VOICESTICK – TEXT READER FOR VISUALLY IMPAIRED

A FINAL YEAR PROJECT REPORT

Submitted by

KAAVIYA.A	**Reg. No.:13BEC059**
KEERTHIKA.T	**Reg. No.:13BEC209**
BANU.P	**Reg. No.:13BEC219**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION

ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE - 641 049

(An Autonomous Institution Affiliated to Anna University, Chennai)

APRIL 2017

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE – 641 049

(An Autonomous Institution Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report titled **“VOICESTICK – TEXT READER FOR VISUALLY IMPAIRED”** is the bonafied work of “**KAAVIYA.A, KEERTHIKA.T, BANU.P ”** who carried out the project work under my supervision. . Certified further that, to the best of my knowledge the work reported herein does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Ms.S.Sasikala M.E.,	Dr. Malarvizhi Ph. D.,
Associate Professor/E.C.E	**HEAD OF THE DEPARTMENT**
Kumaraguru College of Technology	Electronics & Communication Engineering
Coimbatore.	Kumaraguru College of Technology
	Coimbatore.

The candidates with Register numbers 13BEC059,13BEC209,13BEC219 are examined by us in the project viva-voce examination held on …………………….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

First we would like to express our praise and gratitude to the Lord, who has showered his grace and blessing enabling us to complete this project in an excellent manner. He has made all things in beautiful in his time.

We express our sincere thanks to our beloved Joint Correspondent, **Shri. Shankar Vanavarayar** for his kind support and for providing necessary facilities to carry out the project work.

We would like to express our sincere thanks to our beloved Principal **Dr.R.S.Kumar M.E., Ph.D.,** who encouraged us with his valuable thoughts.

We would like to express our sincere thanks and deep sense of gratitude to our HOD, **Dr.K.Malarvizhi,M.E.,Ph.D.,** for her valuable suggestions and encouragement which paved way for the successful completion of the project.

We are greatly privileged to express our deep sense of gratitude to the Project Coordinator **Mr.S.Govindaraju M.E.,** ,Professor, for her continuous support throughout the course.

In particular, We wish to thank and express our everlasting gratitude to the Supervisor **Ms.S.Sasikala,M.Tech.,** Associate Professor for her expert counselling in each and every steps of project work and we wish to convey our deep sense of gratitude to all teaching and non-teaching staff members of ECE Department for their help and cooperation.

Finally, we thank our parents and our family members for giving us the moral support in all of our activities and our dear friends who helped us to endure our difficult times with their unfailing support and warm wishes.

ABSTRACT

In the real world, books and documents are the sources of knowledge. But this knowledge is only bounded to people with clear vision. Our society includes a group of people who does not have a clear vision or people who are blind. For this group, world is like a black illusion. The shape and structure's information of an object is unavailable to them let alone reading a document. For blind acquiring knowledge by reading documents is cumbersome. Braille is one of the methods which is used to read a book or document. In this method, any document has to be converted to braille format to become understandable to a blind. The problem arises due to the fact that, this is an expensive procedure and many times not available. The solution is rather simple, introduce a smart device that can convert any document to the interpreted form to a blind. A blind student can the read book in library by capturing the page as image which is then audibly presented through text to speech engine. “Voice Stick” - developed for both low vision and completely blind which is user friendly and effective interactive reading and navigating device.

TABLE OF CONTENTS

LIST OF ABBREVATIONS

S.NO	ABBREVATIONS	EXPANSIONS
1.	SIFT	Scale Invariant Feature Transform
2.	FLANN	Fast Library for Approximate Nearest Neighbors
3.	BCM	Broad Com
4.	OCR	Optical Character Recognition
5.	PIL	Python Imaging Library

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

.

About 15 percent of the entire world's population has disability. 285 million people are estimated to be visually impaired worldwide. In that 39 million are blind and 246 have low vision. In our world information is generally available in the form of books and documents. It is fully usable for the sighted people. From an ancient time, information is resembled in aural format as no other representation of it is founded in printing format. When an era has come of printing it facilitates the sighted people partially to acquire knowledge. A major problem for a blind or visually impaired person (BVI) to interact with the world to share knowledge. For them information has to be in a special tactile language or in voice format. They are affected in every works of their daily life. Nowadays technology helps them to overcome this difficulty to some extent. Many hardware or software tools are invented to help them . The most difficult task for them is reading text from the books or documents and navigation . We propose an effective and useful text reader module to help the visually impaired to read any sort of printed material. Voicestick is the text reader module which captures the image to be converted into string of text and produce voice output. It also has the feature of object recognition. Voicestick has the navigation module to help the blind reach his destination provided sign direction to the places like library, rest room, class room, temple etc. to be placed on the way of their route. The advantage of this project is that it serves the blind to reach his destination and identify the signs and objects on his way through object detection module. Once desired designation is reached say library, he can disconnect the navigation module and use the text reader module to read any printed material of texts say book in a library independently without any external support. Conventionally text reader machines, softwares and human readers are used to read the books for visually challenged students. This project is mainly designed for the visually challenged students to read the book independently without the support of anything. He can access his own school or college library provided way to library signs displayed on the route. The student can reach the library and before reading disconnect walking stick with ultrasonic sensors attached for navigation purpose. Now he can use the text reader pen like (advanced after Research & Development) module to read the text in the book with the image being captured by the camera and extracting text through digital image processing and producing the speech output. The output is taken through headphones in order to avoid distractions and disturbance on the way and to others. Visually challenged people can access and use this module by pressing on the specific buttons made for each module. There are four buttons provided for power, object recognition,image capturing and voice output . Ultrasonic sensors works on switching on the power button and can be disabled while reading. Camera will be enabled on object detection phase and also in capturing the pages of the book to be read.

CHAPTER 2

LITERATURE SURVEY

The project has been evoled from many papers from past 5 years. the advantages and disanvantages have been listed below

S.NO	AUTHOR	TITLE	ADVANTAGES	DISADVANTAGES
1	David J. Brown; Michael J. Proulx	Audio–Vision Substitution for Blind Individuals: Addressing Human Information Processing Capacity Limita tions(2016)	Sensory substitutues used.	Inaccuracy, Outsude environment inaccessible.
2	Kishor Bhurchandi; Abhinav Kulkarni	Low Cost E-Book Reading Device for Blind People(2015)	Low Cost, Converts text into Braille	Braille Language is mandatory

CHAPTER 3
BLOCK DIAGRAM

The Voicestick - text reader for visually challenged people is a device that has to two main modules connected to one processor raspberry pi 3 model B. The first one is navigation module for reaching the destination using ultrasonic sensor and camera used for object / sign recognition with defined descriptions for each sign say men and women restroom signs are predefined in the program.

