A Mini Project Report

on

“SCRIPT VOICE”

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

by

K. Deekshitha	**18WH1A0401**
S. Tejaswini Harshitha	**18WH1A0402**
K. Saranya	**18WH1A0424**
S. Harika	**18WH1A0442**

under the guidance of

Dr. J. Naga Vishnu Vardhan

Professor, ECE.

Department of Electronics and Communication Engineering

BVRIT HYDERABAD College of Engineering for Women

(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with A Grade

Bachupally, Hyderabad – 500090

2021-22

DECLARATION

We hereby declare that the work described in this report, entitled **"SCRIPT VOICE"** which is being submitted by us in partial fulfillment for the award of the degree of **Bachelor of Technology** in the department of **Electronics and Communication Engineering** at **BVRIT HYDERABAD College of Engineering for Women,** affiliated to **Jawaharlal Nehru Technological University Hyderabad**, Kukatpally, Hyderabad – 500085 is the result of original work carried out by us under the guidance of **Dr. J. Naga Vishnu Vardhan, Professor, ECE.**

This work has not been submitted for any Degree/Diploma of this or any other institute/university to the best of our knowledge and belief.

Place: Hyderabad

Date:

K. Deekshitha

S. Tejaswini Harshitha

S. Harika

K. Saranya

Department of Electronics and Communication Engineering
BVRIT HYDERABAD College of Engineering for Women
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with A Grade
Bachupally, Hyderabad – 500090

Certificate

This is to certify that the mini project report, entitled **" SCRIPT VOICE "** is a record of bonafide work carried out by **Ms. K. Deekshitha (18W1HA0401), Ms. S. Tejaswini Harshitha (18WH1A0402), Ms. K. Saranya (18WH1A0424), Ms. S. Harika (18WH1A0442),** in partial fulfillment for the award of the degree of **Bachelor of Technology** in the department of **Electronics and Communication Engineering** at **BVRIT HYDERABAD College of Engineering for Women,** affiliated to **Jawaharlal Nehru Technological University Hyderabad**, Kukatpally, Hyderabad – 500085.

Supervisor
Dr. J Naga Vishnu Vardhan
Professor, ECE

Head of the Department
Dr. Anwar Bhasha Pattan
Professor & HoD-ECE

External Examiner

ACKNOWLEDGMENT

The satisfaction that accompanies in successful completion of the task would be incomplete without the mention of the people who made it possible.

We wish to express our deep sense of gratitude to our guide **Dr. J. Naga Vishnu Vardhan, Professor**, Department of Electronics and Communication Engineering, BVRIT HYDERABAD College of Engineering for Women, for his able guidance and suggestions, which helped us in completing this project work on time.

We would like to thank **Dr. Anwar Bhasha Pattan**, **Professor and Head,** Department of Electronics and Communication Engineering for his guidance, support and encouragement.

We express our gratitude towards our honorable Principal, **Dr. K V N Sunitha** and the **Management** for providing all the facilities.

We also thank all the **Faculty Members** and **Non-teaching staff members** of Electronics and Communication Engineering department, who supported us directly or indirectly in successful completion of this project work.

Finally, we thank all our **friends** and **family members** for their continuous support and help.

K. Deekshitha

S. Tejaswini Harshitha

K. Saranya

S. Harika

ABSTRACT

According to estimates from WHO about 285 million people are visually impaired worldwide, out of them 39 million are blind and 246 million have low vision. Visually impaired people face difficulty in reading. Although braille is the only way to overcome the challenge of reading, not necessarily each visually impaired individual knows it. So, we came up with a solution named as **SCRIPT VOICE**, where both blind and low vision people can easily know the content in the paper.

In this we translate the typed text which is written in English language to speech. These kind of ideas consider a solution to motivate blind students to complete their education despite all their difficulties. Its main objective is to develop an easy way of reading text for blind people. In this process, first it has to scan a text image and then convert it into audio text, so that the person will listen to the audio through the speaker / headphones connected to the device.

CONTENTS

LIST OF FIGURES AND TABLES

1. Introduction

There are many existing solutions to the problem of assisting individuals who are blind to read, however, they have certain limitations. We focus on improving the competence of blind people by providing them with a solution where the details are given in the form of audio signal.

Script Voice is an automatic document reader for visually impaired people. The proposed project uses a camera-based assistive device which can be used by individuals to read printed text. The scheme is to implement an embedded system based image capturing technique using Raspberry Pi board. The design is inspired by prior research with visually impaired people, and it is small, that helps in achieving result in little setup. Here, we have put forward a text read out system for visually impaired people. Tesseract module and Text-to-Speech synthesis is used to convert images into audio output (Speech).

The proposed apparatus has a camera which act as the input device for digitization and this digitized script is processed by Tesseract (software module). A procedure is followed for recognition of characters and the line of reading. In the context of software development, the OpenCV (Open source Computer Vision) libraries are employed to capture image of text and character recognition. The final identified text document is given to the output devices based on the choice of the user. Headset connected to the Raspberry Pi or a speaker act as the output device.

2. Block Diagram

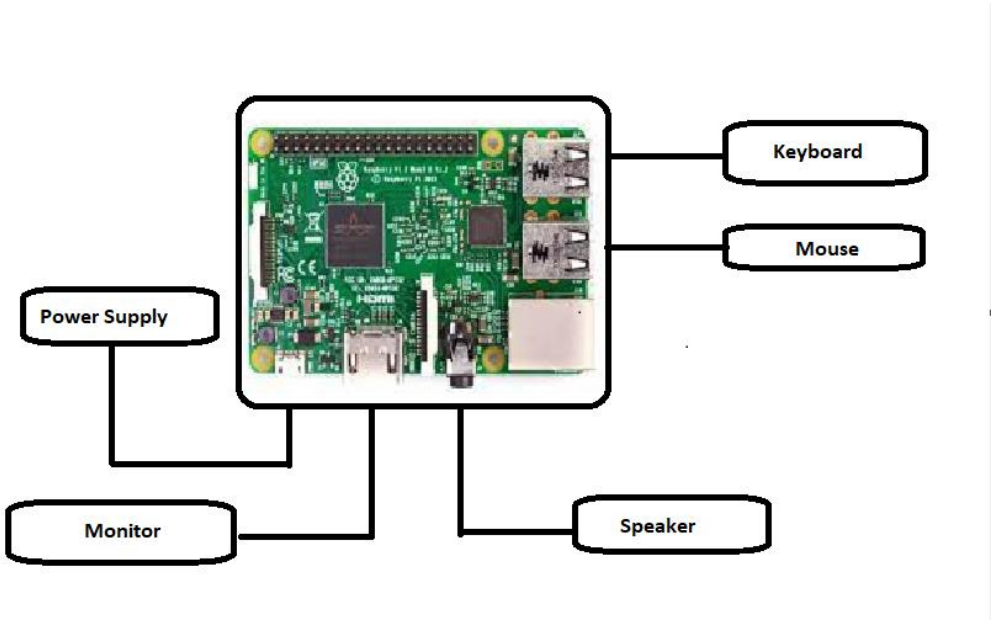

Fig. 1: Block Diagram

Block Diagram description

The above figure depicts the block diagram of our proposed system. The framework used for the proposed system is the Raspberry Pi 3 model B board. The Raspberry Pi board consists of 4 USB ports, 40 GPIO pins for input / output, CSI camera interface, HDMI port, SD card slot and an audio jack. Raspberry pi board is to be powered with a 5V power supply to its micro USB connector through the Switched Mode Power Supply (SMPS). The SMPS converts the 230v AC supply to 5v DC. Pi camera is connected to the CSI camera interface. Screen, mouse and keyboard are connected to the Raspberry Pi module through USB ports available. Speakers or Headphones are connected to the audio jack available on the raspberry pi board.

