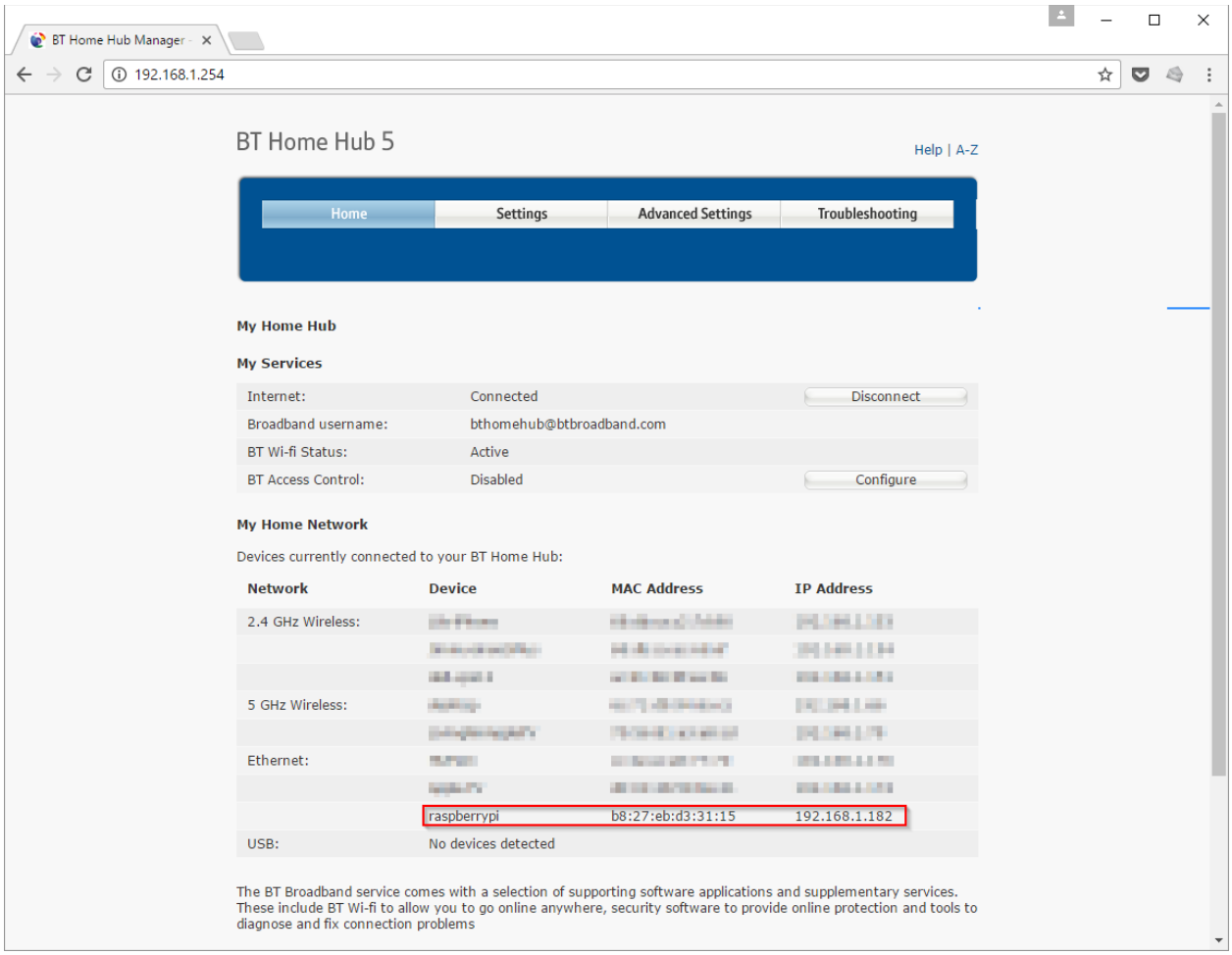


Figure 2.14 Step 5 Of Raspbian Setup

Step 6. SSH into your Pi

Use your favorite SSH client to access the Pi. The default credentials are:

```
username: pi
password: raspberry
```

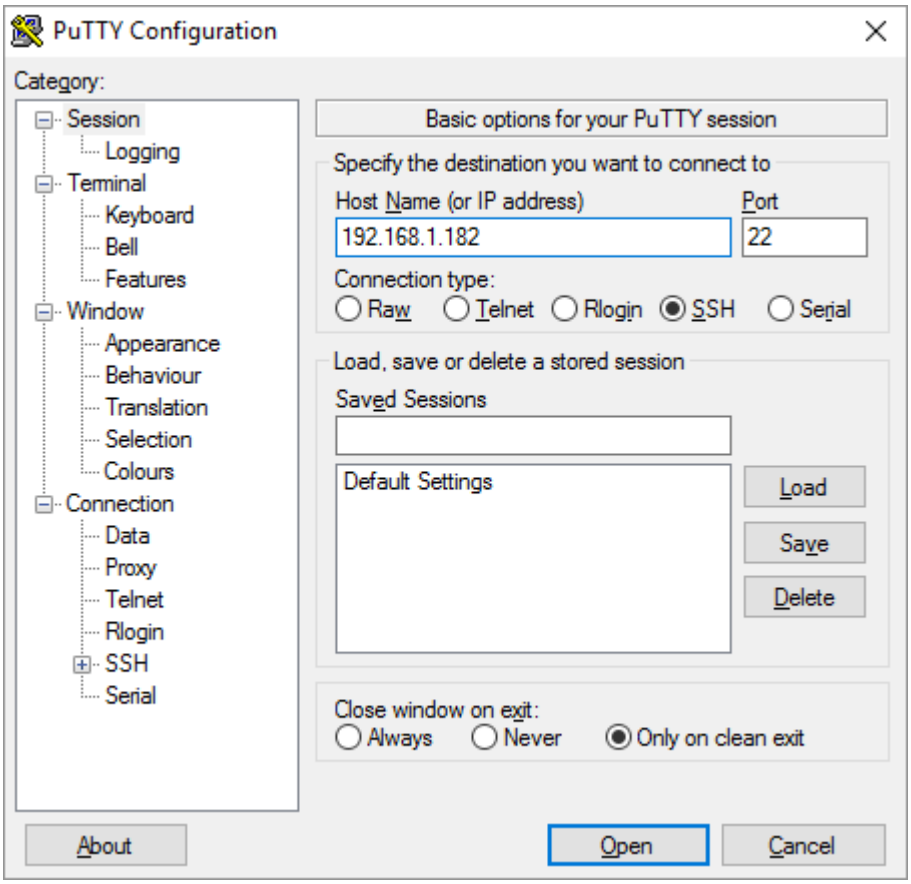


Figure 2.15 Step 6 Of Raspbian Setup

Step 7. Configure your Pi

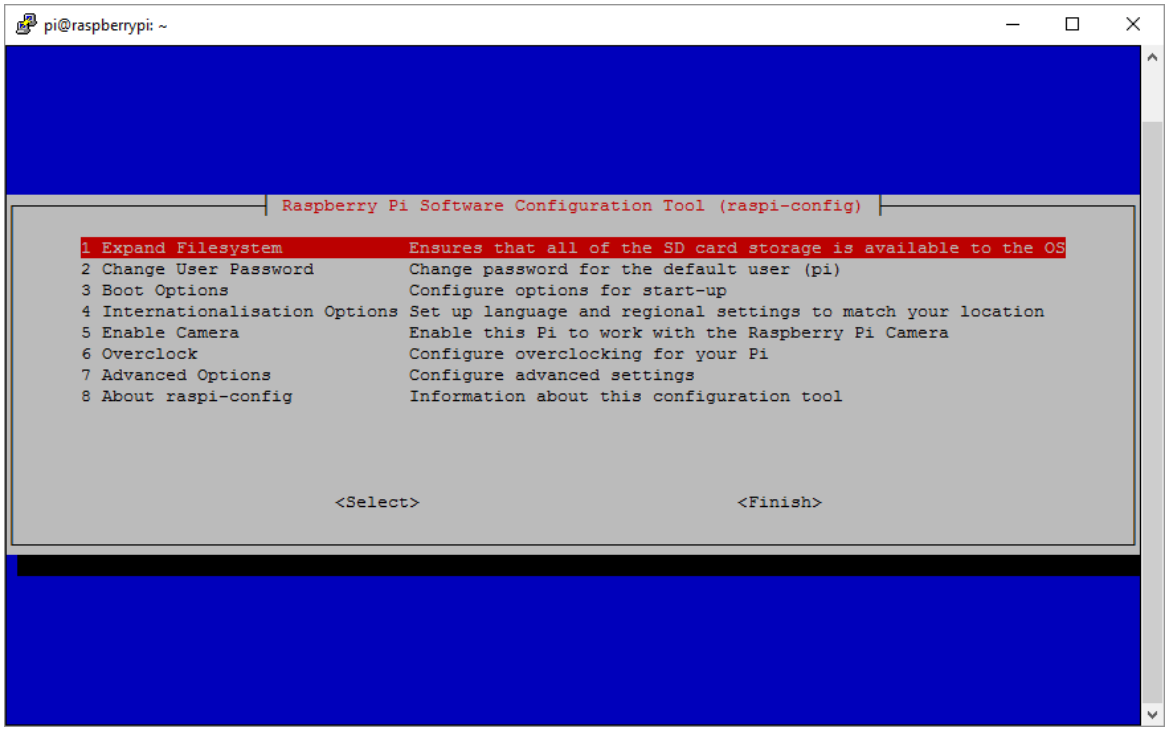


Figure 2.16 Step 7 Of Raspbian Setup

After these steps, the installation of the operating system for the raspberry battery is completed and the Raspberry pi is ready to use.

Figure 2.17 Step 8 Of Raspbian Setup

2.2.3 Raspberry Pi Camera Module v2

The Raspberry Pi Camera Module v2 replaced the original Camera Module in April 2016.

2.2.3.1 Specification

The Raspberry Pi Camera v2 is a high quality 8 mega pixel Sony IMX219 image sensor custom designed add-on board for Raspberry Pi, featuring a fixed focus lens. 8It's capable of 3280 x 2464-pixel static images, and supports 1080p30, 720p60 and 640x480p60/90 video

- High Quality
- High Functionality
- Compatible size with raspberry pi

Figure 2.18 Raspberry Pi Camera Module v2

2.2.4 OLED (Organic Light-Emitting Diode)

OLED is a self-light-emitting technology composed of a thin, multi-layered organic film placed between an anode and cathode in contrast to LCD technology, OLED does not require a backlight. OLED possesses high application potential for virtually all types of displays and is regarded as the ultimate technology for the next generation of flat panel displays

2.2.4.1 Specifications

WEA012864D-03 version 4 pin OLED display module is a diagonal size 0.96-inch COG OLED display with PCB board on it.

This WEA012864D-03 version OLED display is made of 128x64 pixels.

WEA012864D-03 module is built in with SSD1306BZ IC, it communicates via I2C interface only, VCC 3V /5V, I/O level 5V to 3V, with conversion circuit, 1/64 duty cycle.

The WEA012864D-03 model is having a smaller PCB than WEA012864D-01 version of which outline size is 27.3x27.3x2.37mm with mounting holes on board and 4 metal pins on module.

• Size: 0.96 inch

• 3V/5V Power supply

• 4 Pin OLED Display

• 128x64 pixels

Figure 2.19 OLED

2.2.5 USB Microphone

Microphone can use for USB interface**. 360 °** omnidirectional full point, anti-noise, can be used for recording. High-performance microphone, true sound reproduction.

Figure 2.20 USB Microphone

2.2.5.1 Specification

High sensitivity, effective distance of more than 2 meters, bid farewell to the traditional 3.5-pin wheat effective distance 0.5M-1M limitations

- Sensitivity: -47dB ± 4dB
- Working voltage: 4.5V
- Frequency response: 100 ~ 16kHz5. SNR: More than -67dB
- SNR: More than -67dB

2.2.6 Sound system

USB speakers and hands-free (jack 3.5 mm)

Figure 2.21 hands-free (jack 3.5 mm)

Figure 2.22 USB speakers

2.2.6.1 Specification

For Type USB Speakers

• Nominal voltage5V

• Rated power5W

• Rated frequency70-20K HZ

• Speaker interface AES/EBU 3.5mm

• Distortion rate0.3%

• SNR (Signal to Noise Ratio)50db

• Impedance4Ω

2.2.7 Cooling Fan for Raspberry Pi | 3010 Model

Increasing the load and processor's operations on CPU, which would produce extra heat with reduce performance and the speed of the processor is slowed down and limits the speed of operations. The fan is designed for prevent Pi from overheating**.**

Figure 2.23 Cooling Fan

2.2.7.1 Specification

The fan is designed for scenarios where the Raspberry Pi 4 has been overclocked and required to run for lengthy periods, which would produce extra waste heat. Once it reaches say 80°C, the Raspberry Pi 4 would normally throttle back CPU performance to reduce the temperature.