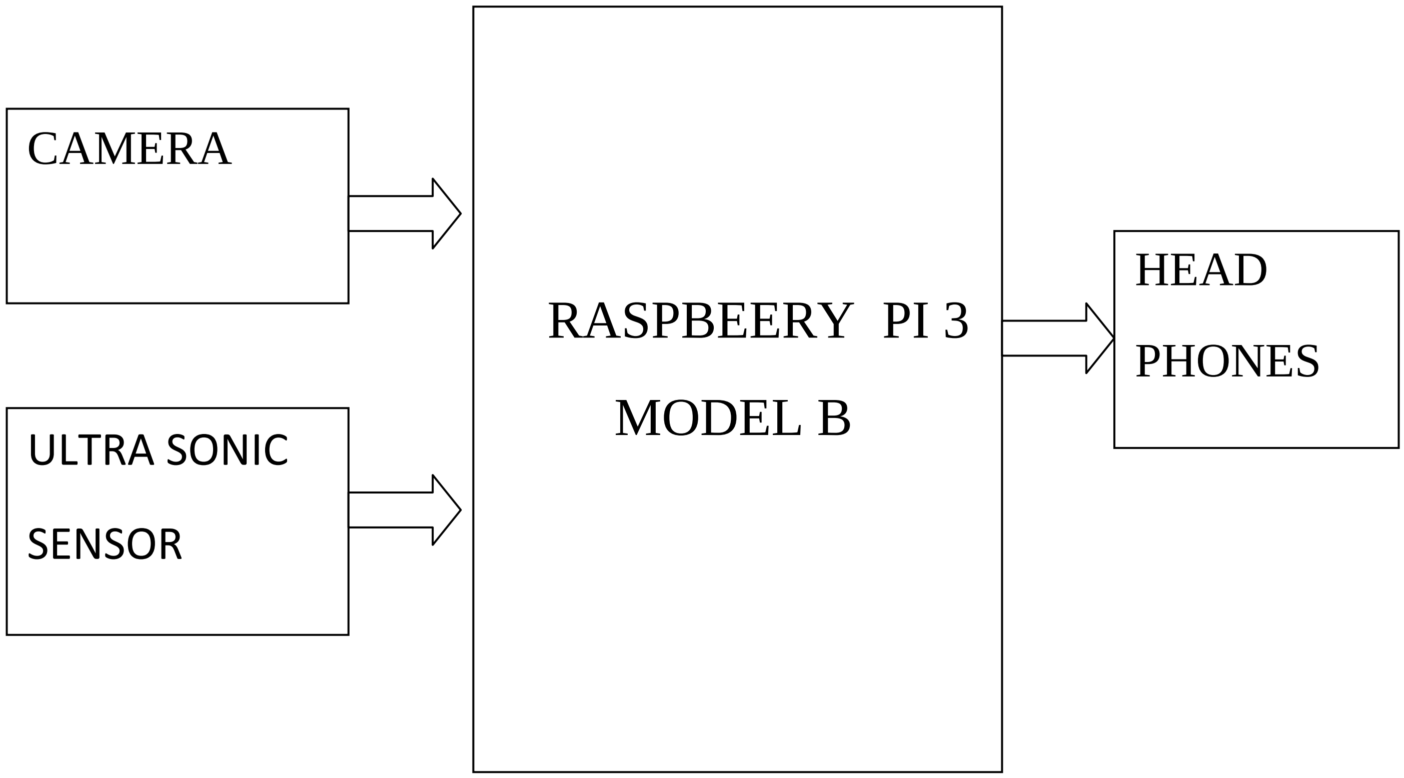


Fig 3 BLOCK DIAGRAM

The second module in the **Fig 3** is for reading purpose to capture the image where the image is converted to strings of text and the speech output is obtained through headphones using voice output buttons. Separate buttons are provided for object recognition capturing image, providing speech output of captured image.

CHAPTER 4

CIRCUIT DIAGRAM

The circuit diagram of Voicestick in **fig 4 -** text reader for visually impaired consists of ultrasonic sensors to detect the obstacles on the way which will be acknowledged by " Obstacle detected " voice output. The trigger and echo pins of ultrasonic sensor is connected to pin 23 and pin24 respectively. Object detection is performed through identification of signs placed on the way. With predefined description and defined voice output for each signs.

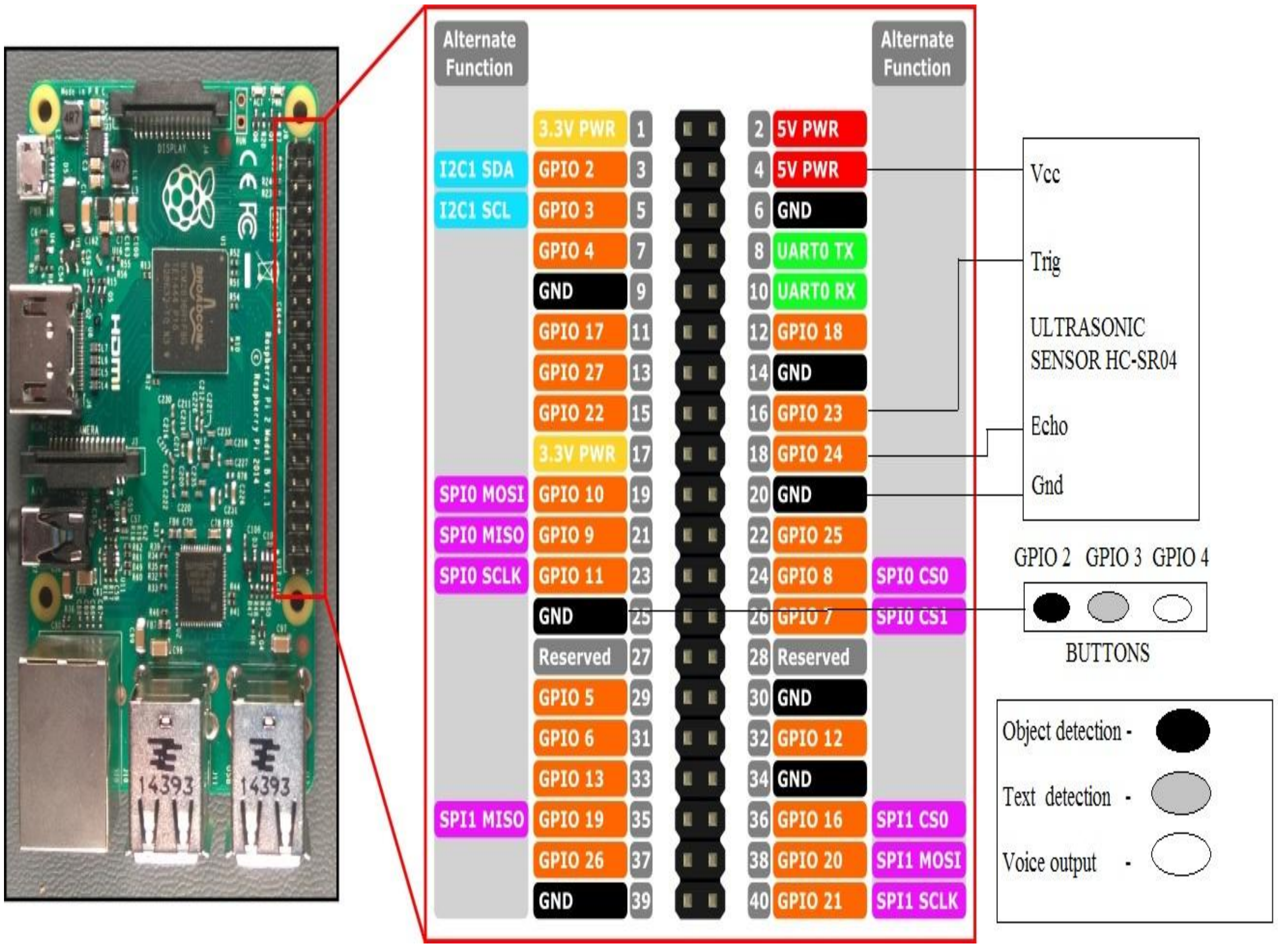


Fig 4 CIRCUIT DIAGRAM

The camera is connected to the camera interface provision of raspberry pi3 model B. Camera is enabled on detecting objects on the way and when pressing the camera button correspondingly pin 3, the image to be read will be captured . The voice output is taken through headphones connected to pin 4. It automatically provides speech output for obstacle detection and object recognition. It provides specific speech output for captured image through special button provided. Raspberry pi 3 model B a mini computer is used as processor combines all these functions through a detailed python coding defined for each module. It uses Linux operating system stored using external storage element. The coding is also embedded on the processor. This is presented as a module for text reading and object recognition further features like face recognition and Braille buttons for blind can be added by including respective coding for each functions.

. CHAPTER 5

HARDWARE DESCRIPTION

5.1 RASPBERRY PI

5.1.1 INTRODUCTION:

The Raspberry Pi is a series of credit card–sized single-board computers developed in England, United Kingdom by the Raspberry Pi Foundation with the intent to promote the teaching of basic computer science in schools and developing countries. The original Raspberry Pi and Raspberry Pi 2 are manufactured in several board configurations through licensed manufacturing agreements with Newark element14 (Premier Farnell), RS Components and Egoman. The hardware is the same across all manufacturers.

All Raspberry Pis include the same VideoCore IV graphics processing unit (GPU), and either a single-core ARMv6-compatible CPU or a newer ARMv7-compatible quad-core one (in Pi 2); and 1 GB of RAM (in Pi 2), 512 MB (in Pi 1 models B and B+), or 256 MB (in models A and A+, and in the older model B). They have a Secure Digital (SDHC) slot (models A and B) or a MicroSDHC one (models A+, B+, and Pi 2) for boot media and persistent storage. In 2014, the Raspberry Pi Foundation launched the Compute Module, for use as a part of embedded systems for the same compute power as the original Pi. In early February 2015, the next-generation Raspberry Pi, Raspberry Pi 2, was released. That new computer board is initially available only in one configuration (model B) and has a quad-core ARM Cortex-A7 CPU and 1 GB of RAM with remaining specifications being similar to those of the prior generation model B+. The Raspberry Pi 2 retains the same US$35 price of the model B,with the US$20 model A+ remaining on sale. In November 2015, the Foundation launched the Raspberry Pi Zero, a smaller product priced at US$5. Raspberry Pi 3 was released on 29 February 2016.

The Foundation provides Debian and Arch Linux ARM distributions for download, and promotes Python as the main programming language, with support for BBC BASIC, C, C++, Java, Perl, Ruby, Squeak Smalltalk and more also available.