3. Components List

S.No	Components	Quantity
1.	Raspberry Pi 3 model B	1
2.	Pi camera	1
3.	Speaker/Headset	1
4.	Cables & Connectors	As required

4. Components Description

Raspberry Pi 3 model B

The Raspberry Pi 3 Model B is the third generation Raspberry Pi. This powerful credit-card sized single board computer can be used for many applications and supersedes the original Raspberry Pi Model B+ and Raspberry Pi 2 Model B. Whilst maintaining the popular board format the Raspberry Pi 3 Model B brings you a more powerful processer, 10x faster than the first-generation Raspberry Pi. Additionally, it adds wireless LAN & Bluetooth connectivity making it the ideal solution for powerful connected designs.

The Raspberry Pi 3 Model B is a tiny credit card size computer. Just add a keyboard, mouse, display, power supply, micro SD card with installed Linux Distribution and one will have a fully fledged computer that can run applications from word processors and spreadsheets to games.

As the Raspberry Pi 3 supports HD video, one can even create a media center with it. The Raspberry Pi 3 Model B is the first Raspberry Pi to be open-source.

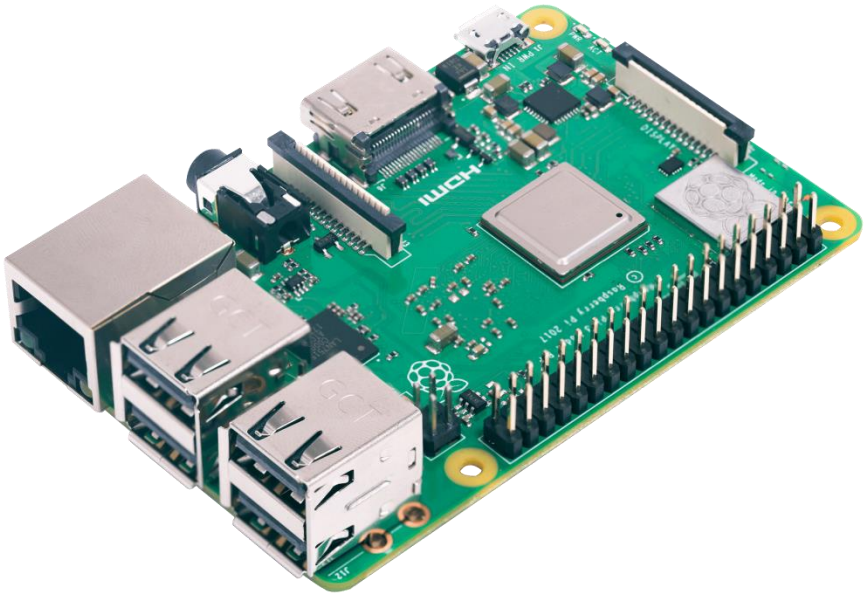

Fig.2: Raspberry Pi 3 model B

Specifications

SoC: Broadcom BCM2837

CPU: 4× ARM Cortex-A53, 1.2GHz

GPU: Broadcom VideoCore IV

RAM: 1GB LPDDR2 (900 MHz)

Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless

Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy

Storage: microSD

GPIO: 40-pin header, populated

Ports: HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)

Raspberry Pi GPIOs

The Raspberry Pi offers up its GPIO over a standard male header on the board. Over the years the header has expanded from 26 pins to 40 pins while maintaining the original pinout.

A GPIO pin can be used as either a digital input or a digital output, and both operate at 3.3V. Unlike the Arduino, the Raspberry Pi does not have any analog inputs. For that you must use an external analog-to-digital converter (ADC) or connect the Pi to an interface board or to an Arduino.

Raspberry Pi pin numbering

There are (at least) two, different numbering schemes you may encounter when referencing Pi pin numbers: (1) Broadcom chip-specific pin numbers and (2) P1 physical pin numbers. We are usually free to use either number-system, but many programs require declaration of scheme, which we are using, at the very beginning of the program.

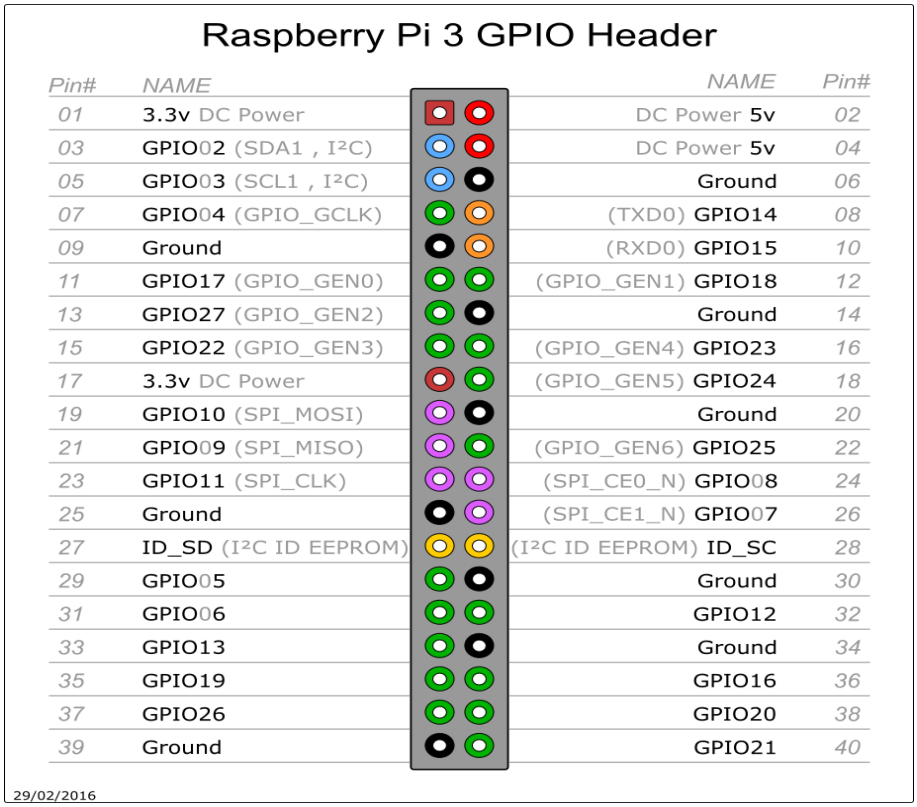

Fig.3: Raspberry Pi 3 GPIO Header

Pi camera

Pi Camera module is a camera which can be used to take pictures and high-definition video. This high-definition camera module compatible with all Raspberry Pi models. Provides high sensitivity, low crosstalk and low noise image capture in an ultra-small and lightweight design.