Rated Voltage: DC:5V

Current: 0.2 A

Noise: 24±10%dBA

Life: 35000 hours

2.3 Software requirements

There are several algorithms and available for different image processing techniques. The techniques used for this project are mainly text detection and text recognition

The manual selection of accurate algorithms in order to perform these techniques is extremely difficult as there is a countless amount of them. Developers have created an intelligent method for automatic selection of suitable algorithms called Open-Source Computer Vision (Open CV) that contains a large number of pre-installed libraries for image processing, by automatic mapping of imaging and selecting your desired technique it chooses the most relevant algorithms required for a process.

2.3.1 Introduction to OpenCV-Python

OpenCV was started at Intel in 1999 by Gary Bradsky and the first release came out in 2000. Vadim Pisarevsky joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle who won 2005 DARPA Grand Challenge. Later its active development continued under the support of Willow Garage, with Gary Bradsky and Vadim Pisarevsky leading the project. Right now, OpenCV supports a lot of algorithms related to Computer Vision and Machine Learning and it is expanding day-by day**.**

Open CV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel. The library is cross-platform, free for use and supports the deep learning frameworks.

The cross-platform library sets its focus on real-time image processing and includes patent-free implementations of the latest computer vision algorithms.

OpenCV comes with a programming interface to C, C++, Python and Android.

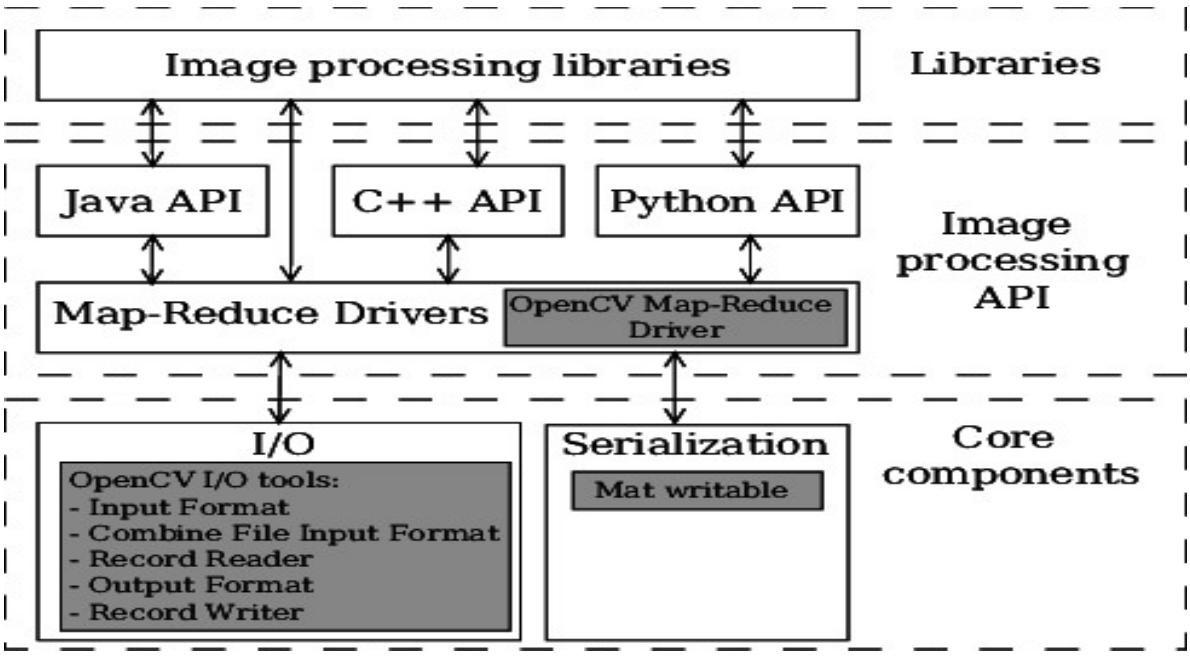


Figure 2.24 Architecture of Open CV framework

Open CV includes a statistical machine learning library that contains:

- Boosting
- Decision tree learning
- Gradient boosting trees
- Expectation-maximization algorithm
- Naive Bayes classifier
- Artificial neural networks
- Random forest

Chapter 3
Embedded Processing

3.1 Image Processing

3.1.1 Introduction

Generally, an image is a visual representation of such a thing. It is a picture which has been formed or copied and stored in electronic form. Also, it is a combination of two dimensional function f(x,y) where x&y the spatial domain coordinates, and at any pair of coordinates (x,y) the amplitude of (f) is known as the intensity of image at this level.

We will call the image a digital image if the x,y and the amplitude values of f are finite and discrete quantities. A digital image contains a finite number of elements that is known as pixels that each of them has a special value and location. Digital image processing performs automatic processing, manipulation and interpretation of such visual information. It focuses on the methods and implementations that we use to extract meaningful information and specific characteristics from the image.

These operations help to enhance or modify the image properties in order to achieve better results after classification. It plays an increasingly important role in many areas of our lives, including a wide range of disciplines and fields in science and technology, with applications such as photography, television, robotics, remote sensing and industrial inspection. It forms core research area within engineering and computer science disciplines too.[13]

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image.

Image processing basically includes the following three steps:

- Importing the image via image acquisition tools.
- Analyzing and manipulating the image.
- Output in which result can be altered image or report that is based on image analysis.

Image processing was one of the fundamental areas for most of the researchers to work on. Text detection and recognition are popular sub-parts of image processing. There are several algorithms available that can do it. There are many algorithms and techniques available for various image processing techniques. Text detection and recognition are the main techniques used in this project. Because there are so many of them, manually selecting accurate algorithms to perform these techniques is extremely difficult. Developers have created a smart method for automatically selecting of appropriate algorithms called Open-Source Computer Vision (Open CV) Which contains a large number of pre-installed libraries for image processing, and it chooses the most applicable algorithms required for an operation by automatically mapping imaging and selecting your desired technique.[14]

3.2 Open CV

3.2.1 Introduction

OpenCV is a library of programming functions mainly used for image processing. It is freely available on the open source Berkely Software Distribution (BSD) license. It was started as a research project by Intel. OpenCV contains various tools to solve computer vision problems. It contains low level image processing functions and high-level algorithms for face and object detection, feature matching and tracking. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS, and Android.

In simple words, with machine learning, computers are able to recognize and process the objects from the video/images as a human can be able to recognize.[15]

3.2.2 Application of Open CV with Python

With the help of Open CV in python, it is possible to process images, videos easily and can extract useful information out of that, as there are lots of functions available. Some of the common applications are:

1. *Image Processing:*

Images can be read, write, show, and processed with the OpenCV, like can generate a new image from that either by changing its shape, color, or extract something useful from the given one and write into a new image.

2. *Face Detection:*

Either from the live streaming using web camera or from the locally stored videos/images utilizing Haar- Cascade Classifiers.

3. *Face Recognition:*

It followed by face detection from the videos using open cv by drawing bounding boxes i.e., rectangle and then model training using ML algorithms to recognize faces.

4. *Object Detection:*

An object detection algorithm can be used to detect objects from the image, videos either moving or stationary objects.[4]

3.2.3 Open CV for image mapping

At the beginning, to implement text detection in the paper we will use open cv with python. Initially image acquisition is done with the help of raspberry pi camera. First image of the paper is captured. This image is saved as reference image at a particular location specified in the program. We will import open cv

and NumPy libraries as shown below. NumPy is a Python library used for working with arrays. NumPy stands for Numerical Python.

```
import cv2
import numpy as np
```

Figure 3.1 importing OpenCV

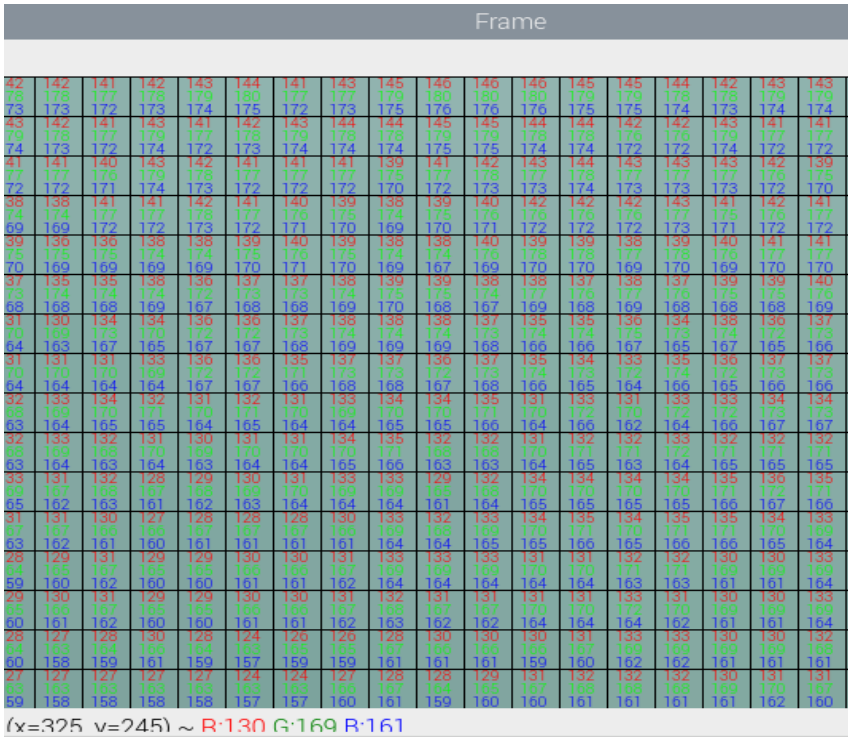


Figure 3.2 importing OpenCV (NumPy arrays)

In addition to this, we will filter the image and each rectangular will cropped and convert it to array, so we deal with it in best way. OpenCV images are stored as three-dimensional NumPy arrays. When you read in images using the library, they are represented as NumPy arrays. With NumPy you can make fast operations on numerical arrays, no matter which dimension, shape they are.

3.3 Threshold Colour Filter

3.3.1 Introduction

Every pixel in an image is made up of three components, according to the color space of that image, those three components descript different properties for that pixel, they each have values ranging from 0 – 255. The standard color space that's always used with images is the RGB color space, it describes the pixel's color based on the amount of the colors red, green, and blue in the pixel.

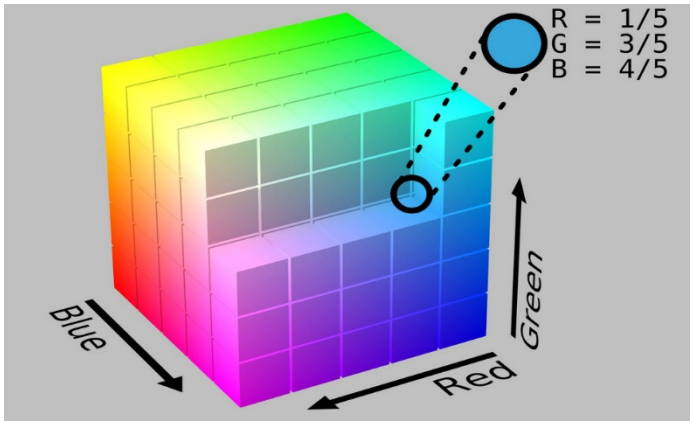


Figure 3.3: RGB colour space.

Thresholding is one of the most common (and basic) segmentation techniques in computer vision and it allows us to separate the foreground (i.e., the objects that we are interested in) from the background of the image.[15]

3.3.2 Thresholding Concept

Thresholding is the binarization of an image. In general, we seek to convert a grayscale image to a binary image, where the pixels are either 0 or 255. a simple thresholding example would be selecting a threshold value T, and then setting all pixel intensities less than T to 0, and all pixel values greater than T to 255. In this way, we are able to create a binary representation of the image.

For example, take a look at the (grayscale) PyImageSearch logo below and its thresholded counterpart:

Figure 3.4: Thresholding the PyImageSearch logo.

On the left, we have the original PyImageSearch logo that has been converted to grayscale. And on the right, we have the thresholded, binary representation of the PyImageSearch logo. To construct this thresholded image I simply set my threshold value T=225. That way, all pixels p in the logo where p < T are set to 255, and all pixels p >= T are set to 0. By performing this thresholding, I have been able to segment the PyImageSearch logo from the background. Normally, we use thresholding to focus on objects or areas of particular interest in an image.