Raspberry Pi 3 has a new BCM2837 SoC retaining compatibility with the GPU, CPU and connectors of its predecessors BCM2835 (Pi 1) and BCM2836 (Pi 2), so all those projects and tutorials for Pi 1 and Pi 2 hardware should continue to work. The 900 MHz 32-bit quad-core ARM Cortex-A7 CPU complex has been replaced by a 1.2 GHz 64-bit quad-core ARM Cortex-A53. Combining a 33% increase in clock speed with various architectural enhancements, this provides a 50–60% increase in performance in 32-bit mode versus Raspberry Pi 2, or roughly a factor of ten over the original Pi 1.

5.1.2 Hardware

The Raspberry Pi hardware has evolved through several versions that feature variations in memory capacity and peripheral-device support. Model A and A+ and Zero lack the Ethernet and USB hub components. The Ethernet adapter is connected to an additional USB port. In model A and A+ the USB port is connected directly to the SoC. On model B+ and later models the USB/Ethernet chip contains a five-point USB hub, of which four ports are available, while model B only provides two. On the model Zero, the USB port is also connected directly to the SoC, but it uses a micro USB (OTG) port.

5.1.3 Processor

The system on a chip (SoC) used in the first generation Raspberry Pi is somewhat equivalent to the chip used in old-er smartphones (such as iPhone, 3G, 3GS). The Raspberry Pi is based on the Broadcom BCM2835 SoC, which includes an 700 MHz ARM1176JZF-S processor, VideoCore IV graphics processing unit (GPU) and RAM. It has a Level 1 cache of 16 KB and a Level 2 cache of 128 KB. The Level 2 cache is used primarily by the GPU. The SoC is stacked underneath the RAM chip, so only its edge is visible.

The Raspberry Pi 2 uses a Broadcom BCM2836 SoC with a 900 MHz 32-bit quad-core ARM Cortex-A7 processor, with 256 KB shared L2 cache.

The Raspberry Pi 3 uses a Broadcom BCM2837 SoC with a 1.2 GHz 64-bit quad-core ARM Cortex-A53 processor, with 512 KB shared L2 cache.Raspberry Pi 2 is based on Broadcom BCM2836 SoC, which includes a quad-core Cortex-A7 CPU running at 900 MHz and 1 GB RAM. It is described as 4–6 times more powerful than its predecessor. The GPU is identical to the original.

5.1.4 RAM:

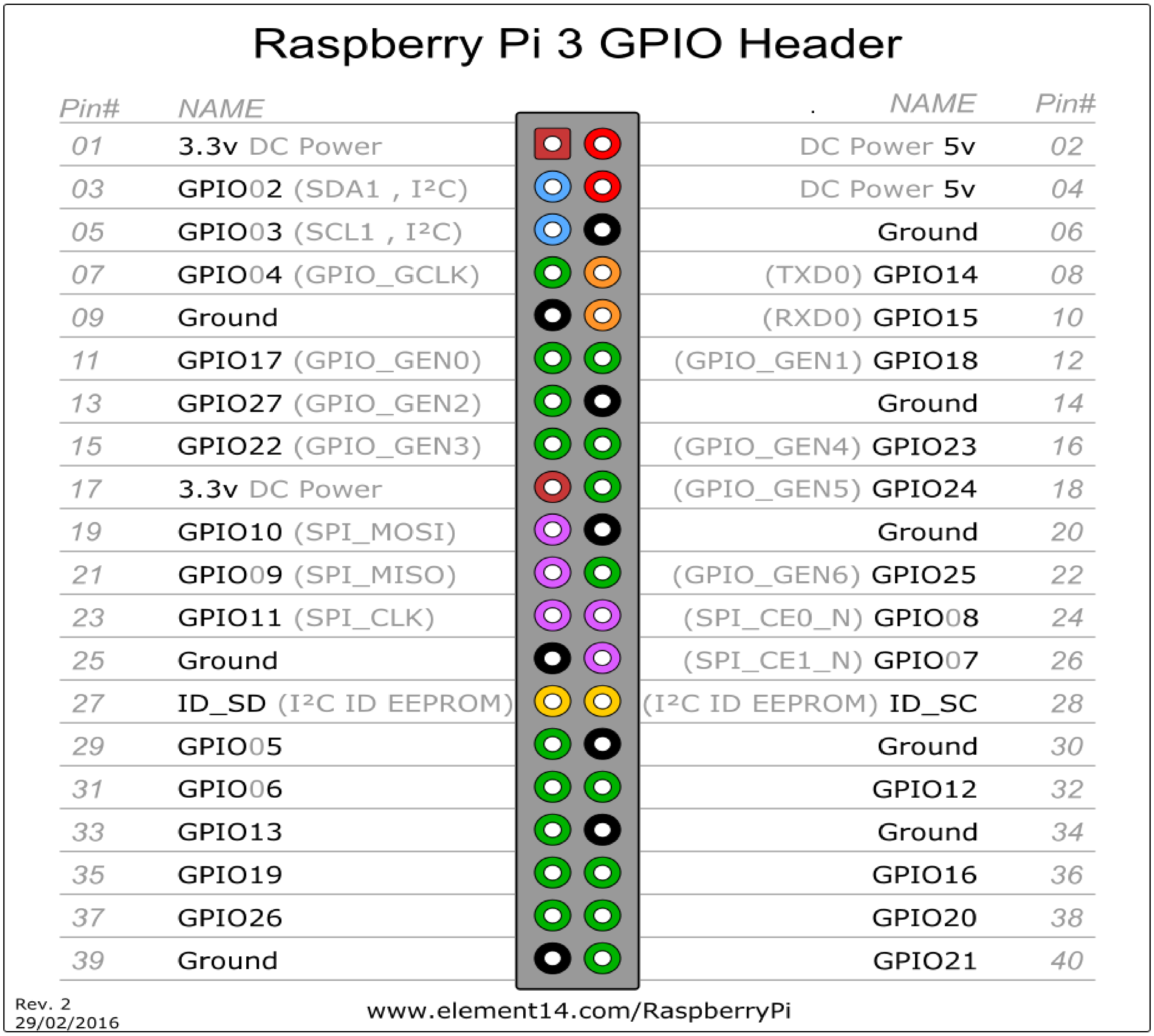
On the older beta model B boards, 128 MB was allocated by default to the GPU, leaving 128 MB for the CPU. On the first 256 MB release model B (and model A), three different splits were possible. The default split was 192 MB (RAM for CPU), which should be sufficient for standalone 1080p video decoding, or for simple 3D, but probably not for both together. 224 MB was for Linux only, with only a 1080p framebuffer, and was likely to fail for any video or 3D. 128 MB was for heavy 3D, possibly also with video decoding (e.g. XBMC). Comparatively the Nokia 701 uses 128 MB for the Broadcom VideoCore IV. For the new model B with 512 MB RAM initially there were new standard memory split files released

The Raspberry Pi 2 and the Raspberry Pi 3 have 1 GB of RAM. The Raspberry PI Zero has 512 MB of RAM.

5.1.5 Networking:

Though the model A and A+ and Zero do not have an 8P8C ("RJ45") Ethernet port, they can be connected to a network using an external user-supplied USB Ethernet or Wi-Fi adapter. On the model B and B+ the Ethernet port is provided by a built-in USB Ethernet adapter. The Raspberry Pi 3 is equipped with 2.4 GHz WiFi 802.11n and Bluetooth 4.1 in addition to the 10/100 Ethernet port.

5.1.6 Pin diagram:



Raspberry Pi 3 GPIO Header

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I²C)	DC Power 5v	04
05	GPIO03 (SCL1 , I²C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)	(I²C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Rev. 2
29/02/2016

www.element14.com/RaspberryPi

Fig 5.1.6 GPIO PIN DIAGRAM

5.1.6 Peripherals

The Raspberry Pi may be operated with any generic USB computer keyboard and mouse.

5.1.7 Video:

The video controller is capable of standard modern TV resolutions, such as HD and Full HD, and higher or lower monitor resolutions and older standard CRT TV resolutions. As shipped (i.e. without custom overclocking) it is capable of the following: 640×350 EGA; 640×480 VGA; 800×600 SVGA; 1024×768 XGA; 1280×720 720p HDTV; 1280×768 WXGA variant; 1280×800 WXGA variant; 1280×1024 SXGA; 1366×768 WXGA variant; 1400×1050 SXGA+; 1600×1200 UXGA; 1680×1050 WXGA+; 1920×1080 1080p HDTV; 1920×1200 WUXGA.