The camera module connects to the Raspberry Pi board via the CSI (Camera Serial Interface) connector designed specifically for interfacing to cameras. This connector is 15-pin ribbon cable used to connect Pi Camera module to Raspberry Pi's CSI port. The CSI bus is capable of extremely high data rates, and it exclusively carries pixel data to the processor.

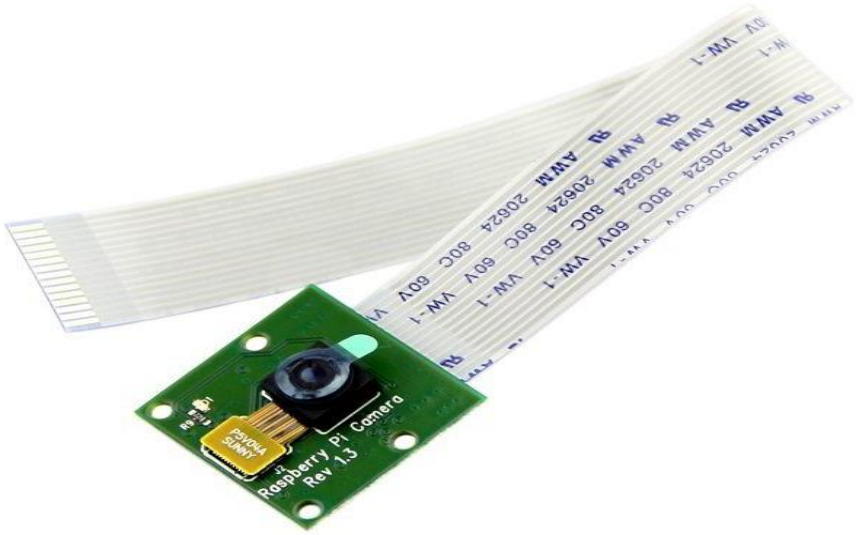

Fig.4: Pi camera

Specifications of Pi camera

Here, Pi camera v1.3 is used. Its specifications are listed below,

- Resolution – 5 MP
- HD Video recording – 1080p @30fps, 720p @60fps, 960p @45fps
- It Can capture wide, still (motionless) images of resolution 2592x1944 pixels
- Size – 25mm x 23 mm x 8mm
- Weight – 3 grams

CSI Interface enabled.

Speakers

Speakers are used to hear the sound converted from text by Raspberry Pi module program.

Fig.5: Speakers

5. Software requirements

5.1 picamera

"picamera" is a package that provides a pure Python interface to the Raspberry Pi camera module for Python 2.7 (or above) or Python 3.2 (or above).

Installation of this package

If the operating system being used is Raspbian distro, it will probably have picamera installed by default. It can be find out simply by starting Python and trying to import picamera by using following commands:

```
$ python -c "import picamera"      #For python 2.7
$ python3 -c "import picamera"     #For python 3
```

Otherwise, to install picamera on Raspbian, it is best to use the system's package manager: apt. This will ensure that all the required packages foe the working of camera are available. Also it makes sure that picamera is available for all users on the system. To install picamera using apt simply run the following commands:

```
$ sudo apt-get update
$ sudo apt-get install python-picamera python3-picamera
```

After running these commands, picamera package will be successfully installed with all the functions or classes in it available to use.

Although picamera library contains numerous classes, the primary one that all users are likely to interact with is PiCamera. This provides a pure Python interface to the Raspberry Pi's camera module. Upon construction, this class initializes the camera. Preview or recording is not started automatically upon construction. There are different functions available to record videos or capture pictures. The **capture()** method is available to capture images, the **start_recording()** method is to begin recording video, or the **start_preview()** method is used to start live display of the camera's input.

Various attributes are provided to adjust the camera's configuration. There are some attributes that can be adjusted while a recording is running, like **brightness**. Other attributes like **resolution**, can only be adjusted when the camera is idle.

5.2 cv2

OpenCV is a tool for processing images and performing computer vision tasks. It is an open-source library that can be used to perform various tasks like face detection, objection tracking, landmark detection, and many more. It supports multiple languages including python, java C++. Although, in this project, we will be using python only.

This CV2 library is equipped with several useful functions and algorithms, which are all freely available to us. Few of these functions are so common and are used in almost every computer vision task. Whereas many of the functions are still unexplored and haven't received much attention yet.

Features of OpenCV library

Using OpenCV library, one can

- Read and write images
- Capture and save videos
- Process images (filter, transform)
- Perform feature detection
- Detect specific objects such as faces, eyes, cars, in the videos or images.
- Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it.

imread()

The Imgcodecs class of the package org.opencv.imgcodecs contains methods to read and write images. Using OpenCV, an image can be read and can also be stored in a matrix so that all the required transformations can be performed on the matrix if needed. Later, processed matrix can be written into a file.

The read() method of the Imgcodecs class is used in reading an image using OpenCV.

Following is the syntax of this method.

imread(filename)

It accepts an argument **(filename)**, a variable of the type string representing the path of the file that is to be read.

threshold()

During processing of an image, For every pixel, the same threshold value is applied. If the pixel value is smaller than the threshold, it is set to 0, otherwise it is set to a maximum value. The function cv.threshold is used to apply the thresholding. The first argument is the source image, which should be a grayscale image. The second argument is the threshold value which is used to classify the pixel values. The third argument is the maximum value which is assigned to pixel values exceeding the threshold. OpenCV provides different types of thresholding which is given by the fourth parameter of the function. Basic thresholding as described above is done by using the type cv.THRESH_BINARY. All simple thresholding types are:

- cv.THRESH_BINARY
- cv.THRESH_BINARY_INV
- cv.THRESH_TRUNC
- cv.THRESH_TOZERO
- cv.THRESH_TOZERO_INV

```
ret,thresh1 = cv.threshold(img,127,255,cv.THRESH_BINARY)

ret,thresh2 = cv.threshold(img,127,255,cv.THRESH_BINARY_INV)

ret,thresh3 = cv.threshold(img,127,255,cv.THRESH_TRUNC)

ret,thresh4 = cv.threshold(img,127,255,cv.THRESH_TOZERO)

ret,thresh5 = cv.threshold(img,127,255,cv.THRESH_TOZERO_INV)
```

adaptiveThreshold()

In the above case, we used one global value as a threshold. But this might not be good in all cases, e.g. if an image has different lighting conditions in different areas. In that case, adaptive thresholding can help. Here, the algorithm determines the threshold for a pixel based on a small region around it. So we get different thresholds for different regions of the same image which gives better results for images with varying illumination.