3.3.3 Filtration Workflow

3.3.3.1 Methodology

Using the RGB color space and applied thresholds to the percentage of the highest significant channel (HSC) and the other two least significant channels (LSCs) in the pixel. For example, if we want to detect all the variations of the red pixels in an image, we firstly look at the code of the pure version of that color. The code of a pure red pixel is (255, 0, 0), thus the HSC here is the red channel and the LSCs are the green and the blue channels. So, to get all the variations of red, we can take all pixels that have the red percentage on them between 0.5 – 1, the green values between 0 – 100 and the blue values between 0

– 100. A volume graph of this relation and the thresholded part (Red slice) looks like this:

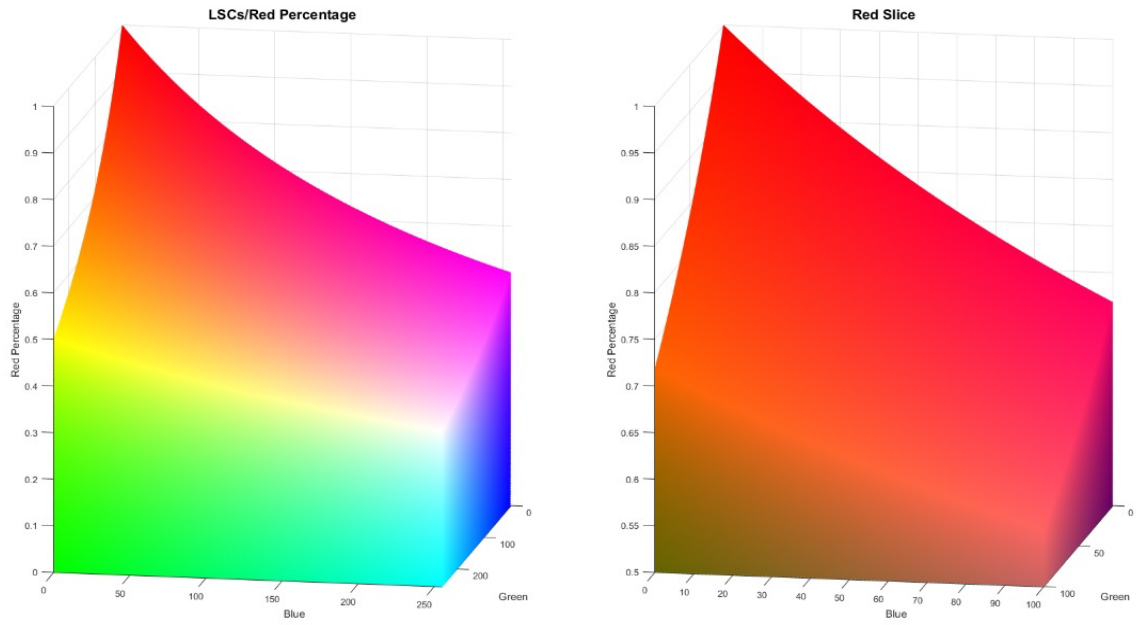


Figure 3.5: Volume Graph of Lscs/Red Percentage and Red Slice

Using a different HSC in the relation graph swaps color regions together. for example, to detect all the variations of the green pixels in an image, we will use the HSC of a pure green pixel, which is the green channel:

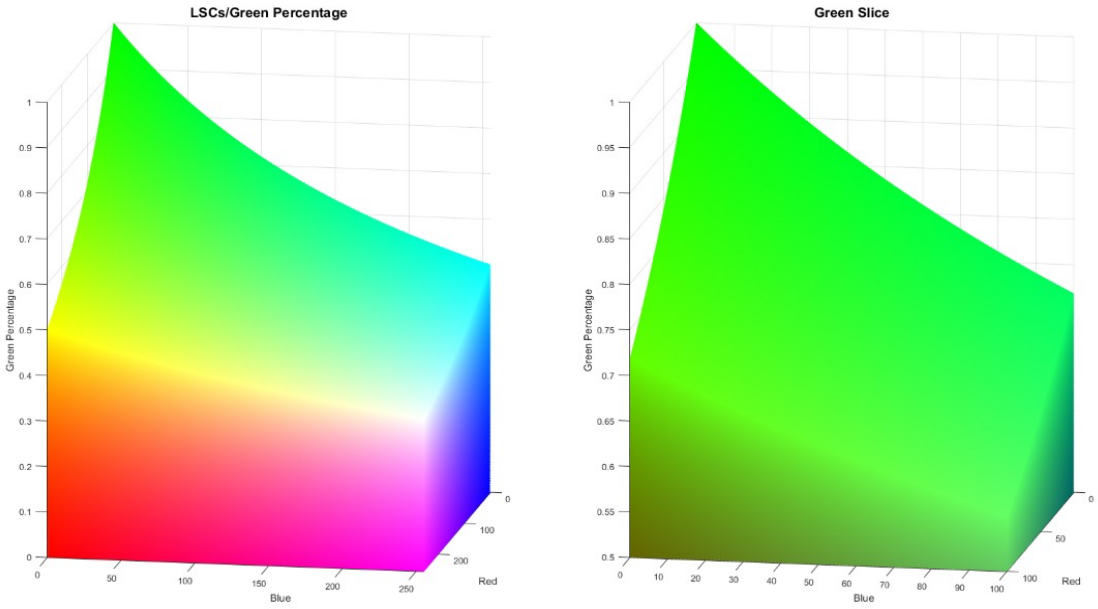


Figure 3.6: Volume Graph of Lscs/Green Percentage and Green Slice

As you can see in the graph, the red region has been swapped with the green region, and the pink region has been swapped with the cyan region, the same applies when we try to detect all the variations of the blue pixels in an image. The same applies when we try to detect all the variations of the black pixels in an image.

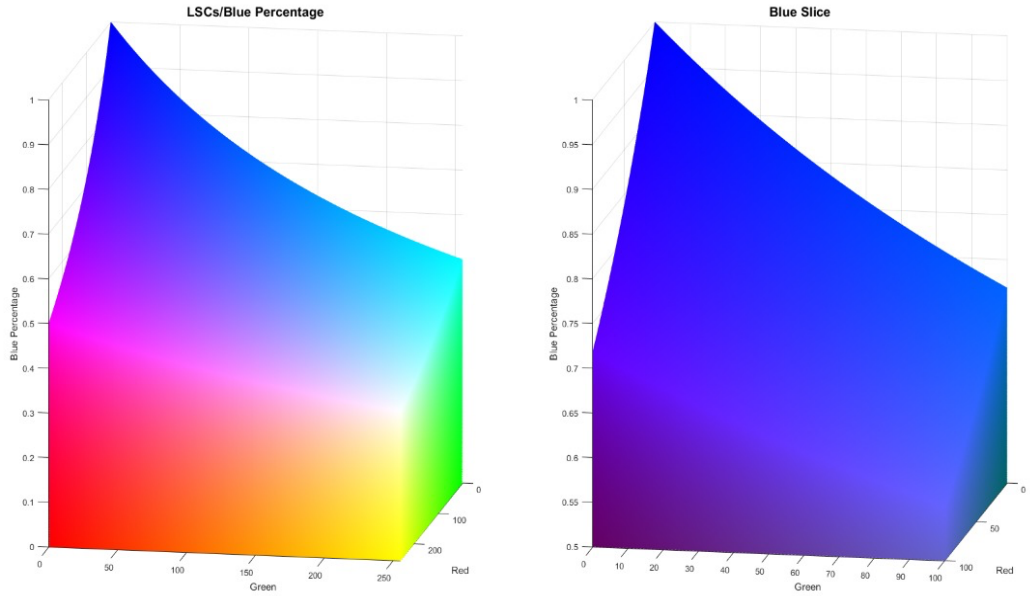


Figure 3.7: Volume Graph of Lscs/Blue Percentage and Blue Slice.

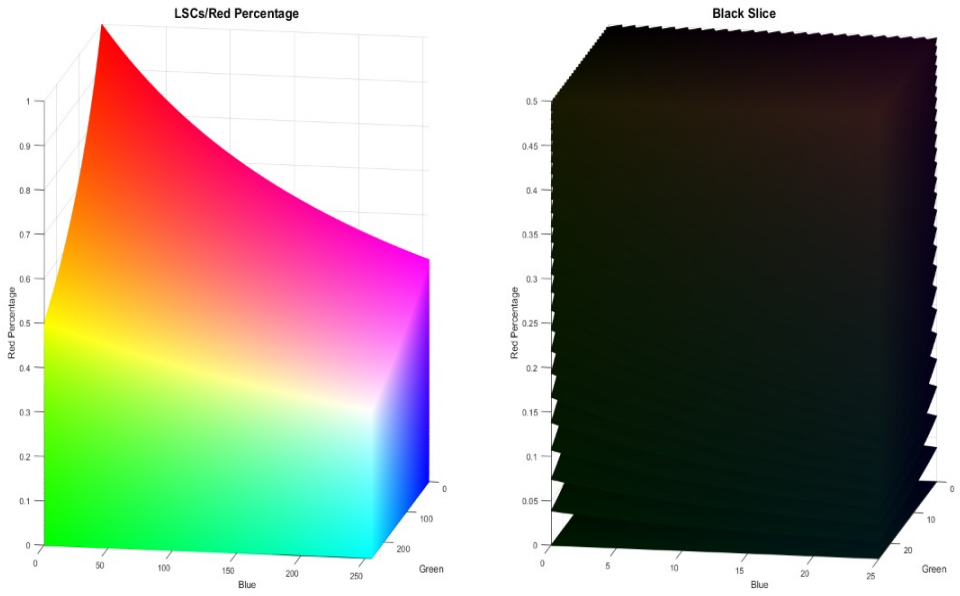
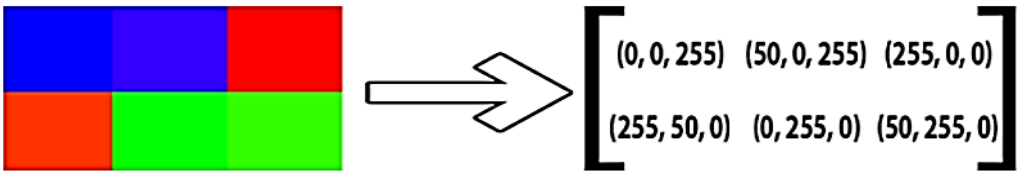


Figure 3.8: Volume Graph of Lscs/Black Percentage and Blue Slice.

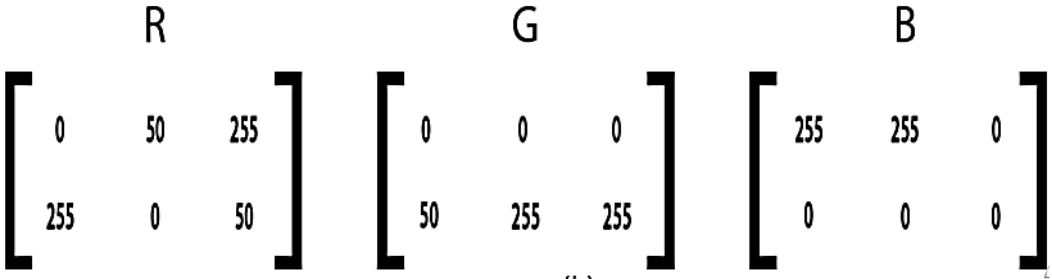
3.3.3.2 Mathematical Operations

we will use a simple 6-pixel image as general example; our aim is to detect the red pixels only. so, the HSC here will be the red channel. Firstly, we should convert the image to a 3D matrix as shown in figure3.9 (a). Then we will split this 3D matrix into three 2D matrixes, every one of them will hold all the values for a color channel, so then we get the three matrixes R, G and B as shown in figure3.9 (b). Then, to get the red percentage matrix, we should substitute by the values of each variable from these three matrixes in the following relation $\frac{R}{R+G+B}$ then the output matrix will be the red percentage matrix as shown in figure3.9 (c). then we apply our thresholds, 0.5 - 1 for the red percentage matrix, 0 – 100 for the G and B matrixes, the numbers that satisfy our thresholds will be replaced with 1, otherwise, it will be 0 as shown in figure3.9 (d).

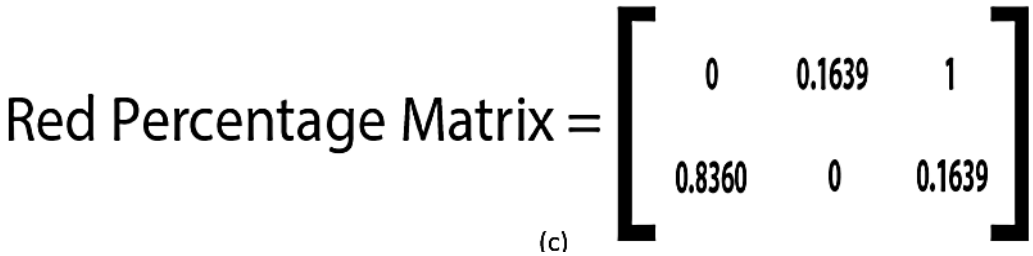
Now for these three matrices. We will multiply each number with its counterpart in the other two matrices. As a result, we will get the result matrix. then we will convert the result matrix to a 3D matrix again by replacing every number in the matrix with a vector that has three copies of that number as shown in figure3.9 (e). finally, we want to make our result be a black and white image, where the white pixels the ones that satisfy our thresholds, and the black pixels are the ones that didn't satisfy them, so we will multiply the result matrix by 255 so that all the (1, 1, 1) vectors change to (255, 255, 255) which is the code for the pure white color as shown in figure3.9 (f).