5.2 LOGITECH C170 WEB CAMERA:

We use a web camera as a surveillance camera in this project. The webcam used here is Logitech C170. It's specification sheet is given below:

5.2.1 Technical Specifications

- Video calling (640 x 480 pixels) with recommended system
- Video capture: Up to 1024 x 768 pixels
- Logitech Fluid Crystal™ Technology
- Photos: Up to 5 megapixels (software enhanced)
- Built-in mic with noise reduction
- Hi-Speed USB 2.0 certified

5.3 ULTRASONIC SENSOR:

Ultrasonic transducers are divided into three broad categories: transmitters, receivers and transceivers. Transmitters convert electrical signals into ultrasound, receivers convert ultrasound into electrical signals, and transceivers can both transmit and receive ultrasound. Ultrasonic sensor is used for object detection which is enabled on navigation and object recognition requirements. The trigger and echo pins of ultrasonic sensor is connected to pin 23 and pin 24 of raspberry pi 3 model B processor. The object is detected by the trigger signal reflected from the object and echoed at the echo pin. Using program the function of ultrasonic sonic sensor is controlled by specifying the distance limit and distance at which the acknowledgement to the blind has to be given. The distance is calculated using the time period of transmission and reflection period of the signal. The program is designed in such a way that the calculated distance will be displayed on the output screen of Linux . It works on the range if threshold value fixed and whenever the distance exceeds the threshold value out of range acknowledgement will be given and whenever the object is detected " Object detected" voice output will be given through the headphones.

5.3.1 Features:

- Ideal for use in beam detection systems.
- Sensitivity >-65dB
- Nominal frequency 40kHz
- Used for measuring wind speed and direction
- Further applications include: humidifiers, sonar, medical ultrasonography ,burglar alarms, non-destructive testing and wireless charging.

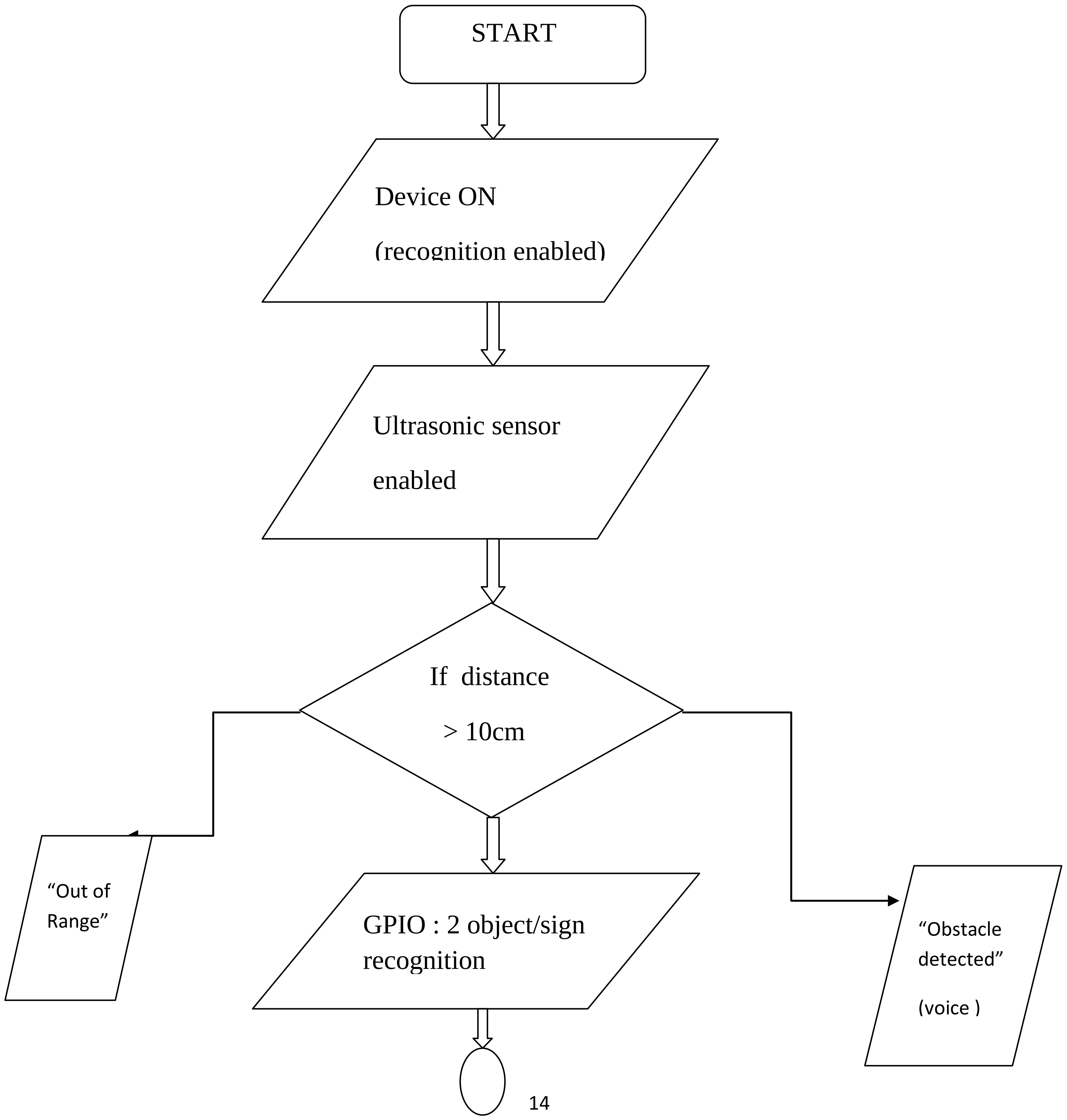
5.4 HEADPHONES :

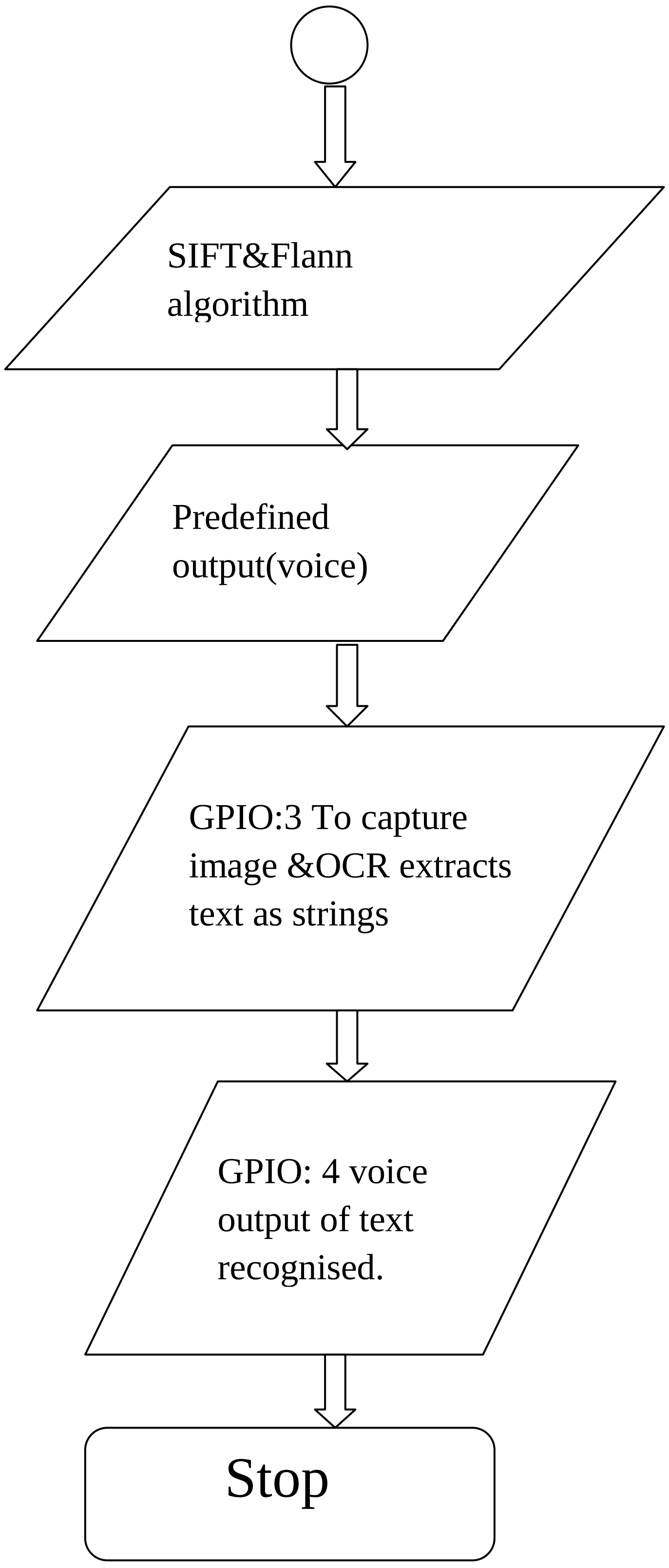
They are electroacoustic transducers, which convert an electrical signal to a corresponding sound in the user's ear. Headphones are designed to allow a single user to listen to an audio source privately, in contrast to a loudspeaker, which emits sound into the open air, for anyone nearby to hear. Headphones are also known as earspeakers or earphones . Headphones are used as external output interface. Headphones work throughout the project whenever an obstacle is detected it gives " Obstacle detected" speech output. Object recognition output will provide with defined descriptions for each predefined sign in the program. Pin 4 specifically provides extracted text output. Headphones are connected to the audio interface pin of the processor**.**

5.5 ETHERNET CABLE:

Used to connect raspberry pi and system. Ethernet cable provides interface between the raspberry pi 3 model B processor and the system which displays the working of Linux operating system on python code programmed for the Voicestick project.