In addition to parameters described above the method **cv.adaptiveThreshold** takes three input parameters:
The **adaptiveMethod** decides how the threshold value is calculated:

- **cv.ADAPTIVE_THRESH_MEAN_C**: The threshold value is the mean of the neighbourhood area minus the constant **C**.
- **cv.ADAPTIVE_THRESH_GAUSSIAN_C**: The threshold value is a gaussian-weighted sum of the neighbourhood values minus the constant **C**.

The **blockSize** determines the size of the neighbourhood area and **C** is a constant that is subtracted from the mean or weighted sum of the neighbourhood pixels.

```
th2 = cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_MEAN_C,\
        cv.THRESH_BINARY,11,2)
th3 = cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_GAUSSIAN_C,\
        cv.THRESH_BINARY,11,2)
```

cvtColor()

The colors present in a given image is represented by color spaces in OpenCV and there are several color spaces each having its own importance and some of the color spaces are RGB, CMYK, etc. and whenever there is a need to convert a given image from one color space to another color space in OpenCV, then we make use of a function called cvtColor() function and there are more than 150 color space conversions codes available in OpenCV, some of them are cv2.COLOR_BGR2GRAY, cv2.COLOR_BGR2HSV etc. and color space conversion

using cvtColor() function is very useful to solve problems in the area of computer vision. Here, we are converting the colour of the image captured to binary image.

The syntax to define cvtColor() function in OpenCV is as follows:

cvtColor(image, code)

where the image is the image whose color space is to be converted, code is the color space conversion code.

copyMakeBorder()

Various borders to an image can be made using the method **copyMakeBorder()** of the class named Core, which belongs to the package **org.opencv.core.** following is the syntax of this method.

copyMakeBorder(src, dst, top, bottom, left, right, borderType)

This method accepts the following parameters

- **src** − An object of the class **Mat** representing the source (input) image.
- **dst** − An object of the class **Mat** representing the destination (output) image.
- **top** − A variable of integer the type integer representing the length of the border at the top of the image.
- **bottom** − A variable of integer the type integer representing the length of the border at the bottom of the image.
- **left** − A variable of integer the type integer representing the length of the border at the left of the image.
- **right** − A variable of integer the type integer representing the length of the border at the right of the image.
- **borderType** − A variable of the type integer representing the type of the border that is to be used.

5.3 Pytesseract

Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and "read" the text embedded in images.

Python-tesseract is a wrapper for Google's Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file.

Installation:

Use the following command to install pytesseract library.

pip install pytesseract

Pytesseract library has the following functions:

- **get_languages** Returns all currently supported languages by Tesseract OCR.
- **get_tesseract_version** Returns the Tesseract version installed in the system.
- **image_to_string** Returns unmodified output as string from Tesseract OCR processing
- **image_to_boxes** Returns result containing recognized characters and their box boundaries
- **image_to_data** Returns result containing box boundaries, confidences, and other information.
- **image_to_osd** Returns result containing information about orientation and script detection.
- **image_to_alto_xml** Returns result in the form of Tesseract's ALTO XML format.
- **run_and_get_output** Returns the raw output from Tesseract OCR. Gives a bit more control over the parameters that are sent to tesseract.

Tesseract tests the text lines to determine whether they are fixed pitch. Where it finds fixed pitch text, Tesseract chops the words into characters using the pitch, and disables the chopper and associator on these words for the word recognition step.

5.4 gTTS

gTTS (Google Text-to-Speech), a Python library and CLI tool to interface with Google Translate's text-to-speech API. Write spoken mp3 data to a file, a file-like object (bytestring) for further audio manipulation, or stdout. Or simply pre-generate Google Translate TTS request URLs to feed to an external program.

Installation

This gTTS module can be simpy downloaded by using the following command.

pip install gTTS

We will import the gTTS library from the gtts module which can be used for speech translation.

tts = gTTS(text)

The **text** variable is a string used to store the user's input. The text can be replaced by anything of your choice within the quotes. Another alternative can be to use the input statement for the user to type their own desired input each time the program is run.

The **tts** variable is used to perform the Google text-to-speech translation on the user's input. The output of the converted text is stored in the form of speech in the tts variable.

The tts.save function allows us to save the converted speech in a format that allows us to play sounds. This audio can be saved in a file with a format called .mp3. Other formats like .wav format can also be used.

5.5 pygame

The pygame library is an open-source module for the Python programming language specifically intended to help you make games and other multimedia applications. Built on top of the highly portable SDL (Simple DirectMedia Layer) development library, pygame can run across many platforms and operating systems.

By using the pygame module, you can control the logic and graphics of your games without worrying about the backend complexities required for working with video and audio.

Installing pygame using the following command

pip install pygame

In order to play music/audio files in pygame, pygame.mixer is used (pygame module for loading and playing sounds). This module contains classes for loading Sound objects and controlling playback. There are basically four steps in order to do so:

- Starting the mixer
  mixer.init()
- Loading the song.
  mixer.music.load("song.mp3")
- Setting the volume.
  mixer.music.set_volume(0.7)
- Start playing the song.
  mixer.music.play()