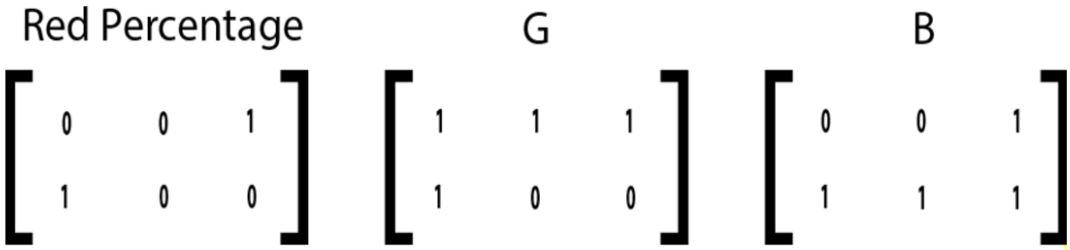
(a)



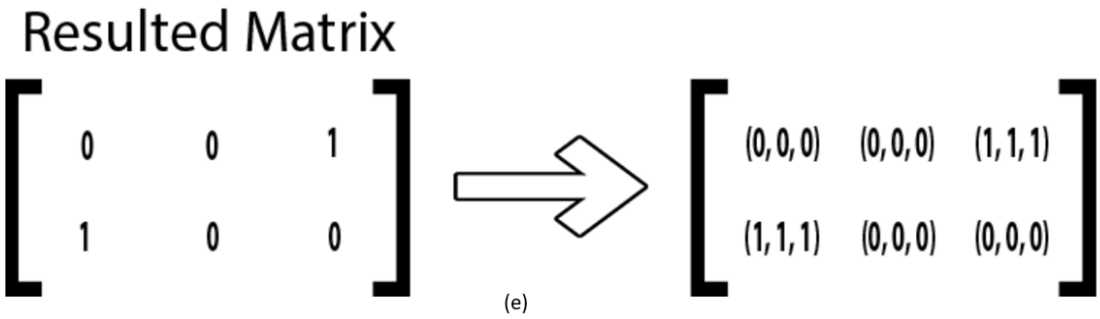
(b)



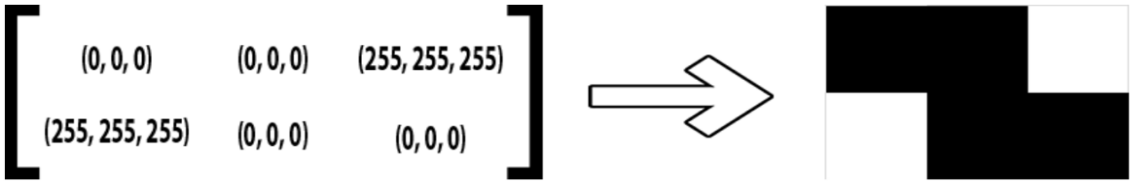
(c)



(d)



(e)



(f)

Figure 3.9: mathematical operation

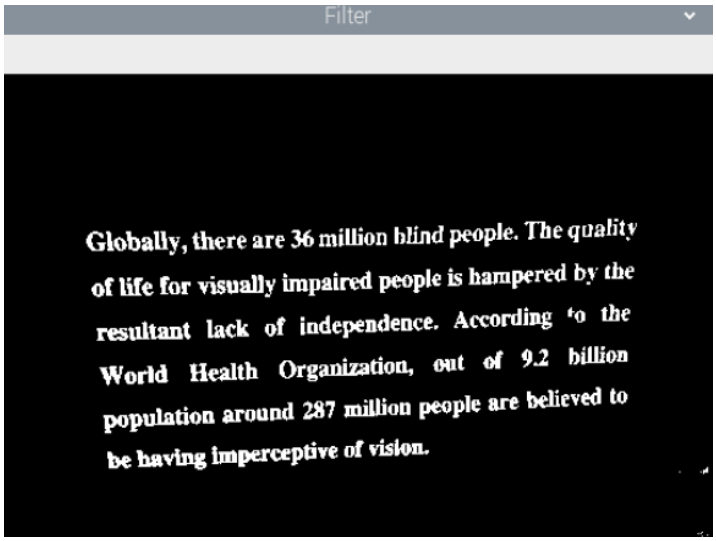


Figure 3.10: filtered frame

3.4 Object detection

3.4.1 Object detection concept

Object Detection is the process of finding real-world object instances like car, bike, TV, flowers, and humans in still images or Videos. It allows for the recognition, localization, and detection of multiple objects within an image which provides us with a much better understanding of an image as a whole. The location is pointed out by drawing a bounding box around object. The bounding box may or may not accurately locate the position of the object. The ability to locate the object inside an image defines the performance of the algorithm used for detection. Face detection is one of the examples of object detection.[13]

These object detection algorithms might be pre-trained or can be trained from scratch.

In this Project, we will focus on OpenCV Object Detection.

3.4.2 Applications of Object Detection

- **People Counting**

Object detection can be also used for people counting, it is used for analyzing store performance or crowd statistics during festivals. These tend to be more difficult as people move out of the frame quickly.

It is a very important application, as during crowd gathering this feature can be used for multiple purposes. [16]

Figure 3.11 Object Detection People counting

- **Self-Driving Cars**

Self-driving cars are the Future, there is no doubt in that. But the working behind it is very tricky as it combines a variety of techniques to perceive their surroundings, including radar, laser light, GPS, odometry, and computer vision.

Figure 3.12 Object Detection Self-Driving Car

- **Facial Recognition**

A deep learning facial recognition system called the “Deep Face” has been developed by a group of researchers in the Facebook, which identifies human faces in a digital image very effectively. Google uses its own facial recognition system in Google Photos, which automatically segregates all the photos based on the person in the image. There are various components involved in Facial Recognition like the eyes, nose, mouth and the eyebrows.

Figure 3.13 Object Detection Facial Recognition

3.4.3 Object Detection Workflow

Every Object Detection Algorithm has a different way of working, but they all work on the same principle.

Feature Extraction: They extract features from the input images at hands and use these features to determine the class of the image. Be it through Open CV or Deep Learning. [16]

Figure 3.14 Object Detection Workflow Feature Extraction

3.4.4 OpenCV for Object Detection

It’s far easier to write code for images captured in controlled lighting conditions than in dynamic conditions with no guarantees. If you are able to

control the environment and, most importantly, the lighting when you capture an image, the easier it will be to write code to process the image. With controlled lighting conditions, you're able to hard-code parameters, including:

- Amount of blurring
- Edge detection bounds
- Thresholding limits
- Etc.

Essentially, controlled conditions allow you to take advantage of your a priori knowledge of an environment and then write code that handles that specific environment rather than trying to handle every edge case or condition. of course, controlling your environment and lighting conditions isn't always possible.[17]

3.4.4.1 morphological operation

Morphological operations are applied to grayscale or binary images and are used for preprocessing for OCR algorithms, detecting barcodes, detecting license plates, to shapes and structures inside of images. And more. And sometimes a clever use of morphological operations can allow you to avoid more complicated (and computationally expensive) machine learning and deep learning algorithms.

We can use morphological operations to increase the size of objects in images as well as decrease them. We can also utilize morphological operations to close gaps between objects as well as open them.

Morphological operations "probe" an image with a structuring element. This structuring element defines the neighborhood to be examined around each pixel. And based on the given operation and the size of the structuring element we are able to adjust our output image.

Morphological operations we'll be covering include:

- Erosion
- Dilation
- Opening
- Closing
- Morphological gradient
- Black hat
- Top hat (also called "White hat")

3.4.4.1.1 Dilation

The opposite of an erosion is a dilation. A dilation will grow the foreground pixels. Dilations increase the size of foreground objects and are especially useful for joining broken parts of an image together. Dilations works by defining a structuring element and then sliding this structuring element from left-to-right and top-to-bottom across the input image. A center pixel p of the structuring element is set to white if ANY pixel in the structuring element is > 0.[18]

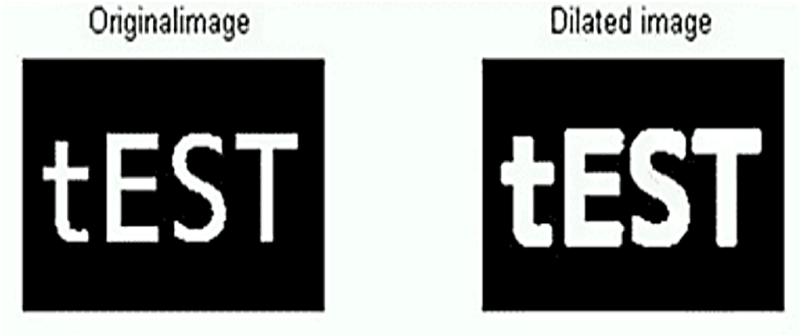


Figure 3.15 original /dilated image

3.4.4.2 OpenCV Contour

Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same colour or intensity.

The contours are a useful tool for shape analysis and object detection and recognition. To draw the contours, cv. drawContours function is used. It can also be used to draw any shape provided you have its boundary points. Its first argument is source image, second argument is the contours which should be passed as a Python list, third argument is index of contours (useful when drawing individual contour. To draw all contours, pass -1) and remaining arguments are color, thickness etc.

Contour Approximation Method: This is the third argument in cv.findContours function. If you pass (cv.CHAIN_APPROX_NONE), all the boundary points are stored. But actually, we need just two end points of that line. This is what (cv.CHAIN_APPROX_SIMPLE) does. It removes all redundant points and compresses the contour, thereby saving memory.

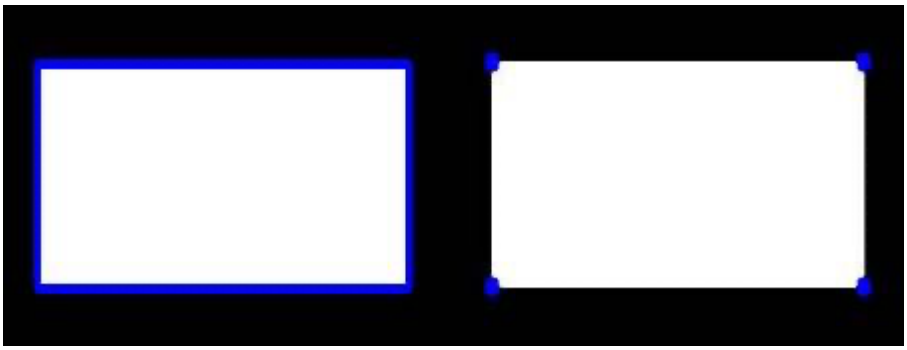


Figure 3.16 CHAIN_APPROX_NONE / CHAIN_APPROX_SIMPLE

Straight Bounding Rectangle: It is a straight rectangle; it doesn't consider the rotation of the object. So, area of the bounding rectangle won't be minimum. It is found by the function cv.boundingRect().[19]

```
filtered = black_filter(f)

rect_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (20, 20))
dilation = cv2.dilate(filtered, rect_kernel, iterations=1)

contours, _ = cv2.findContours(dilation, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

for cnt in sorted(contours, key=lambda x: cv2.contourArea(x), reverse=True)[0:1]:
    x, y, w, h = cv2.boundingRect(cnt)
    rect = cv2.rectangle(f, (x, y), (x + w, y + h), (0, 255, 0), 2)

cv2.imshow("Frame", f)
cv2.imshow("Filter", filtered)
cv2.waitKey(1)
```

Figure 3.17 Image modifications

Figure 3.18 Sample of Cropped rectangular

3.5 OCR

3.5.1 OCR overview

After filter the image, detect, and localize the bounding box coordinates of text contained in an image using OpenCV, each rectangular will cropped and convert it to array so we deal with it in best way. Then we need to recognize this text. The next step is to take each of these areas containing text and OCR the text using Tesseract.

OCR is an abbreviation of optical character recognition method, it is used to convert typed, printed, or handwritten text into machine-encoded text. There are

some OCR software engines which try to recognize any text in images such as Tesseract and ABBYY FineReader. In this project Tesseract version 4 is used because it is the best open-source OCR engines. These digital versions can be highly beneficial to children and young adults who struggle to read. And that's why the digital text may be utilized with several software packages that help with readability.