CHAPTER 6
FLOW CHART




SIFT&Flann algorithm
Predefined output(voice)
GPIO:3 To capture image &OCR extracts text as strings
GPIO: 4 voice output of text recognised.
Stop

6.1 Flow chart description:

In Raspberry pi 3 model B processor, python program is coded which operates in Linux operating system. There are four buttons for power, object recognition, capture image and output speech output f the captured image.

- Initially when the device is turned ON, “Recognition enabled” prompt will be displayed on Linux window.
- Ultrasonic sensor will be enabled stimulating obstacle detection operation
- Distance of the obstacle will be detected by seting a threshold value say 10 cm within which the processor works in two conditions.
- If YES, “Obstacle detected” speech output will be produced
- If No, Out of range will be prompted on linux window.
- On pressing pin 2, Object Recogniton operation performed.
- Camera works automatically and the captured signs on the way is matched with the predefined signs in database .
- If the predefined signs are recognised, corresponding description coded in program will produce the speech output.
- On pressing pin 3, the image to be read is captured, tesseract tool which performs OCR(optical character recognition) operation displays extracted text as strings in linux window for reference.
- On pressing pin 4. The speech output of the extracted image is given to headphones by espeak tool in python coding which converts the text to speech.

CHAPTER 7

SOFTWARE DESCRIPTION

7.1 PYTHON

Introduction:

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library.

Features:

Python is a multi-paradigm programming language: object-oriented programming and structured programming are fully supported, and there are a number of language features which support functional programming and aspect-oriented programming Many other paradigms are supported using extensions, including design by contractand logic programming. Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management.

An important feature of Python is dynamic name resolution (late binding), which binds method and variable names during program execution.The design of Python offers some support for functional programming in the Lisp tradition.

Libraries:

Python has a large standard library, commonly cited as one of Python's greatest strengths, providing tools suited to many tasks. This is deliberate and has been described as a "batteries included" Python philosophy. For Internet-facing applications, a large number of standard formats and protocols (such as MIME and HTTP) are supported. Modules for creating graphical user interfaces, connecting to relational databases, pseudorandom number generators, arithmetic with arbitrary precision decimals,manipulating regular expressions, and doing unit testing are also included.

Development Environment:

Python's development is conducted largely through the Python Enhancement Proposal (PEP) process. The PEP process is the primary mechanism for proposing major new features, for collecting community input on an issue, and for documenting the design decisions that have gone into Python.Outstanding PEPs are reviewed and commented upon by the Python community and by Van Rossum, the Python project's benevolent dictator for life.The mailing list python-dev is the primary forum for discussion about the language's development; specific issues are discussed in the Roundup bug tracker maintained at python.org.Development takes place on a self-hosted source code repository running Mercurial.

7.2 OpenCV:

The **OpenCV** (*Open Source Computer Vision*) is a library of programming functionsmainly aimed at real-time computer vision.

The goals of OpenCV were described as:

- Advance vision research by providing not only open but also optimized code for basic vision infrastructure.
- Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.
- Advance vision-based commercial applications by making portable, performance-optimized code available for free—with a license that did not require code to be open or free itself.
- The first alpha version of OpenCV was released to the public at the IEEE Conference on computer vision and pattern recognition in 2000.
- OpenCV runs on a variety of platforms.
  Desktop: Windows, Linux, macOS, FreeBSD, NetBSD, OpenBSD;
  Mobile: Android, iOS, Maemo, BlackBerry 10

7.2.1 Installing open CV in Raspberry pi

To install OpenCV the following commands are typed in the command line of the RPi.

```
sudo apt-get update
sudo apt-get upgrade
#check to see that the webcam, or pi-cam is being read by the pi
#for easy package installation, downloading the synaptic
sudo apt-get install synaptic
```

```
#install the python scipy stack
sudo apt-get install python-numpy python-scipy python-matplotlib
ipython- notebook python-pandas python-sympy python-nose
#to get the OpenCV zip or the tar.bz2 or the tar.gz:
wget http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.9/opencv-2.4.9.zip
#install basic development environment
sudo apt-get install build-essential cmake pkg-config
sudo apt-get install default-jdk ant
sudo apt-get install libgtkglext1-dev
sudo apt-get install bison
sudo apt-get install qt4-dev-tools libqt4-dev libqt4-core libqt4-gui
sudo apt-get install v4l-utils
sudo apt-get install qtcreator
sudo apt-get install libjpeg8 libjpeg8-dev libjpeg8-dbg libjpeg-progs ffmpeg libavcodec-dev libavcodec53
libavformat53 libavformat-dev libgstreamer0.10-0-dbg libgstreamer0.10-0 libgstreamer0.10-dev libxine1-
ffmpeg libxine-dev libxine1-bin libunicap2 libunicap2-dev swig libv4l-0 libv4l-dev python-numpy
libpython2.7 python-dev python2.7-dev libgtk2.0-dev
#ready to start compiling opencv
#now extract the opencv folder by unzipping the zip file downloaded from Sourceforge
unzip opencv-2.4.9.zip
#change directory to the folder that was extracted
cd opencv-2.4.9.1
#create a directory called build
mkdir build
#change into that recently created directory
cd build
sudo make
sudo make install
#create the following file:
sudo nano /etc/ld.so.conf.d/opencv.conf
#enter the following line into the empty file:
/usr/local/lib
#after saving the file, enter the following command:
sudo ldconfig
```

7.3 TESSERACT (TEXT TO STRING CONVERSION)

Python-tesseract is a python wrapper for google's Tesseract-OCRPython-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. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Python Imaging Library, including jpeg, png, gif, bmp, tiff, and others, whereas tesseract-ocr by default only supports tiff and bmp.

Additionally, if used as a script, Python-tesseract will print the recognized text in stead of writing it to a file. Support for confidence estimates and bounding box data is planned for future releases.

USAGE:

```
> import Image
> except ImportError
> from PIL import Image
> import pytesseract
> print(pytesseract.image_to_string(Image.open('test.png')))
> print(pytesseract.image_to_string(Image.open('test-european.jpg')
```

7.4 PIL COMMAND

The **Python Imaging Library (PIL)** adds image processing capabilities to your Python interpreter. This library supports many file formats, and provides powerful image processing and graphics capabilities.

7.5 GPIO PINS

The GIPO pins on a Raspberry pi are a great way to interface physical devices like buttons and leds with the little linux processor. if you're a phython developer, there's a sweet library called RPi.GPIO that handles interfacing with the pins.in just three lines of code,you can get an LED blinking on one of the GPIO pins.
To add the GPIO library to a phython sketch,you must first import it:

```
import RPI.GPIO as GPIO
```

Then we need to declare the type of numbering system we're going to use for our pins:

```
#set up GPIO using BCM numbering
GPIO.setmode(GPIO.BCM)
```

7.6 NAVIGATION

The ultrasonic sensor is used for obstacle detection on the way of reaching library. it is coded in software as follows:

Provide TRIGGER to Raspberry Pi
> Listen for ECHO
> Measure the time period of ECHO
> Convert it to Distance by using Speed Distance Time Equation

```
distance = pulse_duration * 17150

importRPi.GPIO as GPIO        #Import GPIO library
import time                    #import time library
GPIO.setmode(GPIO.BCM)        #set GPIO pin numbering

TRIG= 23            #Associate pin 23 to TRIG
```

```
ECHO =24                  #Associate  pin   24 to ECHO

print"distance is measurement in progress"
GPIO.setup(TRIG,GPIO.OUT)       #set pin as GPIO out
GPIO.setup(ECHO,GPIO.IN)          #set pin as GPIO in
```

7.7 REMOTE DESKTOP CONNECTION:

Since it is not convenient to access the Pi by connecting the peripherals every time to the Pi, it is better to access the Pi over the network using Remote Desktop Connection from our Personal Computer. Using this method it is enough if we just connect the LAN cable to the Pi and power it up. The Pi’s IP address is entered into the dialog box when we open Remote Desktop Connection and then we can access the Pi from our PC. To use this feature a software called xrdp has to be installed on the Pi. Xrdp can be installed on the Pi using the following command:

sudo apt-get install xr

CHAPTER 8
WORKING PRINCIPLE

The idea of the project is to make the visually impaired people to independently reach the library and read the books. The image from the text is extracted using digital image processing. Extracted text is taken for text to speech conversion whose output is taken through the headset. Ultrasonic sensor is deployed to detect the obstacles on the way. This is connected to the processor circuit. This will be mounted on the walking stick which is portable and foldable. The software deployed in the raspberry pi processor has buttons for image to speech conversion and text to speech conversion using keyboard button for input.When the device is switched on the ulatrasonic sensor starts working whenever an obstacle is detected within 10 cm the device gives “Obstacle Detected”. Being designed for blind student to access his college library, the way to library signs are requested to be placed on the way. Sign recognition parallelly works with the ultrasonic sensor to identify the signs on the way.After reaching the library the reader button captures the image to be read, the whole page is captured and letters are extracted using optical character recognition operation. Voice output command button works with espeak software fot text to speech conversion operation.