6. Flow of Process

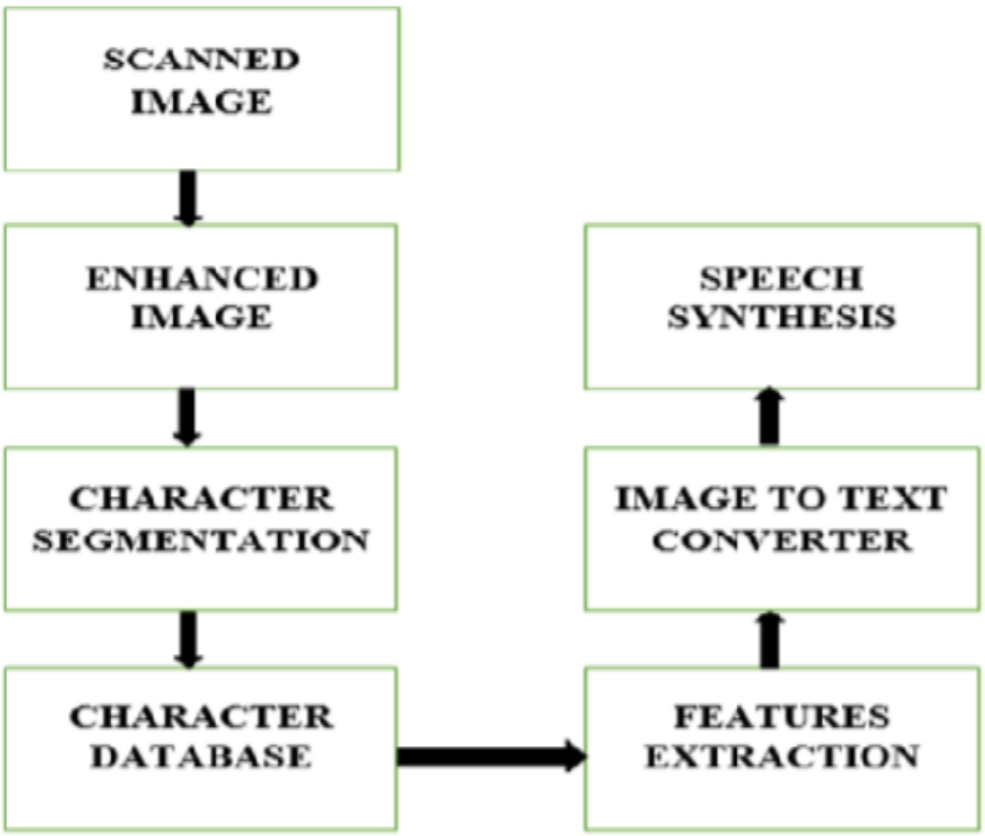

Fig.6: Flow of Process

The details of flow of process is explained below:

Image capturing

The primary step during which the device is moved over the printed page and therefore the camera captures the image of the text. the standard of the image captured are high so on have fast and clear recognition thanks to the high-resolution camera.

Pre-Processing

The pre-processing stage consists of three steps: Skew Correction, Linearization and Noise Removal. The captured image is checked for skewing. There are possibilities of the image getting skewed with either left or right orientation. Here the image is first brightened and binarized. The function for skew detection checks for an angle of orientation between ±15 degrees and if detected then an easy image rotation is distributed till the lines match with actuality horizontal axis, which produces a skew

corrected image. The noise introduced during capturing or because of the poor quality of the page needs to be cleared before further processing.

Segmentation

After pre-processing, the noise free image is passed to the segmentation phase. it's an operation that seeks to decompose a picture of sequence o characters into sub-image of individual symbol (characters). The binarized image is checked for inter line spaces. If inter line spaces are detected then the image is segmented into sets of paragraphs across the interline gap. The lines within the paragraphs are scanned for horizontal space intersection with relevancy the background. Histogram of the image is employed to detect the width of the horizontal lines. Then the lines are scanned vertically for vertical space intersection. Here histograms
are accustomed detect the width of the words. Then the words are decomposed into characters using character width computation.

Feature Extraction

Feature extraction is that the individual image glyph is taken into account and extracted for features. First a character glyph is defined by the subsequent attributes:

(1) Height of the character;

(2) Width of the character;

(3) Numbers of horizontal lines present—short and long;

(4) Numbers of vertical lines present—short and long;

(5) Numbers of circles present;

(6) Numbers of horizontally oriented arcs;

(7) Numbers of vertically oriented arcs;

(8) Centroid of the image;

(9) Position of the various features;

(10) Pixels within the various regions.

Image to Text Converter

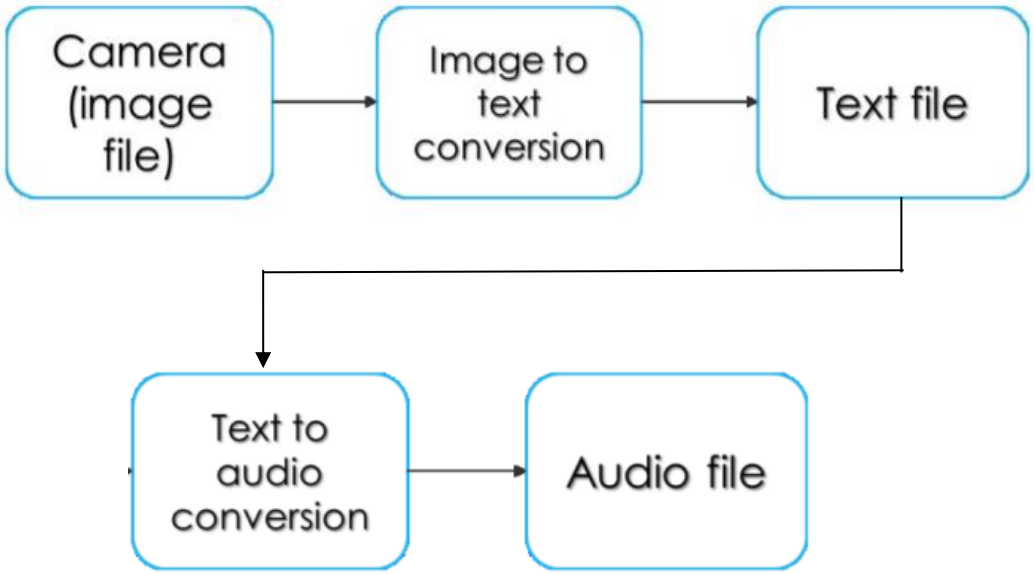

Fig.7: Image to Audio conversion

Tesseract tests the text lines to determine whether they are fixed pitch. Where it finds fixed pitch text, Tesseract chops the words into characters using the pitch, and disables the chopper and associator on these words for the word recognition step.

7. Working

The main steps that are used to obtain the required output are explained above in the form of flowchart. Working of this project in detailed is explained below.

The central idea of this project is to read out the images, that are captured using the camera module. Raspberry pi 3 model B, Pi camera, speaker/headphones are used in order to achieve this. To obtain any output using raspberry pi module, a code must be written in its corresponding software.

An operating system called Raspbian stretch is downloaded into the SD card of raspberry pi module being used. Once the Raspbian software is downloaded, insert the SD card into the SD card slot of raspberry pi. This software consists of terminal that can be used to the piece of code, which performs the required function.

Connect the raspberry pi module to monitor, which is used as screen to use the operating system installed in the SD card of raspberry pi. This monitor is connected to the HDMI port of the raspberry pi. Also connect mouse and keyboard to the USB ports of raspberry pi module.

Finally pi camera is to be connected to raspberry pi in-order to complete the setup. Before connecting the camera, ensure that the raspberry pi is turned off, to avoid any damages. To interface pi camera with raspberry pi, locate the camera module port. Gently pull upon the edges of the port's plastic clip. Insert the camera module ribbon cable; make sure the connectors at the bottom of the ribbon cable are facing the contacts in the port. Now, put the clip back into the place. After making all the connections, the setup looks like the one in the Fig.

Fig.8 : Raspberry pi board after making all the connections

A python program written to attain the required output is divided into five sub tasks.