3.5.2 Tesseract

Tesseract, a highly popular OCR engine, was originally developed by Hewlett Packard in the 1980s and was then open sourced in 2005. Google adopted the project in 2006 and has been sponsoring it ever since. As of 2018, it now includes built-in deep learning capability making it a robust OCR tool.

Tesseract was in the top three OCR engines in terms of character accuracy in 1995.[8] It is available for Linux, Windows, and Mac OS X. However, due to limited resources it is only rigorously tested by developers under Windows and Ubuntu. The initial versions of Tesseract could only recognize English-language text. Tesseract v2 added six additional Western languages (French, Italian, German, Spanish, Brazilian Portuguese, Dutch). Version 3 extended language support significantly to include ideographic (Chinese & Japanese) and right-to-left (e.g., Arabic, Hebrew) languages, as well as many more scripts. New languages included Arabic, Bulgarian, Catalan, Chinese (Simplified and Traditional), Croatian, Czech, Danish, German (Fraktur script), Greek, Finnish, Hebrew, Hindi, Hungarian, Indonesian, Japanese, Korean, Latvian, Lithuanian, Norwegian, Polish, Portuguese, Romanian, Russian, Serbian, Slovak (standard and Fraktur script), Slovenian, Swedish, Tagalog, Tamil, Thai, Turkish, Ukrainian and Vietnamese. V3.04, released in July 2015, added an additional 39 language/script combinations, bringing the total count of support languages to over 100. New language codes included:

amh (Amharic), asm (Assamese), aze_cyrl (Azerbaijana in Cyrillic script), bod (Tibetan), bos (Bosnian), ceb (Cebuano), cym (Welsh), dzo (Dzongkha), fas (Persian), gle (Irish), guj (Gujarati), hat (Haitian and Haitian Creole), iku (Inuktitut), jav (Javanese), kat (Georgian), kat_old (Old Georgian), kaz (Kazakh), khm (Central Khmer), kir (Kyrgyz), kur (Kurdish), lao (Lao), lat (Latin), mar (Marathi), mya (Burmese), nep (Nepali), ori (Oriya), pan (Punjabi), pus (Pashto), san (Sanskrit), sin (Sinhala), srp_latn (Serbian in Latin script), syr (Syriac), tgk (Tajik), tir (Tigrinya), uig (Uyghur), urd (Urdu), uzb (Uzbek), uzb_cyrl (Uzbek in Cyrillic script), yid (Yiddish).

Version 4 adds LSTM the Long Short-Term Memory (LSTM) OCR model which is far more accurate than the previous versions of Tesseract and models for many additional languages and scripts, bringing the total to 116 languages. [20]

Tesseract can work very well under controlled conditions but will perform quite poorly if there is a significant amount of noise or your image is not properly pre-processed and cleaned before applying Tesseract. That is the reason why we made preparatory operations on the image before it reached to OCR.[20]

3.5.3 Long Short-Term Memory (LSTM) network

Just as deep learning has impacted nearly every facet of computer vision, the same is true for character recognition and handwriting recognition. Deep learning-based models have managed to obtain unprecedented text recognition accuracy, far beyond traditional feature extraction and machine learning approaches. It was only a matter of time until Tesseract incorporated a deep learning model to further boost OCR accuracy.

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can process not only single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected and writing recognition, speech recognition and anomaly detection in network traffic or IDSs (intrusion detection systems).

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

LSTM networks are well-suited to classifying, processing, and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. [21]

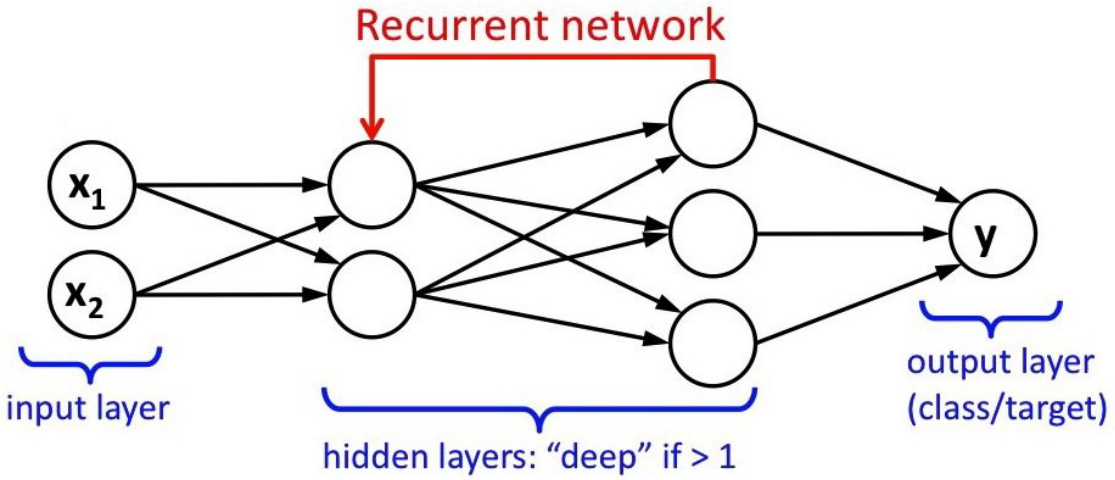


Figure 3.19: Long Short-Term Memory (LSTM) cell

3.5.4 Tesseract OCR Mechanism

Tesseract OCR process consists of multiple stages:

- First: pre-processing

The main goal of this step is to reduce the noise that resulted from scanning the document where the characters might be broken or smeared and causes poor rates of recognition. pre-processing is done by smoothing the digitized characters through filling and thinning. Another aim of this step is to normalize the data to get characters of uniform size, rotation, and slant.
Moreover, compression in the amount of information to be kept through thresholding and thinning techniques.

- Second: Segmentation

In this process, the characters or words will be isolated. The words will be segmented into isolated characters that are recognized separately. Most of OCR algorithms segment words into isolated characters which are recognized individually. Usually, segmentation is done by isolating every connected component.

- Third: Feature Extraction

This process will capture the significant features of symbols and it has two types of algorithms which are pattern recognition and feature extraction/detection.

- Fourth: Classification

OCR systems use the techniques of pattern recognition that assigns an unknown sample into a predefined class. One of the ways to help in character classifying is using English dictionary.

- Fifth: post processing

This process includes grouping. In grouping, the symbols relate to strings. The result of plain symbol recognition in the text is a group of individual symbols.

```
filtered = black_filter(f)

rect_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (20, 20))
dilation = cv2.dilate(filtered, rect_kernel, iterations=1)

contours, _ = cv2.findContours(dilation, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

text = ""
for cnt in sorted(contours, key=lambda x: cv2.contourArea(x), reverse=True)[0:1]:
    x, y, w, h = cv2.boundingRect(cnt)
    cropped = f[y:y + h, x:x + w]

    temp = pytesseract.image_to_string(cropped)
    text += " " + temp

text = re.sub(r'[^\x00-\x7f]', r'', text).strip().replace("\n", " ")
print(f"----------\n{text}\n----------")
```

Figure 3.11: text recognition.

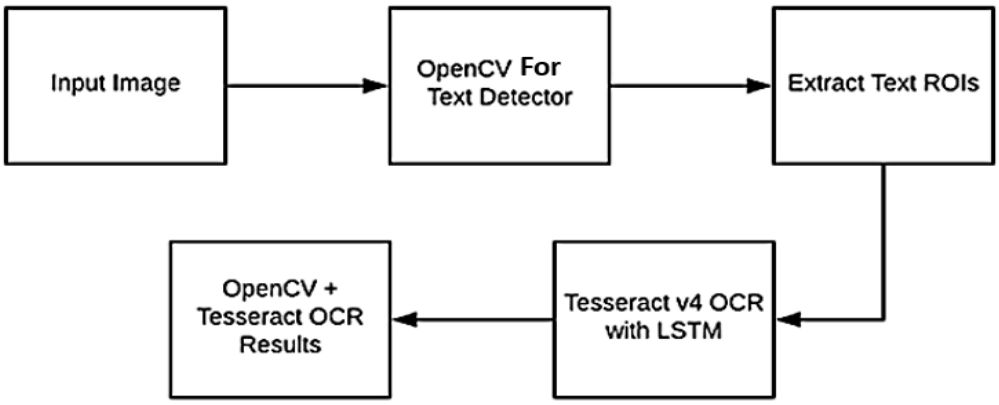


Figure 3.20: The OpenCV + OCR pipeline.

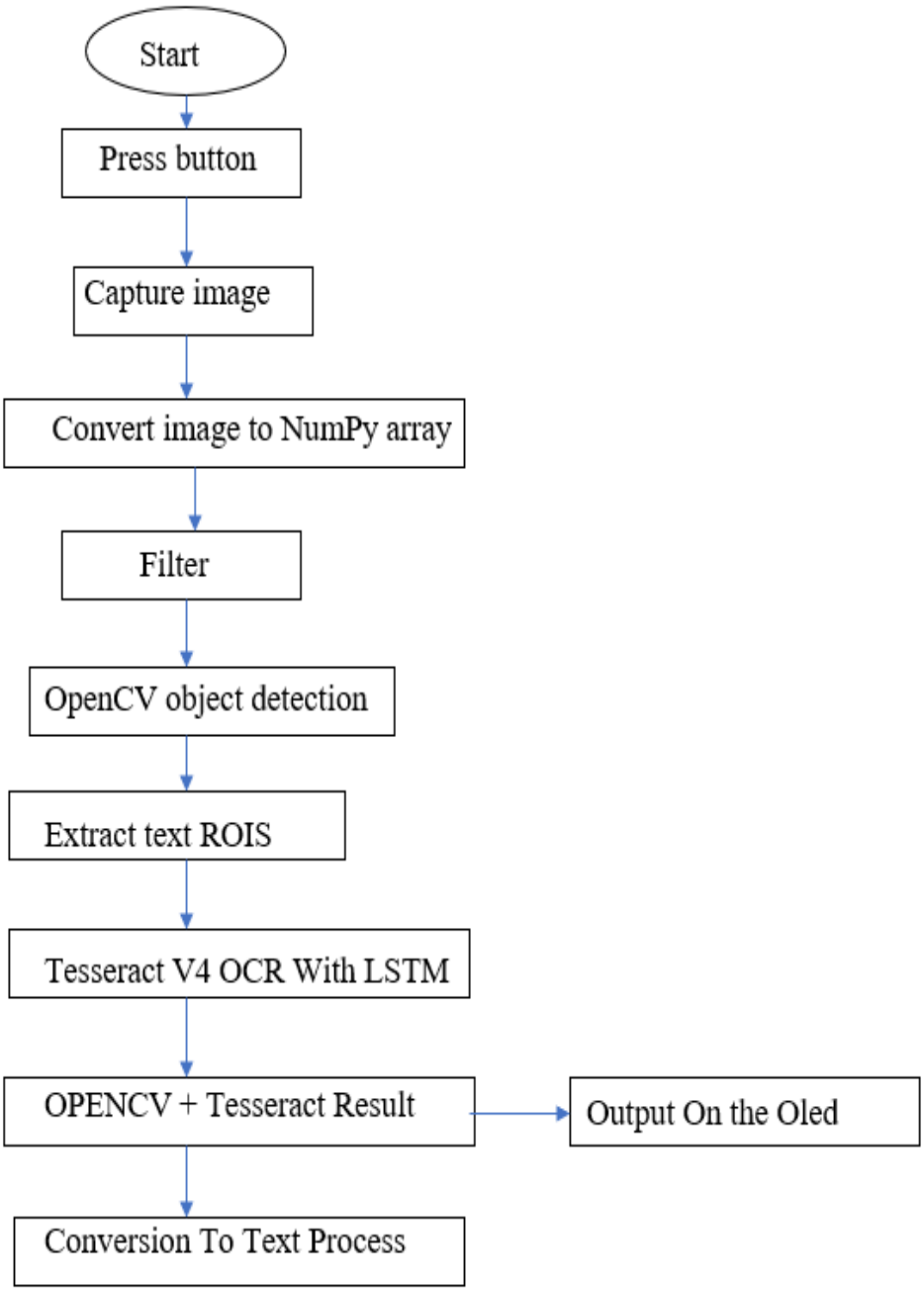


Figure 3.21 text detection and recognition flow chart of a captured image.

3.6 voice computing in python.

3.6.1 The importance of text to voice "TTS" conversion.

• Why the sudden need for TTS conversion?

A simple task such as reading can be a real struggle for many people especially those with disabilities including vision impairment or dyslexia or many other reasons. Thereby speech recognition is a vital assistive technology came to remove environmental barriers for people with a wide range of disabilities.

3.6.2 Text to voice conversion.

1. What is TTS?

is the ability of a computer software to identify and understand words and phrases and produce an artificial humans' speech.