8.1 OBJECT RECOGNITION

Scale-invariant feature transform (**SIFT**) is an algorithm in computer vision to detect and describe local features in images. SIFT keypoints of objects are first extracted from a set of reference images and stored in a database. An object is recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their feature vectors. From the full set of matches, subsets of keypoints that agree on the object and its location, scale, and orientation in the new image are identified to filter out good matches. The determination of consistent clusters is performed rapidly by using an efficient hash table implementation of the generalized Hough transform. Each cluster of 3 or more features that agree on an object and its pose is then subject to further detailed model verification and subsequently outliers are discarded. Finally the probability that a particular set of features indicates the presence of an object is computed, given the accuracy of fit and number of probable false matches. Object matches that pass all these tests can be identified as correct with high confidence.

ALGORITHM

There are mainly four steps involved in SIFT algorithm. Described one-by-one as follows.

8.1.1. Scale-space Extrema Detection

From the image above, it is obvious that we can’t use the same window to detect keypoints with different scale. It is OK with small corner. But to detect larger corners we need larger windows. For this, scale-space filtering is used. In it, Laplacian of Gaussian is found for the image with various values. LoG acts as a blob detector which detects blobs in various sizes due to change in . In short, acts as a scaling parameter. For eg, in the above image, gaussian kernel with low gives high value for small corner while guassian kernel with high fits well for larger corner. So, we can find the local maxima across the scale and space which gives us a list of values which means there is a potential keypoint at (x,y) at scale.

But this LoG is a little costly, so SIFT algorithm uses Difference of Gaussians which is an approximation of LoG. Difference of Gaussian is obtained as the difference of Gaussian blurring of an image with two different , let it be and . This process is done for different octaves of the image in Gaussian Pyramid. It is represented in below image:

Once this DoG are found, images are searched for local extrema over scale and space. For eg, one pixel in an image is compared with its 8 neighbours as well as 9 pixels in next scale and 9 pixels in previous scales. If it is a local extrema, it is a potential keypoint. It basically means that keypoint is best represented in that scale. It is shown in below image:

Regarding different parameters, the paper gives some empirical data which can be summarized as, number of octaves = 4, number of scale levels = 5, initial , etc as optimal values.

8.1.2. Keypoint Localization

Once potential keypoints locations are found, they have to be refined to get more accurate results. They used Taylor series expansion of scale space to get more accurate location of extrema, and if the intensity at this extrema is less than a threshold value (0.03 as per the paper), it is rejected. This threshold is called **contrastThreshold** in OpenCV

DoG has higher response for edges, so edges also need to be removed. For this, a concept similar to Harris corner detector is used. They used a 2x2 Hessian matrix (H) to compute the pricipal curvature. We know from Harris corner detector that for edges, one eigen value is larger than the other. So here they used a simple function, If this ratio is greater than a threshold, called **edgeThreshold** in OpenCV, that keypoint is discarded. It is given as 10 in paper.

So it eliminates any low-contrast keypoints and edge keypoints and what remains is strong interest points.

8.13. Orientation Assignment

Now an orientation is assigned to each keypoint to achieve invariance to image rotation. A neigbourhood is taken around the keypoint location depending on the scale, and the gradient magnitude and direction is calculated in that region. An orientation histogram with 36 bins covering 360 degrees is created. (It is weighted by gradient magnitude and gaussian-weighted circular window with equal to 1.5 times the scale of keypoint. The highest peak in the histogram is taken and any peak above 80% of it is also considered to calculate the orientation. It creates keypoints with same location and scale, but different directions. It contribute to stability of matching.

8.1.4. Keypoint Descriptor

Now keypoint descriptor is created. A 16x16 neighbourhood around the keypoint is taken. It is devided into 16 sub-blocks of 4x4 size. For each sub-block, 8 bin orientation histogram is created. So a total of 128 bin values are available. It is represented as a vector to form keypoint descriptor. In addition to this, several measures are taken to achieve robustness against illumination changes, rotation etc.

8.1.5. Keypoint Matching

Keypoints between two images are matched by identifying their nearest neighbours. But in some cases, the second closest-match may be very near to the first. It may happen due to noise or some other reasons. In that case, ratio of closest-distance to second-closest distance is taken. If it is greater than 0.8, they are rejected. It eliminaters around 90% of false matches while discards only 5% correct matches, as per the paper.

So this is a summary of SIFT algorithm. For more details and understanding, reading the original paper is highly recommended. Remember one thing, this algorithm is patented. So this algorithm is included in Non-free module in OpenCV.

8.1.6 SIFT in OpenCV

SIFT functionalities are available in OpenCV. Let’s start with keypoint detection and draw them. First we have to construct a SIFT object. We can pass different parameters to it which are optional and they are well explained in docs.

```
import cv2 import numpy as np img = cv2.imread('home.jpg') gray=
cv2.cvtColor(img,cv2.COLOR_BGR2GRAY) sift = cv2.SIFT() kp =
sift.detect(gray,None) img=cv2.drawKeypoints(gray,kp)
cv2.imwrite('sift_keypoints.jpg',img)
```

sift.detect() function finds the keypoint in the images. You can pass a mask if you want to search only a part of image. Each keypoint is a special structure which has many attributes like its (x,y) coordinates, size of the meaningful neighbourhood, angle which specifies its orientation, response that specifies strength of keypoints etc.

To calculate the descriptor, OpenCV provides two methods.

Since you already found keypoints, you can call **sift.compute()** which computes the descriptors from the keypoints we have found. Eg:kp,des = sift.compute(gray,kp)If you didn’t find keypoints, directly find keypoints and descriptors in a single step with the function,**sift.detectAndCompute()**.

We will see the second method:

sift = cv2.SIFT() kp, des = sift.detectAndCompute(gray,None)

Here kp will be a list of keypoints and des is a numpy array of shape .

So we got keypoints, descriptors etc. Now we want to see how to match keypoints in different images. This can be done through Flann Index Tree algorithm.

8.2 FLANN LIBRARY

An open source library named Fast Library for Approximate Nearest Neighbors is used in a large number of both research and industry projects and is widely used in the computer vision community, in part due to its inclusion in OpenCV , the popular open source computer vision library. FLANN also is used by other well known open source projects, such as the point cloud library (PCL) and the robot operating system (ROS) . FLANN has been packaged by most of the mainstream Linux distributions such as Debian, Ubuntu, Fedora, Arch, Gentoo and their derivatives.

For many computer vision and machine learning problems, large training sets are key for good performance. However, the most computationally expensive part of many computer vision and machine learning algorithms consists of finding nearest neighbor matches to high dimensional vectors that represent the training data. We propose new algorithms for approximate nearest neighbor matching and evaluate and compare them with previous algorithms. For matching high dimensional features, we find two algorithms to be the most efficient: the randomized k-d forest and the priority search k-means tree. There's a new algorithm for matching binary features by searching multiple hierarchical clustering trees and show it outperforms methods typically used in the literature.It shows that the optimal nearest neighbor algorithm and its parameters depend on the data set characteristics and describe an automated configuration procedure for finding the best algorithm to search a particular data set. In order to scale to very large data sets that would otherwise not fit in the memory of a single machine, we propose a distributed nearest neighbor matching framework that can be used with any of the algorithms described in the paper. All this research has been released as an open source library called fast library for approximate nearest neighbors (FLANN), which has been incorporated into OpenCV and is now one of the most popular libraries for nearest neighbor matching.

Nearest-neighbor search is a fundamental part of many computer vision algorithms and of significant importance in many other fields, so it has been widely studied. This section presents a review of previous work in this area.

8.2.1 Nearest Neighbor Matching Algorithms

We review the most widely used nearest neighbor techniques, classified in three categories: partitioning trees, hashing techniques and neighboring graph techniques.