1. Importing all the required libraries
2. Function to control camera capture
3. Function to process the image captured
4. Function to extract the text from the image captured in previous stage and store it in a file
5. Function to read the text in the file saved in above stage

All the libraries that are required are imported as below.

```
from picamera import PiCamera
import time
import cv2
import pytesseract
import numpy as np
import os
from gtts import gTTS
import pygame
```

picamera is the package that consists of PiCamera function, which allows to capture images using the camera module used. CV2 is the OpenCV package that is used to process the captured image. Pytesseract module is used to extract the text from the processed image. This text is stored in a file. gTTS (Google Text to Speech) is used to convert the text to audio and it is stored in audio file. Pygame module methods are used to play this audio file saved.

Camera capture function

```
camera = PiCamera()

def camera_capture():
        while(1):
                print("Image Capturing")
                camera.start_preview()
                time.sleep(5)
                camera.capture('/home/pi/opencv_blind/hello.jpg')
                camera.stop_preview()
return;
```

camera is the object that is used to call the PiCamera() class, which is available in picamera package. On calling the camera capture function, the preview of the content being captured is displayed on the screen that is connected to raspberry pi. Then after 5 seconds the image gets captured and is stored in the given location (/home/pi/opencv_blind/hello.jpg). After storing the captured image, the preview on the screen connected to raspberry pi module stops.

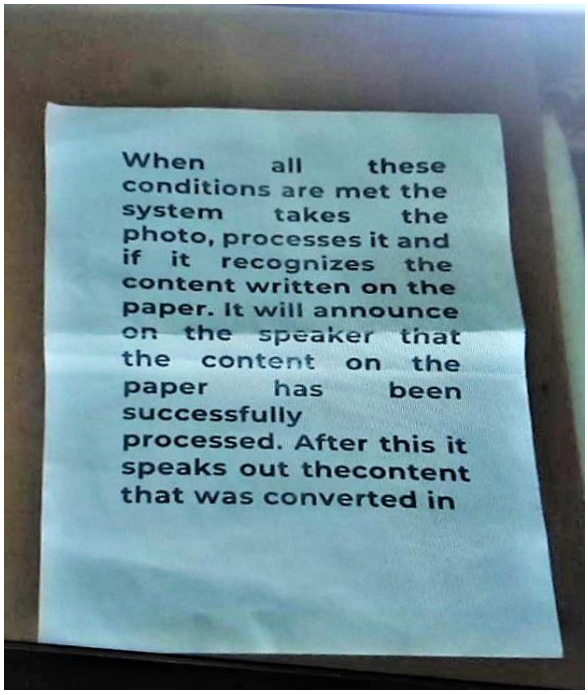

Fig.9 : Preview of the picture being captured on the screen.

Image processing function

```
def image_convert():

    print("Processing")
    img = cv2.imread('/home/pi/opencv_blind/hello.jpg')
    retval, threshold = cv2.threshold(img,127, 126, cv2.THRESH_BINARY)

    grayscaled = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    retval2, threshold2 = cv2.threshold(grayscaled, 127, 255,
    cv2.THRESH_BINARY)
    gaus = cv2.adaptiveThreshold(grayscaled, 255,
    cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 115,
    15)
    offset=50
    height, width = gaus.shape
    gaus=cv2.copyMakeBorder(gaus,offset,offset,offset,offset,cv2.BORDER_
```

```
CONSTANT,value=(255,255,255))
cv2.imwrite("/home/pi/opencv_blind/demo.jpg",gaus)
print("Process complete")
return;
```

The image captured in the previous step is to be preprocessed before extracting text out of it. In order to process the image, it is read using the imread() function of open CV package CV2. The function cv.threshold is used to apply the thresholding. The first argument is the source image, which should be a grayscale image. The second argument is the threshold value which is used to classify the pixel values. The third argument is the maximum value which is assigned to pixel values exceeding the threshold. OpenCV provides different types of thresholding which is given by the fourth parameter of the function. The image during preprocessing is converted into binary image using cvtColor function. After converting to binary image after using adaptive thresholding, a border is to be made to select the area in which the text is to be extracted. Finally after completion of the processing the image is to be stored as demo.jpg in location specified using imwrite() function as shown in the above block of code.

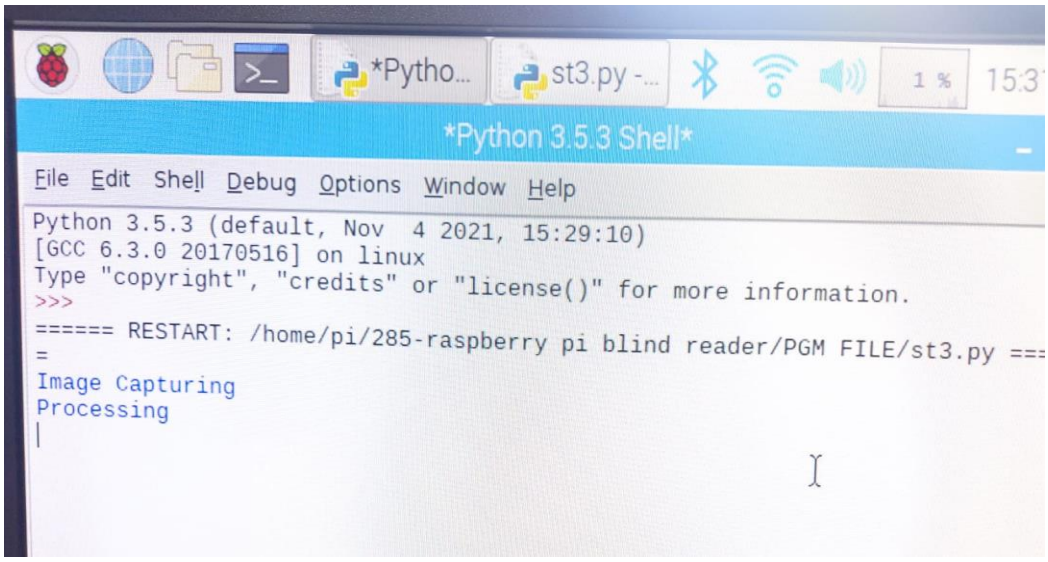

Fig. 10: Processing the text from the image captured

Extracting and saving the text from the preprocessed the image

```
def text_file():
    print("Converting To text")
    os.system('tesseract /home/pi/opencv_blind/demo.jpg /tmp/outfile')
    return;
```

Tesseract is the library that is used to extract the text from the preprocessed image.