2. How it works?

the system uses an acoustic model to read the processed text. The ML algorithm establishes the connection between phonemes and sounds, giving them accurate intonations. Then the system uses a sound wave generator to create a vocal sound. The frequency characteristics of phrases obtained from the acoustic model are eventually loaded into the sound wave generator.(1)

3. Text to voice conversion process parts.

TTS conversion composed of two parts the front-end process and a back-end process.

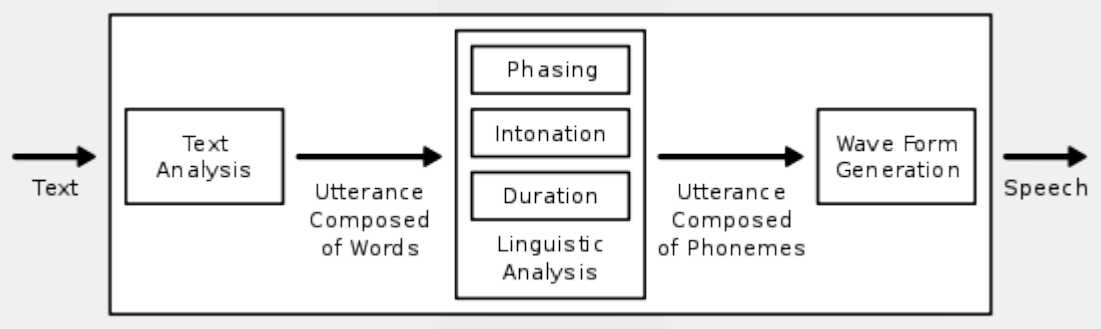


Figure 3.22 TTS conversion process parts.[2]

a. The front-end process where NLP takes place as grapheme to phoneme transcription occur. The front-end process has two major tasks. First, it converts raw text containing symbols like numbers and abbreviations into the equivalent of written-out words. This process is often called text normalization. Then it assigns phonetic transcriptions to each word, and divides and marks the text into prosodic units, like phrases, and sentences. The process of assigning phonetic transcriptions to words is called text-to-phoneme conversion. Phonetic transcriptions and prosody information together make up the symbolic linguistic representation that is output by the front-end.(3)

b. The back-end process—often referred to as the synthesizer—where DSP takes place. It is responsible of converting the symbolic linguistic representation into sound. In certain systems, this part includes the computation of the target prosody (pitch contour, phoneme durations), which is then imposed on the output speech.(4)

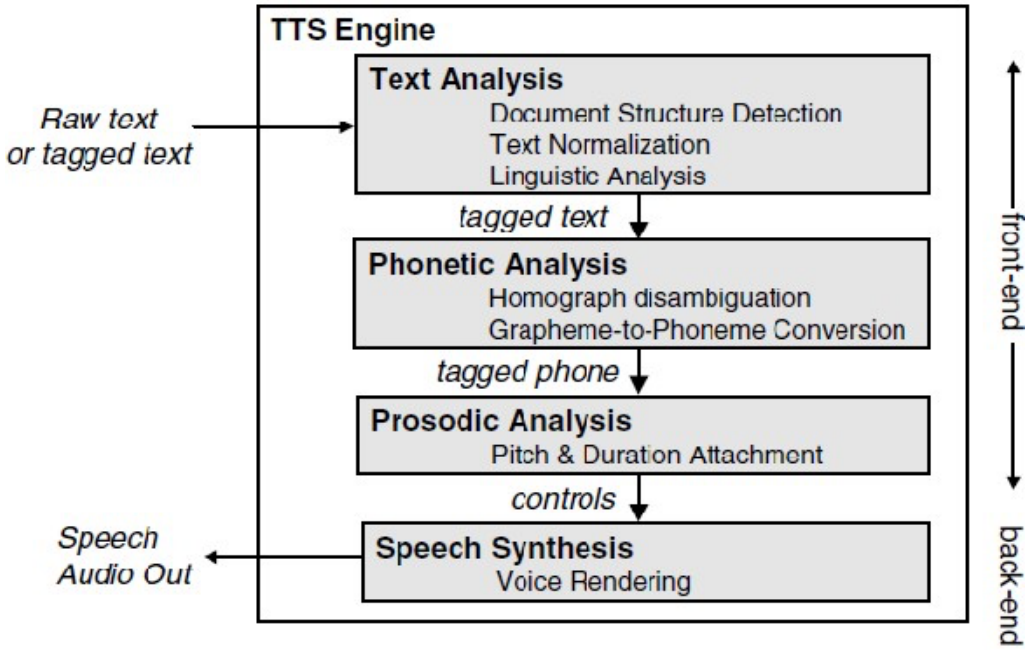


Figure 3.23 Chart front-end and back-end process.(5)

3.6.3 Speech Synthesis concept

1. what is speech synthesis?

speech synthesis is used by developers for the artificial production of human speech.[6]

2. The most important qualities of a speech synthesis system are:

a. Naturalness:[7] describes the quality of the speech generated in terms of how closely the output sounds like human speech.

b. Comprehensibility:[7] is the ease with which the output is understood.

3. There exist several different methods to synthesize speech.

a. Articulatory synthesis refers to computational techniques for synthesizing speech based on models of the human vocal tract and the articulation processes occurring there. It uses computational biomechanical models of speech production, such as models for the glottis (that generates the periodic and aspiration excitation) and the moving vocal tract.[8]

b. Concatenative synthesis is based on high-quality audio clips recordings, which are combined together to form the speech. Generally, concatenative synthesis produces the highest quality of audio in terms of intelligibility. However, such systems are very time consuming because they require huge databases, and hard-coding to form these words.[9]

c. Formant synthesis it does not use human speech samples at runtime. Instead, the synthesized speech output is created using artificial signals based on a set of specified rules mimicking the formant structure and other spectral properties of natural speech.

The synthesized speech is produced using an additive synthesis and an acoustic model (physical modelling synthesis). The acoustic model uses parameters like, voicing, fundamental frequency, noise levels, etc. that varied over time to create a waveform of artificial speech.(9)

As a result, formant synthesize method was chosen to be the most suitable for this project as it has many advantages. AS It is reliably intelligible at very high speeds, avoiding the acoustic glitches that commonly plague concatenative systems. Also, formant synthesize is less dependent on a speech corpus to produce the output speech. Moreover, it is well-suited for embedded systems; where memory and microprocessor power are limited.

Nonetheless formant synthesize has disadvantages as well. For example, it has low naturalness; as it generates artificial robotic sounding speech that would never be mistaken for human speech. Additionally, it is difficult to design rules that specify the timing of the source and the dynamic values of all filter parameters for even simple words.

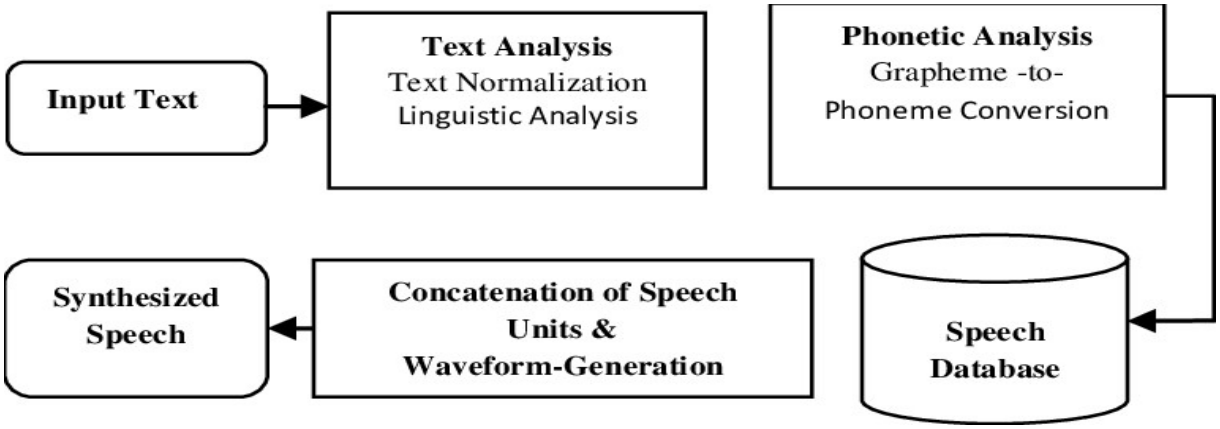


Figure 3.24 TTS Synthesis system.(10)

3.6.4 Method used for text to speech conversion.

The library used in python is pyttsx3.

it is a text-to-speech conversion library in Python.

3.6.3 Pyttsx3.

1- **What is pyttsx3?** is a text-to-speech conversion library in Python.

2- **Why was pyttsx3 chosen?**

Rather than saving the text as audio file, pyttsx3 actually speaks it there. This makes it more reliable to use for voice-based projects. Furthermore, pyttsx3 is compatible with both Python 3 and Python 2 and supports multiple TTS engines. Also contrary to gTTS API "Google Text to Speech "it can work offline. Moreover, it is able to save output as a wav file. Additionally, pyttsx3 is very easy to use.

However, as pyttsx3 has so many advantages over other API types it also has it is own downfalls as it generates audio like a metallic voice or robotic voice. As well pyttsx3 dese not support as many languages as gTTS API

3- **Ptysx3 library instillation**

Pip install pyttsx3

4- **Supported drivers by pyttsx3**

The pyttsx3.init() documentation explains how to select a specific synthesizer by name as well as the default for each platform.

pyttsx3 includes drivers for the following text-to-speech synthesizers.

a. sapi5 – SAPI5 on Windows
b. nsss – NSSpeechSynthesizer on Mac OS X
c. espeak – eSpeak on every other platform
d.

3.6.6 Espeak

1- What is espeak?

is a compact open source software speech synthesizer for languages.

2- Espeak speech synthesize method.

ESpeak uses a formant synthesis method. This allows many languages to be provided in a small size. The speech is clear, and can be used at high speeds, but is not as natural or smooth as larger synthesizers which are based on human speech recordings.

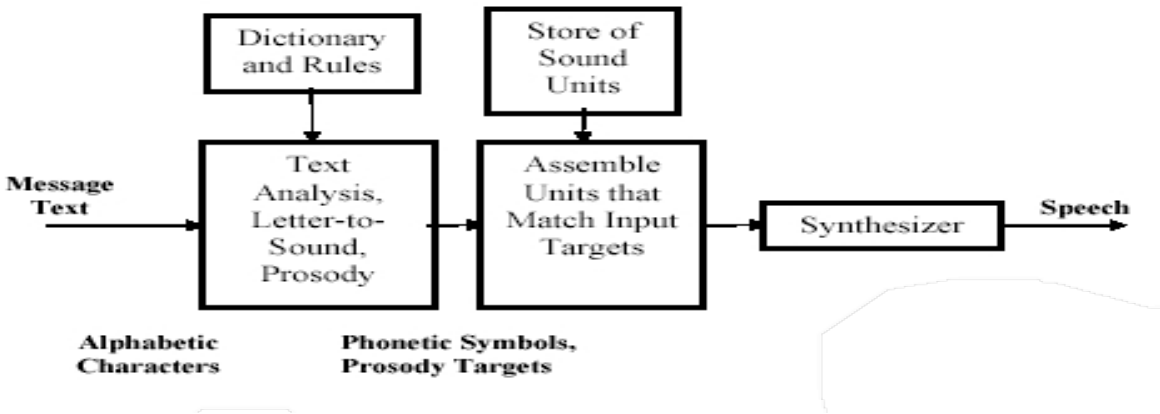


Figure 3.25 TTS conversion in python.[11]

3- Espeak installation

1- Place the espeek executable file in the command path (sudo apt-get install espeak).

2- place the "espeak-data" as /usr/share/espeak-data.

3.2.7 TTS conversion process

1- Hardware requirments:

a. Headphones.

2- Software requirments:

a. Espeak engine.

b. Python 3 libraries.

3- Code:

```
import cv2
import pytesseract
import pyttsx3



cap = cv2.VideoCapture(0)
engine = pyttsx3.init()

while(True):
    t, f = cap.read()
    cv2.imshow("Frame", f)

    if cv2.waitKey(1) & 0xFF == ord('s'):
        text = pytesseract.image_to_string(f)
        print(f"----------\n {text}\n----------")
        engine.say(text)
        engine.runAndWait()
        engine.stop()



cv2.destroyAllWindows()
```

Figure 3.26 TTS conversion code in python.

4- Process:

1- Install espeak engine on raspberry pi that has the tools needed for TTS conversion.

2- import pytesseract lipraries for OCR (the processe of images to text conversion).

3- Import pytsx3 API liperaries to be able to deal with espeak engine from python directly.