8.2.2 Partitioning Trees

The kd-tree is one of the best known nearest neighbor algorithms. While very effective in low dimensionality spaces, its performance quickly decreases for high dimensional data. Arya et al. propose a variation of the k-d tree to be used for approximate search by considering (1+€)-approximate nearest neighbors, points for which dist(p,q)<(1+€)dist(p*,q) where p* is the true nearest neighbor. The authors also propose the use of a priority queue to speed up the search. This method of approximating the nearest neighbor search is also referred to as "error bound" approximate search. Another way of approximating the nearest neighbor search is by limiting the time spent during the search, or "time bound" approximate search. This method is proposed where the k-d tree search is stopped early after examining a fixed number of leaf nodes. In practice the time-constrained approximation criterion has been found to give better results than the error-constrained approximate search. Multiple randomized k-d trees are proposed as a means to speed up approximate nearest-neighbor search. In this we perform a wide range of comparisons showing that the multiple randomized trees are one of the most effective methods for matching high dimensional data. Variations of the k-d tree using non-axis-aligned partitioning hyperplanes have been proposed: the PCA-tree , the RP-tree , and the trinary projection tree . We have not found such algorithms to be more efficient than a randomized k-d tree decomposition, as the overhead of evaluating multiple dimensions during search outweighed the benefit of the better space decomposition. Another class of partitioning trees decompose the space using various clustering algorithms instead of using hyperplanes as in the case of the k-d tree and its variants. Both these methods are shown to be efficient at searching large data sets and they should be considered for further evaluation and possible incorporation into FLANN.

8.2.3 Hashing Based Nearest Neighbor Techniques

Perhaps the best known hashing based nearest neighbor technique is locality sensitive hashing (LSH) which uses a large number of hash functions with the property that the hashes of elements that are close to each other are also likely to be close. Variants of LSH such as multi-probe LSH improves the high storage costs by reducing the number of hash tables, and LSH Forest adapts better to the data without requiring hand tuning of parameters. The performance of hashing methods is highly dependent on the quality of the hashing functions they use and a large body of research has been targeted at improving hashing methods by using data-dependent hashing functions computed using various learning techniques: parameter sensitive hashing , spectral hashing , randomized LSH hashing from learned metrics , kernelized LSH , learnt binary embeddings , shift-invariant kernel hashing , semi-supervised hashing , optimized kernel hashing and complementary hashing . The different LSH algorithms provide theoretical guarantees on the search quality and have been successfully used in a number of project,show that in practice they are usually outperformed by algorithms using space partitioning structures such as the randomized k-d trees and the priority search k-means tree.

8.2.4 **FAST APPROXIMATE NN MATCHING**

Exact search is too costly for many applications, so this has generated interest in approximate nearest-neighbor search algorithms which return non-optimal neighbors in some cases, but can be orders of magnitude faster than exact search. After evaluating many different algorithms for approximate nearest neighbor search on data sets with a wide range of dimensionality , we have found that one of two algorithms gave the best performance: the priority search k-means tree or the multiple randomized k-d trees.

8.2.5 The Randomized k-d Tree Algorithm

The randomized k-d tree algorithm , is an approximate nearest neighbor search algorithm that builds multiple randomized k-d trees which are searched in parallel. The trees are built in a similar manner to the classic k-d tree , with the difference that where the classic kd-tree algorithm splits data on the dimension with the highest variance, for the randomized k-d trees the split dimension is chosen randomly from the top ND dimensions with the highest variance. We used the fixed value in our implementation, as this performs well across all our data sets and does not benefit significantly from further tuning. When searching the randomized k-d forest, a single priority queue is maintained across all the randomized trees. The priority queue is ordered by increasing distance to the decision boundary of each branch in the queue, so the search will explore first the closest leaves from all the trees. Once a data point has been examined (compared to the query point) inside a tree, it is marked in order to not be reexamined in another tree. The degree of approximation is determined by the maximum number of leaves to be visited (across all trees), returning the best nearest neighbor candidates found up to that point. It can be seen that the performance improves with the number of randomized trees up to a certain point (about 20 random trees in this case) and that increasing the number of random trees further leads to static or decreasing performance. The memory overhead of using multiple random trees increases linearly with the number of trees, so at some point the speedup may not justify the additional memory used. When the query point is close to one of the splitting hyperplanes, its nearest neighbor lies with almost equalprobability on either side of the hyperplane and if it lies on the opposite side of the splitting hyperplane, further exploration of the tree is required before the cell containing it will be visited. Using multiple random decompositions increases the probability that in one of them the query point and its nearest neighbor will be in the same cell.

8.2.6 **The Priority Search K-Means Tree Algorithm**

We have found the randomized k-d forest to be very effective in many situations, however on other data sets a different algorithm, the priority search k-means tree, has been more effective at finding approximate nearest neighbors, especially when a high precision is required. The priority search k-means tree tries to better exploit the natural structure existing in the data, by clustering the data points using the full distance across all dimensions, in contrast to the (randomized) k-d tree algorithm which only partitions the data based on one dimension at a time. Nearest-neighbor algorithms that use hierarchical partitioning schemes based on clustering the data points have been previously proposed in the literature [18], [19], [24]. These algorithms differ in the way they construct the partitioning tree (whether using k-means, agglomerative or some other form of clustering) and especially in the strategies used for exploring the hierarchical tree. We have developed an improved version that explores the k-means tree using a best-bin-first strategy, by analogy to what has been found to significantly improve the performance of the approximate kd-tree searches.

8.2.7 SCALING NEAREST NEIGHBOR SEARCH

Many researches have shown that using simple non-parametric methods in conjunction with large scale data sets can lead to very good recognition performance. Scaling to such large data sets is a difficult task, one of the main challenges being the impossibility of loading the data into the main memory of a single machine. F itting the data in memory is even more problematic for data sets of the size of those used. When dealing with such large amounts of data, possible solutions include performing some dimensionality reduction on the data, keeping the data on the disk and loading only parts of it in the main memory or distributing the data on several computers and using a distributed nearest neighbor search algorithm. Dimensionality reduction has been used in the literature with good results, however even with dimensionality reduction it can be challenging to fit the data in the memory of a single machine for very large data sets. Storing the data on the disk involves significant performance penalties due to the performance gap between memory and disk access times. In FLANN we used the approach of performing distributed nearest neighbor search across multiple machines.

8.2.8 FEATURE MATCHING USING FLANN ALGORITHM

FLANN stands for Fast Library for Approximate Nearest Neighbors contains a collection of algorithms optimized for fast nearest neighbor search in large datasets and for high dimensional features. It works more faster than BFMatcher for large datasets. We will see the second example with FLANN based matcher.

For FLANN based matcher, we need to pass two dictionaries which specifies the algorithm to be used, its related parameters etc. First one is IndexParams.

FeatureDetector_create() which creates detector and **DescriptorExtractor_ create()** which creates a descriptor to extract keypoints.

Import required modules

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

To Load Images:

```
img = cv2.imread("kpimg.png")
template = cv2.imread("kptemplate.png")
```

FIND KEYPOINTS:

```
detector = cv2.FeatureDetector_create("SIFT")
descriptor = cv2.DescriptorExtractor_create("SIFT")

skp = detector.detect(img)
skp, sd = descriptor.compute(img, skp)

tkp = detector.detect(template)
tkp, td = descriptor.compute(template, tkp)
```

skp is a list of all the keypoints found on the image. sd is the descriptor for the image.

tkp is a list of all the keypoints found on the template. td is the descriptor for the template.

Matching Keypoints using FLANN:

It was very easy to match keypoints in C++ using FlannMatcher, but it’s a bit difficult to that in Python.

```
    flann_params = dict(algorithm=1, trees=4)
    flann = cv2.flann_Index(sd, flann_params)
    idx, dist = flann.knnSearch(td, 1, params={})
    del flann
```

This method does a fast local approximate nearest neighbors (FLANN) calculation between two sets of feature vectors. The result are two numpy arrays the first one is a list of indexes of the matches and the second one is the match distance value. For the match indices or idx, the index values correspond to the values of td, and the value in the array is the index in td

i.e. j = idx[i] is where td[i] matches sd[j].

The second numpy array, at the index i is the match distance between td[i] and sd[j]. Lower distances mean better matches.

Perform the similar exercise on template to match keypoints of template to the image so that any erroneous keypoint match may be eliminated.