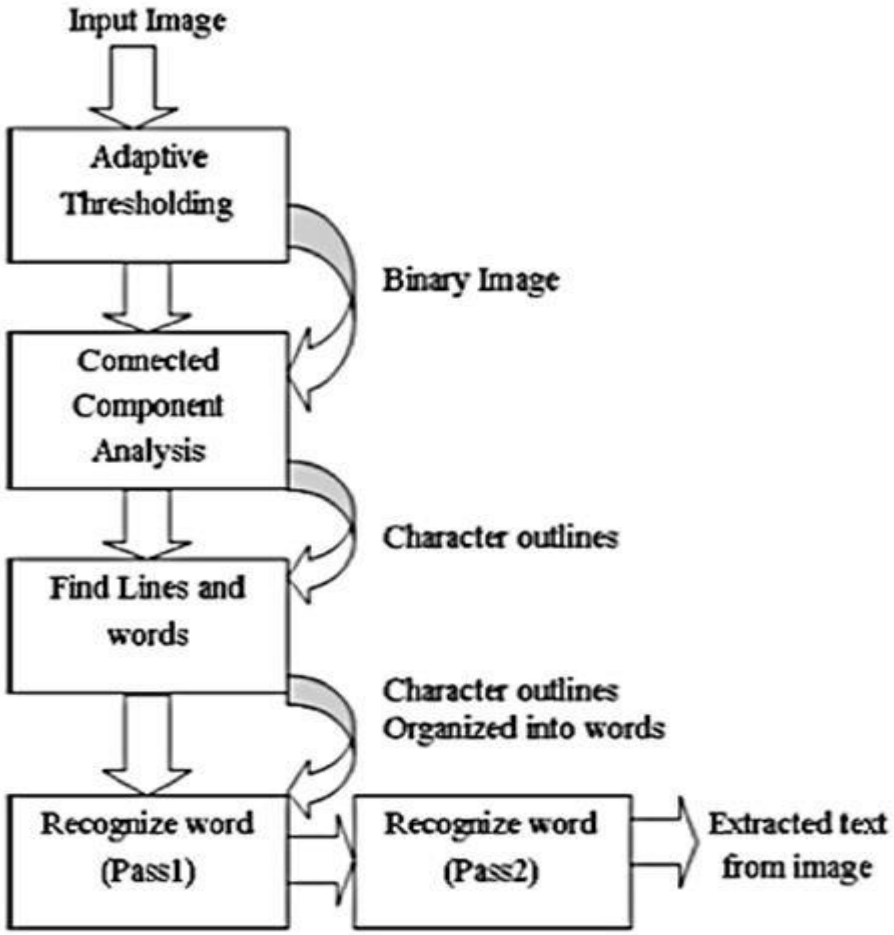

Fig.11: Tesseract flow

The above picture depicts the internal working of tesseract. The binary image that is obtained after adaptive thresholding is checked for connected component analysis, which is the stage-1 of recognition. The outlining of character is determined in this stage-1 analysis. This character outlining helps in determining words, sentences / lines. These character outlines is used to determine the words,

which is the second stage of recognition. After this stage-2 of recognition, all the words that are extracted are stored in a file, which in this case is named as outfile.txt.

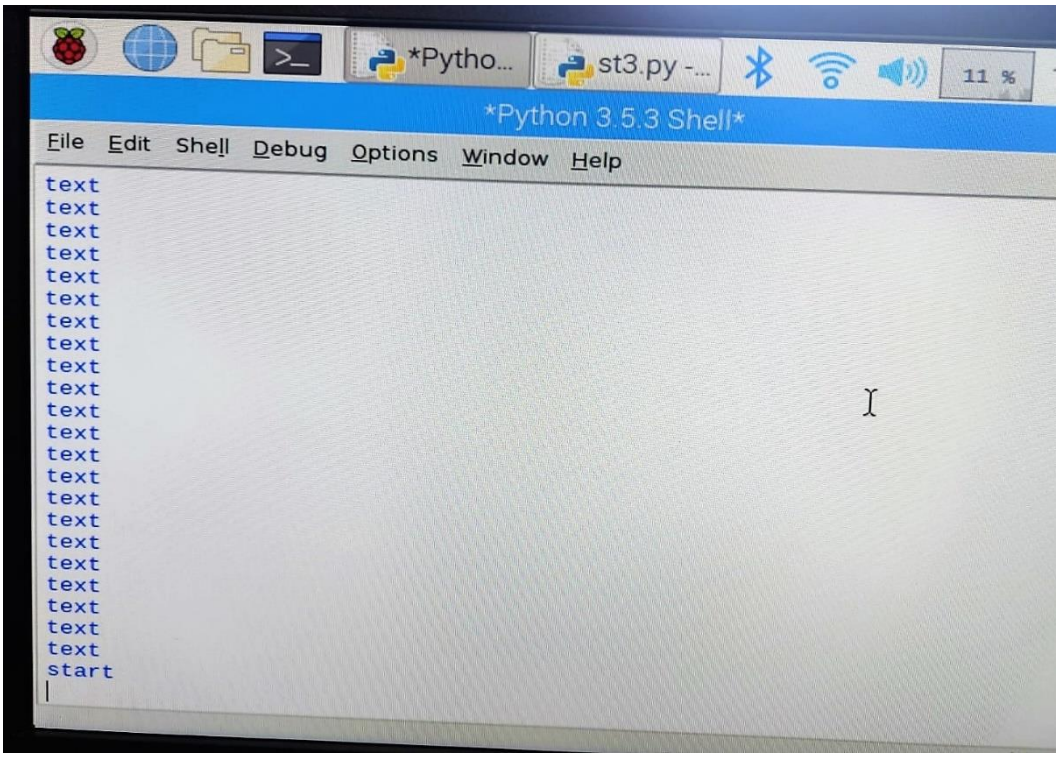

Fig.12: Text detection

Final stage of reading the text in the file:

```
def file_reader():
        print("Inside file reader")
        pygame.init()
        pygame.mixer.init()

        text=""
        with open("/tmp/outfile.txt","r") as file:
        for line in file:
        text=text+line
        print("text")
        print("start")
        speech = gTTS(text)
        speech.save('/home/pi/project/hello.mp3')
```

```
print("Saved the data")
time.sleep(1)

pygame.mixer.music.load('/home/pi/project/hello.mp3')
print("playing Speaker")
pygame.mixer.music.play()
time.sleep(50)
pygame.mixer.music.stop()
```

The file that contains the text extracted in the previous stage is opened in the read mode. Each line from this file is selected and is added to the variable called text . And now, the google text to speech module is given this text as input to convert it to audio form. This audio of the text is saved in the form of mp3 file. With the help of pygame library that is initialized n the beginning of this process, the audio file is played. User can listen to this through either headphones or speaker connected to the device. After the entire audio is played, this pygame library to play the music stops.