4- For TTS conversion first the front-end process occur where phonetic transcriptions assigned to each word, and divides and marks the text into prosodic units, like phrases, and sentences.

5- Next the back-end operations “synthesizer” takes place as it converts the symbolic linguistic representation into sound.

6- This is where espeak engine tools do the TTS conversion using formant synthesis method. This allows many languages to be provided in a small size also for the speech to be clear, and can be used at high speeds.

7- Final the speech output can be heard from speaker/headphones.

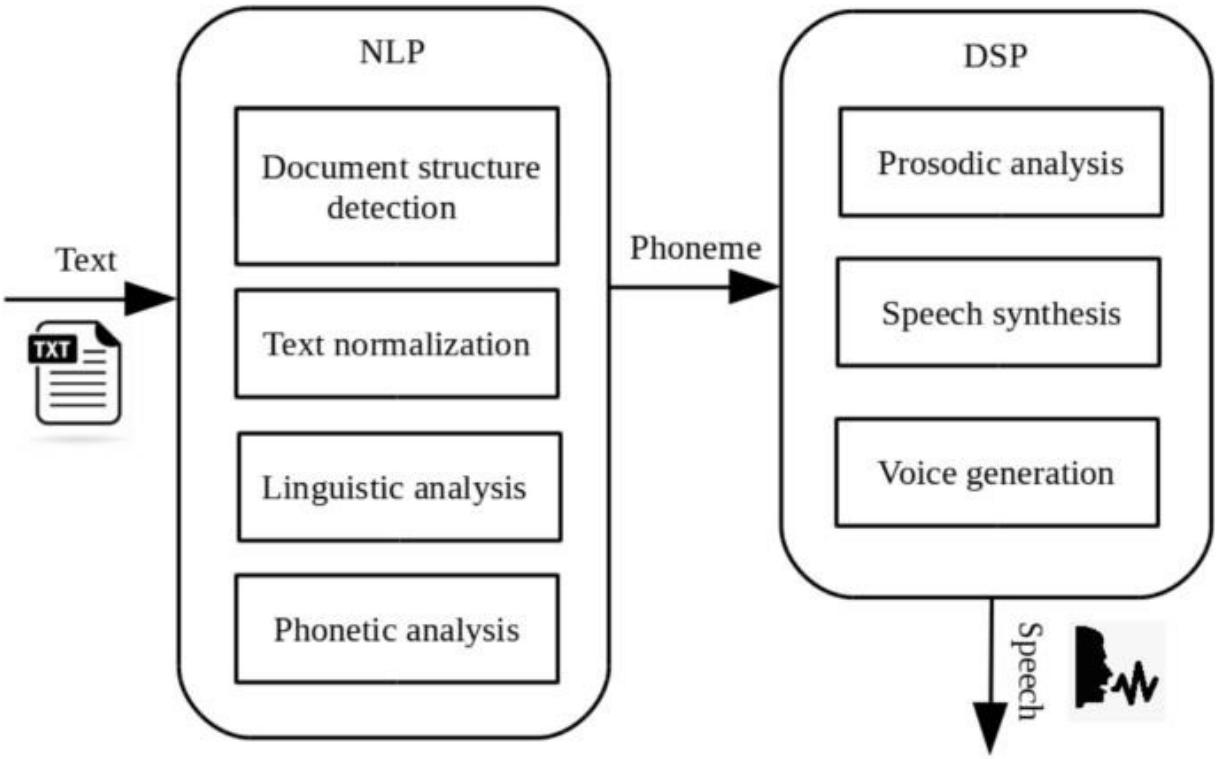


Figure 3.27 Chart TTS conversion architecture.[12]

Chapter 4
Voice Recognition

4.1 Voice To Text Project:

Aim: we want to help the “Deaf People”, they Can’t hear the Human Voice which is our Problem so in this project we trying to help them & and make them understand the people and to make them Communicate with us and other people

Idea: so Simply the Idea Using a Microphone, Raspberry Bi, Google Speech Recognition API & OLED Screen .. we Convert the Human Speech “which they Can’t hear it” to Text in front their Glasses “which’s they can’t Read it” and we hoply that help them .

Voice To Text Project Implementation:

At first the Human Speech Recorded by the Mic “which is Connected with Raspberry bi” and we Record it , And the next step: we upload this record Automatically to Google Speech recognition to Covert it to Text .

The next step the Text will get out on the OLED Screen “which’s Connected with Raspberry bi too”

Lastly “in the Next Semester” We Reflect The Text which is on the Oled, using mirrors to make it appear in front of the glasses of the deaf Person .

So that’s the main idea .. so we now go Deeply to know how it works with Details .

4.2 Speech Recognition:

Speech recognition, or speech-to-text, is the ability of a machine or program to identify words spoken aloud and convert them into readable text. Rudimentary speech recognition software has a limited vocabulary and may only identify words and phrases when spoken clearly. More sophisticated software can handle natural speech, different accents and various languages.

Speech recognition uses a broad array of research in computer science, linguistics and computer engineering. Many modern devices and text-focused programs have speech recognition functions in them to allow for easier or hands-free use of a device. Speech recognition and voice recognition are two different technologies and should not be confused:

- **Speech recognition** is used to identify words in spoken language.
- **Voice recognition** is a biometric technology for identifying an individual's voice.

4.2.1 Speech Recognition Concept

Speech recognition systems use computer algorithms to process and interpret spoken words and convert them into text. A software program turns the sound a microphone records into written language that computers and humans can understand, following these four steps:

1. analyze the audio;
2. break it into parts;
3. digitize it into a computer-readable format; and
4. use an algorithm to match it to the most suitable text representation.

Speech recognition software must adapt to the highly variable and context-specific nature of human speech. The software algorithms that process and organize audio into text are trained on different speech patterns, speaking styles, languages, dialects,

accents and phrasings. The software also separates spoken audio from background noise that often accompanies the signal.

To meet these requirements, speech recognition systems use two types of models:

- **Acoustic models.** These represent the relationship between linguistic units of speech and audio signals.
- **Language models.** Here, sounds are matched with word sequences to distinguish between words that sound similar.

4.2.2 Speech Recognition Applications

Speech recognition systems have quite a few applications. Here is a sampling of them.

Mobile devices. Smartphones use voice commands for call routing, speech-to-text processing, voice dialing and voice search. Users can respond to a text without looking at their devices. On Apple iPhones, speech recognition powers the keyboard and Siri, the virtual assistant. Functionality is available in secondary languages, too. Speech recognition can also be found in word processing applications like Microsoft Word, where users can dictate words to be turned into text.

Education. Speech recognition software is used in language instruction. The software hears the user's speech and offers help with pronunciation.

Customer service. Automated voice assistants listen to customer queries and provides helpful resources.

Healthcare applications. Doctors can use speech recognition software to transcribe notes in real time into healthcare records.

Disability assistance. Speech recognition software can translate spoken words into text using closed captions to enable a person with hearing loss to understand what others are saying. Speech recognition can also enable those with limited use of their hands to work with computers, using voice commands instead of typing.

Court reporting. Software can be used to transcribe courtroom proceedings, precluding the need for human transcribers.

Emotion recognition. This technology can analyze certain vocal characteristics to determine what emotion the speaker is feeling. Paired with sentiment analysis, this can reveal how someone feels about a product or service.

Hands-free communication. Drivers use voice control for hands-free communication, controlling phones, radios and global positioning systems, for instance.

The Features of speech recognition systems

Good speech recognition programs let users customize them to their needs. The features that enable this include:

- **Language weighting.** This feature tells the algorithm to give special attention to certain words, such as those spoken frequently or that are unique to the conversation or subject. For example, the software can be trained to listen for specific product references.
- **Acoustic training.** The software tunes out ambient noise that pollutes spoken audio. Software programs with acoustic training can distinguish speaking style, pace and volume amid the din of many people speaking in an office.
- **Speaker labeling.** This capability enables a program to label individual participants and identify their specific contributions to a conversation.
- **Profanity filtering.** Here, the software filters out undesirable words and language.

4.2.3 Types of Speech Recognition Algorithms

The power behind speech recognition features comes from a set of algorithms and technologies. They include the following:

- **Hidden Markov model.** HMMs are used in autonomous systems where a state is partially observable or when all of the information necessary to make a decision is not immediately available to the sensor (in speech recognition's case, a microphone). An example of this is in acoustic modeling, where a program must match linguistic units to audio signals using statistical probability.
- **Natural language processing.** NLP eases and accelerates the speech recognition process.
- **N-grams.** This simple approach to language models creates a probability distribution for a sequence. An example would be an algorithm that looks at the last few words spoken, approximates the history of the sample of speech and uses that to determine the probability of the next word or phrase that will be spoken.
- **Artificial intelligence.** AI and machine learning methods like deep learning and neural networks are common in advanced speech recognition software. These systems use grammar, structure, syntax and composition of audio and voice signals to process speech. Machine learning systems gain knowledge with each use, making them well suited for nuances like accents.

4.2.4 Advantages of Speech Recognition

There are several advantages to using speech recognition software, including the following:

- **Machine-to-human communication.** The technology enables electronic devices to communicate with humans in natural language or conversational speech.
- **Readily accessible.** This software is frequently installed in computers and mobile devices, making it accessible.
- **Easy to use.** Well-designed software is straightforward to operate and often runs in the background.
- **Continuous, automatic improvement.** Speech recognition systems that incorporate AI become more effective and easier to use over time. As systems complete speech recognition tasks, they generate more data about human speech and get better at what they do.

4.2.5 Disadvantages of Speech Recognition

While convenient, speech recognition technology still has a few issues to work through. Limitations include:

- **Inconsistent performance.** The systems may be unable to capture words accurately because of variations in pronunciation, lack of support for some languages and inability to sort through background noise. Ambient noise can be especially challenging. Acoustic training can help filter it out, but these programs aren't perfect. Sometimes it's impossible to isolate the human voice.
- **Speed.** Some speech recognition programs take time to deploy and master. The speech processing may feel relatively slow**.**
- **Source file issues.** Speech recognition success depends on the recording equipment used, not just the software.

Speech recognition is an evolving technology. It is one of the many ways people can communicate with computers with little or no typing. A variety of communications-based business applications capitalize on the convenience and speed of spoken communication that this technology enables.

Speech recognition programs have advanced greatly over 60 years of development. They are still improving, fueled in particular by AI.

Learn more about the AI-powered business transcription software in this Q&A with Wilfried Schaffner, chief technology officer of Speech Processing Solutions.

And today we've many "API" or Apps can Perform Speech Recognition like:

- Google Speech API
- IBM Watson API
- SpeechAPI
- Speech to Text API
- Text-to-Speech API
- Rev.AI API
- ReadSpeaker API
- Speech2Topics API
- Siri API
- Wit API

4.3 Introduction to application programming interface (API)

An application programming interface, or API, enables companies to open up their applications' data and functionality to external third-party developers, business partners, and internal departments within companies. This allows services and products to communicate with each other and leverage each other's data and functionality through a documented

interface. Developers don't need to know how an API is implemented; they simply use the interface to communicate with other products and services. API use has surged over the past decade, to the degree that many of the most popular web applications today would not be possible without APIs.

4.3.1 API Concept

An API is a set of defined rules that explain how computers or applications communicate with one another. APIs sit between an application and the web server, acting as an intermediary layer that processes data transfer between systems.

Here's how an API works:

1. **A client application initiates an API call** to retrieve information—also known as a *request*. This request is processed from an application to the web server via the API's Uniform Resource Identifier (URI) and includes a request verb, headers, and sometimes, a request body.
2. **After receiving a valid request**, the API makes a call to the external program or web server.
3. **The server sends a *response*** to the API with the requested information.
4. **The API transfers the data** to the initial requesting application.

While the data transfer will differ depending on the web service being used, this process of requests and response all happens through an API. Whereas a user interface is designed for use by humans, APIs are designed for use by a computer or application.

APIs offer security by design because their position as middleman facilitates the abstraction of functionality between two systems—the API endpoint decouples the consuming application from the infrastructure providing the service. API calls usually include authorization credentials to

reduce the risk of attacks on the server, and an API gateway can limit access to minimize security threats. Also, during the exchange, HTTP headers, cookies, or query string parameters provide additional security layers to the data.

For example, consider an API offered by a payment processing service. Customers can enter their card details on the frontend of an application for an ecommerce store. The payment processor doesn't require access to the user's bank account; the API creates a unique token for this transaction and includes it in the API call to the server. This ensures a higher level of security against potential hacking threats.