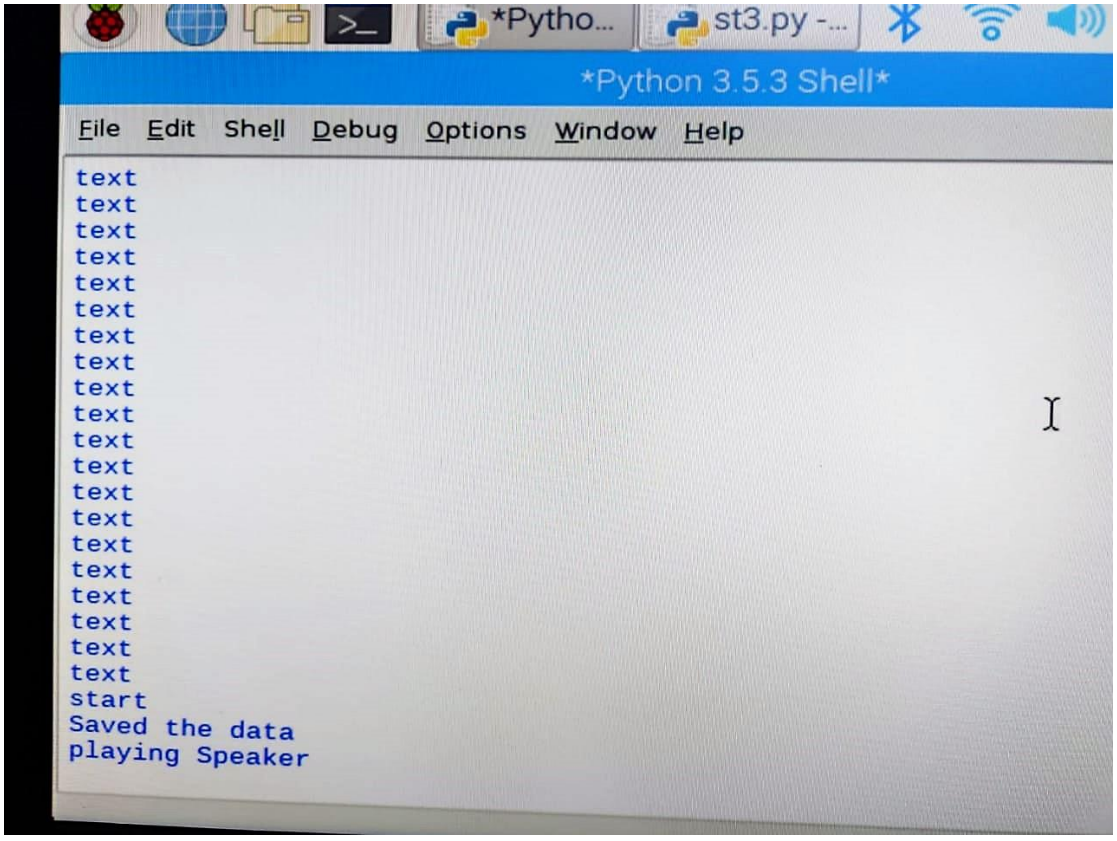

Fig.13: Speaker playing the audio

The final setup of the device is as below:

Fig.14: Final setup

8. Current Products in the market

- **Intelligent Camera App:** Seeing the AI app Microsoft created, a programme to assist people with limited eyesight by reporting on the world and assisting them in finding their way around. It can recognise Indian bank notes, find people's faces, and read facial traits, age, emotion, and other information. The app can also instantaneously hear brief bits of text and receive audio instruction for capturing complete papers.

Fig.15: Intelligent Camera App

- **Smart glasses for those with Low Vision:** The NuEyes Pro is the first wireless, voice-activated wearable gadget for the vision impaired, according to NuEyes. It works like electronic magnifying glasses, with a video camera on the front amplifying what you're looking at and displaying it on the inside of the spectacles. People suffering from macular degeneration, glaucoma, diabetic retinopathy, retinitis pigmentosa, and other visual diseases are helped from it.

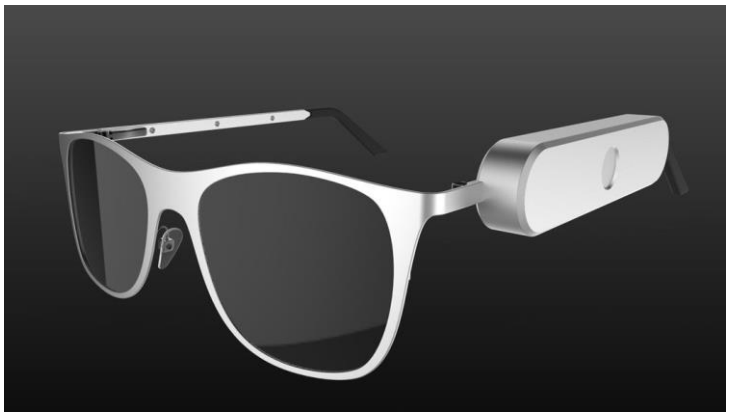

Fig.16: Smart glasses for those with Low Vision

- **Glasses for Remote Visual Assistance:** Aira Horizon smart glasses come with a human assistant service, by broadcasting live-streaming footage from the glasses camera to the company's agents, who can then deliver audio directions to the visually impaired user. The hands-free glasses have a trendy style and are equipped with AI to provide helpful technology. By broadcasting live-streaming footage from the glasses camera to the company's agents, who can then deliver audio directions to the visually impaired user, Aira Horizon smart glasses come with a human assistant service. The hands-free glasses have a trendy style and are equipped with AI to provide helpful technology.

Fig.17: Glasses for Remote Virtual Assistance

9. Advantages

- It is easy to use.
- It helps visually impaired people who does not know braille.
- It is easy to carry as it is small in size.

10.Limitations

- We require internet to run the program.
- Improper lighting may lead to low efficiency.
- Damage of camera may effect the output of the program

11.Futurescope

To interpret the image and separate the text from the photos, we can utilize more robust and efficient algorithms in the future. The captured image was blurry, and we needed to de-blur it quickly so that we could extract the data and convert it to voice effectively. Our suggested project will aid the visually impaired as well as the blind by taking into account all of these factors. We can also extend it to other languages and help people where language won't be a limitation to use.

12.Conclusion

The current products in market have few limitations. For Eg: Glasses for Visual assistance product can't assist you directly as the footage from the camera needs to be sent to company agents for assisting. So, in order to overcome such limitations, we used the Raspberry Pi to create an image-to-speech conversion mechanism. The image is effectively processed by our system, and it is read out clearly. This is a cost-effective and effective solution for visually impaired persons. We have successfully applied our algorithm to a large number of photos. The gadget is little and beneficial to society.

13.References

1. A. Ravi, SK. Khasimbee, T. Asha, T. Naga Sai Joshna, P. Gnana Jyothirmai, "Raspberry Pi based Smart Reader for Blind People"2020 International Conference on Electronics and Sustainable Communication System (ICESC), Coimbatore, India, July 2020, DOI: 10.1109/ICESC48915.2020.9155941.

2. AVM.Manikandan, Shouham Choudary, Souptik Majumder, "Text reader for visually impaired people: Any reader" 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), Chennai, India, September 2017, DOI: 10.1109/ICPCSI.2017.8392145.

3. Shahed Anzarus Sabab, Md. Hamjajul Ashmafee, "Blind Reader: An intelligent assistant for bling" 2016 19th International Conference on Computer and Information Technology (ICCIT), Dhaka, Bangladesh, December 2016, DOI: 10.1109/ICCITECHN.2016.7860200.