4.3.2 Importance of APIs

Whether you're managing existing tools or designing new ones, you can use an application programming interface to simplify the process. Some of the main benefits of APIs include the following:

- **Improved collaboration:** The average enterprise uses almost 1,200 cloud applications (link resides outside of IBM), many of which are disconnected. APIs enable integration so that these platforms and apps can seamlessly communicate with one another. Through this integration, companies can automate workflows and improve workplace collaboration. Without APIs, many enterprises would lack connectivity and would suffer from informational silos that compromise productivity and performance.
- **Easier innovation:** APIs offer flexibility, allowing companies to make connections with new business partners, offer new services to their existing market, and, ultimately, access new markets that can generate massive returns and drive digital transformation. For example, the company Stripe began as an API with just seven lines of code. The company has since

partnered with many of the biggest enterprises in the world, diversified to offer loans and corporate cards, and was recently valued at USD 36 billion (link resides outside of IBM).

- **Data monetization:** Many companies choose to offer APIs for free, at least initially, so that they can build an audience of developers around their brand and forge relationships with potential business partners. However, if the API grants access to valuable digital assets, you can monetize it by selling access (this is referred to as the API economy). When AccuWeather (link resides outside of IBM) launched its self-service developer portal to sell a wide range of API packages, it took just 10 months to attract 24,000 developers, selling 11,000 API keys and building a thriving community in the process.
- **Added security:** As noted above, APIs create an added layer of protection between your data and a server. Developers can further strengthen API security by using tokens, signatures, and Transport Layer Security (TLS) encryption; by implementing API gateways to manage and authenticate traffic; and by practicing effective API management.

Common APIs

Because APIs allow companies to open up access to their resources while maintaining security and control, they have become a valuable aspect of modern business. Here are some popular examples of application programming interfaces you may encounter:

- **Universal logins:** A popular API example is the function that enables people to log in to websites by using their Facebook, Twitter, or Google profile login details. This convenient feature allows any website to leverage an API from one of the more popular services to quickly

authenticate the user, saving them the time and hassle of setting up a new profile for every website service or new membership.

- **Third-party payment processing:** For example, the now-ubiquitous "Pay with PayPal" function you see on ecommerce websites works through an API. This allows people to pay for products online without exposing any sensitive data or granting access to unauthorized individuals.
- **Travel booking comparisons:** Travel booking sites aggregate thousands of flights, showcasing the cheapest options for every date and destination. This service is made possible through APIs that provide application users with access to the latest information about availability from hotels and airlines. With an autonomous exchange of data and requests, APIs dramatically reduce the time and effort involved in checking for available flights or accommodation.
- **Google Maps:** One of the most common examples of a good API is the Google Maps service. In addition to the core APIs that display static or interactive maps, the app utilizes other APIs and features to provide users with directions or points of interest. Through geolocation and multiple data layers, you can communicate with the Maps API when plotting travel routes or tracking items on the move, such as a delivery vehicle.

- **Twitter:** Each Tweet contains descriptive core attributes, including an author, a unique ID, a message, a timestamp when it was posted, and geolocation metadata. Twitter makes public Tweets and replies available to developers and allows developers to post Tweets via the company's API.

4.3.3 Types of APIs

Nowadays, most application programming interfaces are web APIs that expose an application's data and functionality over the internet. Here are the four main types of web API:

- **Open APIs** are open source application programming interfaces you can access with the HTTP protocol. Also known as public APIs, they have defined API endpoints and request and response formats.
- **Partner APIs** are application programming interfaces exposed to or by strategic business partners. Typically, developers can access these APIs in self-service mode through a public API developer portal. Still, they will need to complete an onboarding process and get login credentials to access partner APIs.
- **Internal APIs** are application programming interfaces that remain hidden from external users. These private APIs aren't available for users outside of the company and are instead intended to improve productivity and communication across different internal development teams.
- **Composite APIs** combine multiple data or service APIs. These services allow developers to access several endpoints in a single call. Composite APIs are useful in microservices architecture where performing a single task may require information from several sources.

4.3.4 Types of API protocols

As the use of web APIs has increased, certain protocols have been developed to provide users with a set of defined rules that specifies the accepted data types and commands. In effect, these API protocols facilitate standardized information exchange:

- **SOAP** (Simple Object Access Protocol) is an API protocol built with XML, enabling users to send and receive data through SMTP and HTTP. With SOAP APIs, it is easier to share information between apps or software components that are running in different environments or written in different languages.
- **XML-RPC** is a protocol that relies on a specific format of XML to transfer data, whereas SOAP uses a proprietary XML format. XML-RPC is older than SOAP, but much simpler, and relatively lightweight in that it uses minimum bandwidth.
- **JSON-RPC** is a protocol similar to XML-RPC, as they are both remote procedure calls (RPCs), but this one uses JSON instead of XML format to transfer data. Both protocols are simple. While calls may contain multiple parameters, they only expect one result.
- **REST** (Representational State Transfer) is a set of web API architecture principles, which means there are no official standards (unlike those with a protocol). To be a REST API (also known as a RESTful API), the interface must adhere to certain architectural constraints. It's possible to build RESTful APIs with SOAP protocols, but the two standards are usually viewed as competing specifications.

So we now want to know how the output will get out Simply we connect the "OLED-Screen" with Raspberry bi to Show the Text on the Screen .

4.3.5 The Code:

```
        with sr.Microphone() as s:
            r.adjust_for_ambient_noise(s, duration=4)

            v = r.listen(s)
with sr.Microphone() as s:
    print("Say something!")
    audio = r.listen(s,timeout=4,phrase_time_limit=4)


            try:
                text = r.recognize_google(v)
                text = text.lower()

            except sr.RequestError or sr.UnknownValueError:
                text = r.recognize_sphinx(v)
                text = text.lower()

            print(f"----------\n{text}\n----------")

            offset = 0
            s = ""
            for w in text.split(" "):
                font_width, font_height = font.getsize(s + " " + w)

                if font_width < 128:
                    s += " " + w

                else:
                    font_width, font_height = font.getsize(s)
                    draw.text((width/2 - font_width/2, offset), s, fill="white", font=font)
                    offset += 20
                    s = w
```

Chapter 5
Conclusion and Future Work

Chapter 5 Conclusion and Future Recommendation

6.1 Conclusions

Technology has played a critical part in our lives. We use it virtually all of the time and almost everywhere. The distinct and quick development that we observe every day proves to us that it is pointless to give up and struggle with our life's challenges. Technology provides us with numerous substantial solutions to our issues as well as numerous drawbacks. Our responsibility is to make proper use of it in order to achieve a degree of achievement that benefits individuals, society, and the entire country.

Our project was a success since we were able to achieve my aim of creating a set of smart glasses for Deaf or hearing impaired persons that can translate sounds around them, such as the voice of the person they're speaking to, into the glasses' lens.Despite this, there are a few things that could be done to better the gadget.

6.2 Future Recommendation:

While the team members were working on the implementation, they thought of many ideas and improvements for the “Smart Glasses”. However, they wished they have more time and knowledge to do them. “Smart Glasses” can be improved in the future for blind people and people who have vision difficulties by adding new techniques. For instance, direction and warning messages to prevent expected accidents, messages to tell the user about the battery level, video detection to provide a full healthy life for people with vision difficulties, develop mobile application to control “Smart Glasses”, use 270 camera to have more wider view angle. provide the glasses with GPS notification and develop the glasses’ design to have little, small and light components so the user can wear it easily.

References

[1] Wang, T., Wu, D. J., Coates, A., & Ng, A. Y. (2012, November). End-to-end text recognition with convolutional neural networks. In Pattern Recognition (ICPR), 2012 21st International Conference on (pp. 3304-3308). IEEE.

[2] Koo, H. I., & Kim, D. H. (2013). Scene text detection via connected component clustering and nontext filtering. IEEE transactions on image processing, 22(6), 2296-2305.

[3] Bissacco, A., Cummins, M., Netzer, Y., & Neven, H. (2013). Photoocr: Reading text in uncontrolled conditions. In Proceedings of the IEEE International Conference on Computer Vision (pp. 785-792).

[4] Yin, X. C., Yin, X., Huang, K., & Hao, H. W. (2014). Robust text detection in natural scene images. IEEE transactions on pattern analysis and machine intelligence, 36(5), 970-983.

[5] Neumann, L., & Matas, J. (2012, June). Real-time scene text localization and recognition.In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on (pp. 3538-3545). IEEE.

[6] Neumann, L., & Matas, J. (2013). Scene text localization and recognition with oriented stroke detection. In Proceedings of the IEEE International Conference on Computer Vision (pp. 97-104).

[7] Zhou, X., Ylao, C., Wen, H., Wang, Y., Zhou, S., He, W., & Liang, J. (2017). EAST: an efficient and accurate scene text detector. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 5551-5560).

[8] Mitrasinovic, S.; Camacho, E.; Trivedi, N.; Logan, J.; Campbell, C.; Zilinyi, R.; Lieber, B.; Bruce, E.; Taylor, B.; Martineau, D.

[9] Clinical and Surgical Applications of Smart Glasses. Technol. Health Care 2015, 23, 381–401. [CrossRef]

[10] Klinker, K.; Berkemeier, L.; Zobel, B.; Wüller, H.; Huck, V.; Wiesche, M.; Remmers, H.; Thomas, O.; Krcmar, H. Structure for Inno-
vations: A Use Case Taxonomy for Smart Glasses in Service Processes. In Proceedings of the Multikonferenz Wirtschaftsinformatik
2018, Lüneburg, Germany, 6–9 March 2018; pp. 1599–1610.

References

[11] Hofmann, B.; Haustein, D.; Landeweerd, L. Smart-Glasses: Exposing and Elucidating the Ethical Issues. Sci. Eng. Ethics 2017, 23, 701–721. [CrossRef] [12] Lee, L.-H.; Hui, P. Interaction Methods for Smart Glasses: A Survey. IEEE Access 2018, 6, 28712–28732. [CrossRef] Klinker, K.; Obermaier, J.; Wiesche, M. Conceptualizing passive trust: The case of smart glasses in healthcare. In Proceedings of the 27th European Conference on Information Systems (ECIS), Stockholm & Uppsala, Sweden, 8–14 June 2019; p. 12

[13] https://www.raspberrypi.org/products/camera-module-v2/

[14]https://sisu.ut.ee/imageprocessing/book/1#:~:text=1.-,Introduction%20to%20image%20processing,some%20useful%20information%20from%20it.&text=Output%20in%20which%20result%20can,is%20based%20on%20image%20analysis

[15] https://opencv.org/about/

[16] https://www.edureka.co/blog/tensorflow-object-detection-tutorial/#object

[17] OpenCV Thresholding (cv2.threshold) - PyImageSearch

[18] Detecting low contrast images with OpenCV, scikit-image, and Python - PyImageSearch

[19] OpenCV Morphological Operations - PyImageSearch

[20] OpenCV: Contours : Getting Started

[12] Tesseract (software) - Wikipedia

[21] OpenCV OCR and text recognition with Tesseract - PyImageSearch

[22] Long short-term memory - Wikipedia

[23] https://medrectech.com/blog/Text-to-speech

[24]https://slidetodoc.com/presentation_image_h/f3c8f07ae0a36e5e2bd13b2ec8f0e408/image-5.jpg

References

[25] van Santen, Jan P. H.; Sproat, Richard W.; Olive, Joseph P.; Hirschberg, Julia (1997). Progress in Speech Synthesis.
Springer. ISBN 978-0-387-94701-3.

[26] ^ Van Santen, J. (April 1994). "Assignment of segmental duration in text-to-speech synthesis". Computer Speech & Language. 8 (2): 95–128. doi:10.1006/csla.1994.1005

[27] https://www.semanticscholar.org/paper/A-Comparative-Study-of-Arabic-Text-to-Speech-Bakhsh-Alshomrani/fcc9bf2f046b4d28154c2869810d8bb11aa52513

[28] Allen, Jonathan; Hunnicutt, M. Sharon; Klatt, Dennis (1987). From Text to Speech: The MITalk system. Cambridge University Press. ISBN 978-0-521-30641-6.

[29] Pisoni, D. B. et al., “Perception of synthetic speech generated by rule,” in Proceedings of the IEEE, 1985, pp. 1665–1676.

[30] Sondhi, M. M., and Schroeter, J., Speech Production Models and Their Digital Implementations, in: The Digital Signal Processing Handbook, V. K. Madisetti, D. B. Williams (Eds.), CRC Press, Boca Raton, Florida, pp. 44-1 to 44-21, 1997.

[31] Kuligowska, K, Kisielewicz, P. and Wlodarz, A. (2018) Speech synthesis systems: disadvantages and limitations, International Journal of Engineering & Technology, [S.l.], v. 7, n. 2.28, p. 234–239.

[32]https://www.researchgate.net/figure/Block-diagram-ofText-to-Speech-System-Techniques-of-speech-synthesis-5a_fig3_304601298

[33] https://acoustics.org/pressroom/httpdocs/148th/schroeter.ht ml

[34] https://en.wikipedia.org/wiki/Speech_synthesis