Initially insert the libraries which is just the numpy and the cv2 library

```
import cv2

import numpy as np
```

After adding this we need to add the SIFT/SURF feature extractor, which will extract some distinct features from images as key points for our object recognition and we will also need a feature matcher which will match the features from the sample/ training image with the current image from the webcam,

```
detector=cv2.SIFT()
FLANN_INDEX_KDITREE=0
flannParam=dict(algorithm=FLANN_INDEX_KDITREE,tree=5)
flann=cv2.FlannBasedMatcher(flannParam,{})
```

Now lets load the training image from the folder that we created earlier, and extract its features first,

```
trainImg=cv2.imread("TrainingData/TrainImg.jpg",0)
trainKP,trainDesc=detector.detectAndCompute(trainImg,None)
```

In the above code we used cv2.imread() to load the image which we saved earlier, and next we used the feature extractor to detect features and stored them in two variables one is trainKP which is the list of key points / coordinates of the features, and other in trainDesc which is list of descriptions of the corresponding key points

We will need these to to find visually similar objects in our live video,

Now lets initialize the camera of the VideoCapture object

```
cam=cv2.VideoCapture(0)
```

Main LOOP:

Matching the captured image with pre-defined images in library

```
while True:
    ret, QueryImgBGR=cam.read()
    QueryImg=cv2.cvtColor(QueryImgBGR,cv2.COLOR_BGR2GRAY)
    queryKP,queryDesc=detector.detectAndCompute(QueryImg,None)
    matches=flann.knnMatch(queryDesc,trainDesc,k=2)
```

In the above code, we first captured a frame from the camera, then converted it to gray scale, then we extracted the features like we did in the training image,after that we used the ***flann*** feature matcher to match the features in both images, and stored the matches results in ***matches*** variable

here flann is using knn to match the features with k=2, so we will get 2 neighbors

After this we have to filter the matches to avoid false matches

```
    goodMatch=[]
    for m,n in matches:
        if(m.distance<0.75*n.distance):
            goodMatch.append(m)
```

In the above code we created an empty list named ***goodMatch*** and the we are checking distance from the most neatest neighbor m and the next neatest neighbor n, and we are considering the match is a good match if the distance from point ***"m"*** is less the 70% of the distance on point "***n***" and appending that point to ***"goodMatch"***

We also need to make sure that we have enough feature matches to call these a match, for that we are going to set a threshold “***MIN_MATCH_COUNT***” and if the number of matches are greater than then value then only we are going to consider them as match

```
    MIN_MATCH_COUNT=30
    if(len(goodMatch)>=MIN_MATCH_COUNT):
        tp=[]
        qp=[]
        for m in goodMatch:
            tp.append(trainKP[m.trainIdx].pt)
            qp.append(queryKP[m.queryIdx].pt)
        tp,qp=np.float32((tp,qp))
        H,status=cv2.findHomography(tp,qp,cv2.RANSAC,3.0)
        h,w=trainImg.shape
        trainBorder=np.float32([[[0,0],[0,h-1],[w-1,h-1],[w-1,0]]])
        queryBorder=cv2.perspectiveTransform(trainBorder,H)
        cv2.polylines(QueryImgBGR,[np.int32(queryBorder)],True,(0,255,0),5)
    else:
 print "Not Enough match found-
%d/%d"%(len(goodMatch),MIN_MATCH_COUNT)
```

So in the above code we first check if number of matched features are more than the minimum number of threshold then we are going to do further operation, Now we created two pre defined images to get the coordinates of the matched features from the training image as well as from the query image, and converted that to numpy lists

We used cv2.findHomography(tp,qp,cv2.RANSAC,3.0) to find transformation constant to translate points from training points to query image points, Now we want to draw border around the object so we want to get the coordinates of the border corners from the training image, which are (0,0), (0,h-1), (w-1,h-1),(w-1,0) where h,w is the height and width of the training image

Now using the transformation constant "H" that we got from earlier we will translate the coordinates from training image to query image, Finally we are using "cv2.polylines()" to draw the borders in the query image

Lastly if the number of features are less that the minimum match counts then we are going to print it in the screen in the else part

Now lets display the image, and close the window if loop ends

```
    cv2.imshow('result',QueryImgBGR)
    if cv2.waitKey(10)==ord('q'):
        break
cam.release()
cv2.destroyAllWindows()
```

8.3 TEXT TO SPEECH CONVERSION

A computer system used to create artificial speech is called a speech synthesizer, and can be implemented in software or hardware products.

A text-to-speech (TTS) system converts normal language text into speech. In python we used Espeak synthesizer.

eSpeak is a compact open source software speech synthesizer for English and other languages, for Linux and Windows.

Features.

- Includes different Voices, whose characteristics can be altered.
- Can produce speech output as a WAV file.
- SSML (Speech Synthesis Markup Language) is supported (not complete), and also HTML.
- Compact size. The program and its data, including many languages, totals about 2 Mbytes.

8.4 ADVANTAGES AND APPLICATIONS

ADVANTAGE

- Portable
- Hand-held model.
- Assistive device for students to access library.
- Assistive reader for visually impaired

APPLICATIONS

- Read books without reader for visually impaired.
- Read magazines ,news papers and other printed text independently for visually impaired and aged people.
- Braille notations used as switches for compatibility of Visually impaired.

  Voice to text and keyboard button facilitates taking notes individually

CHAPTER 8
RESULT

There are four buttons each serving one purposes.

1.Power button:

While switching on the device the ultrasonic sensor starts working. Distance will be calculated and displayed on linux window. Whenever an object is detected within the specified range the “Object detected” speech output will be obtained

2.Object Recognition:

Camera will be enabled which prompts “Processing camera” on linux window.This recognizes the signs on the destination path and displays the corresponding command on linux window and similarly the speech output.

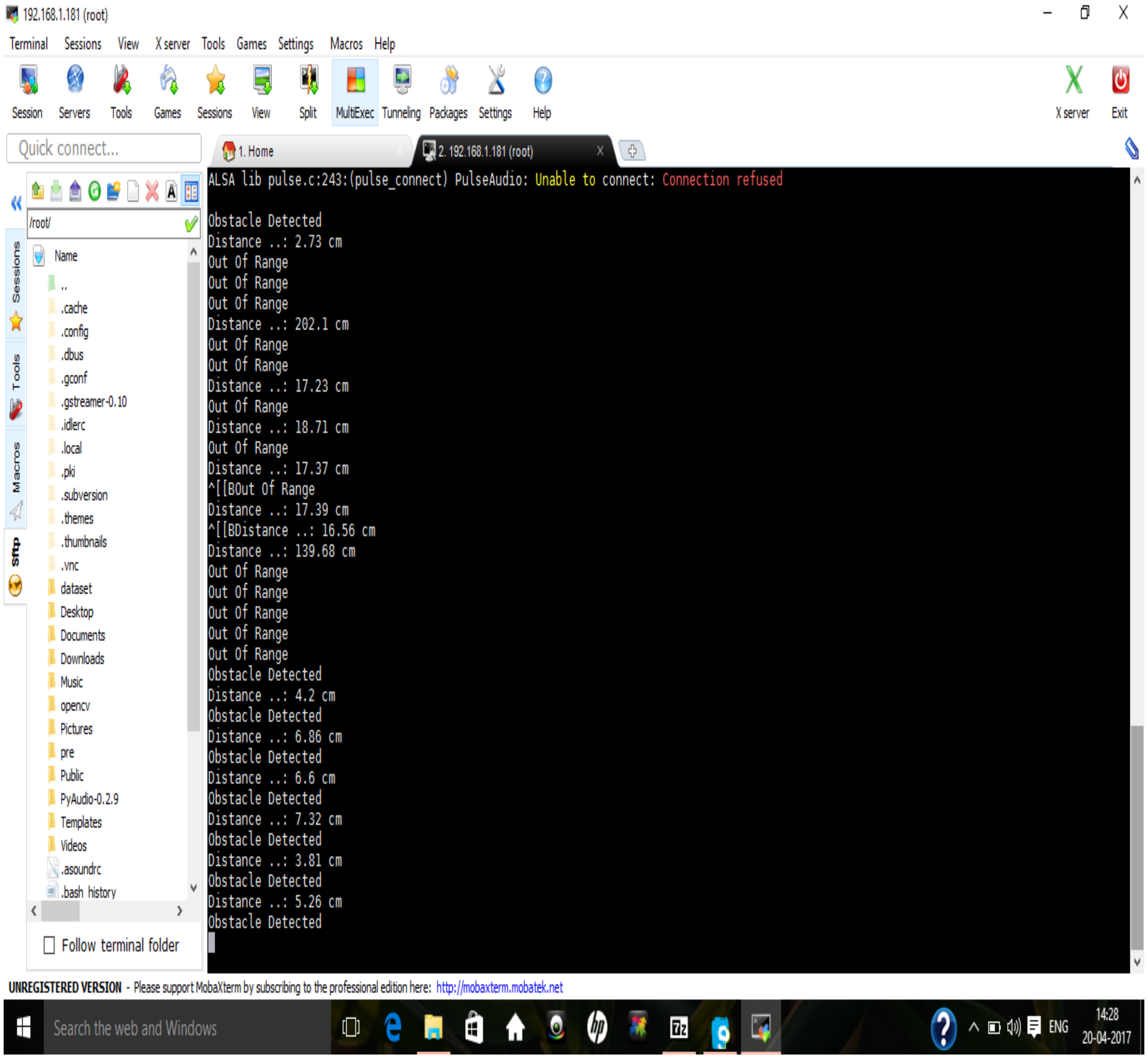
3. Image capturing:

Captures the image say the page of the book to be read. Extracted text is displayed as string on linux window.

4. Voice output:

Specially reads out the text extracted from the image captured.

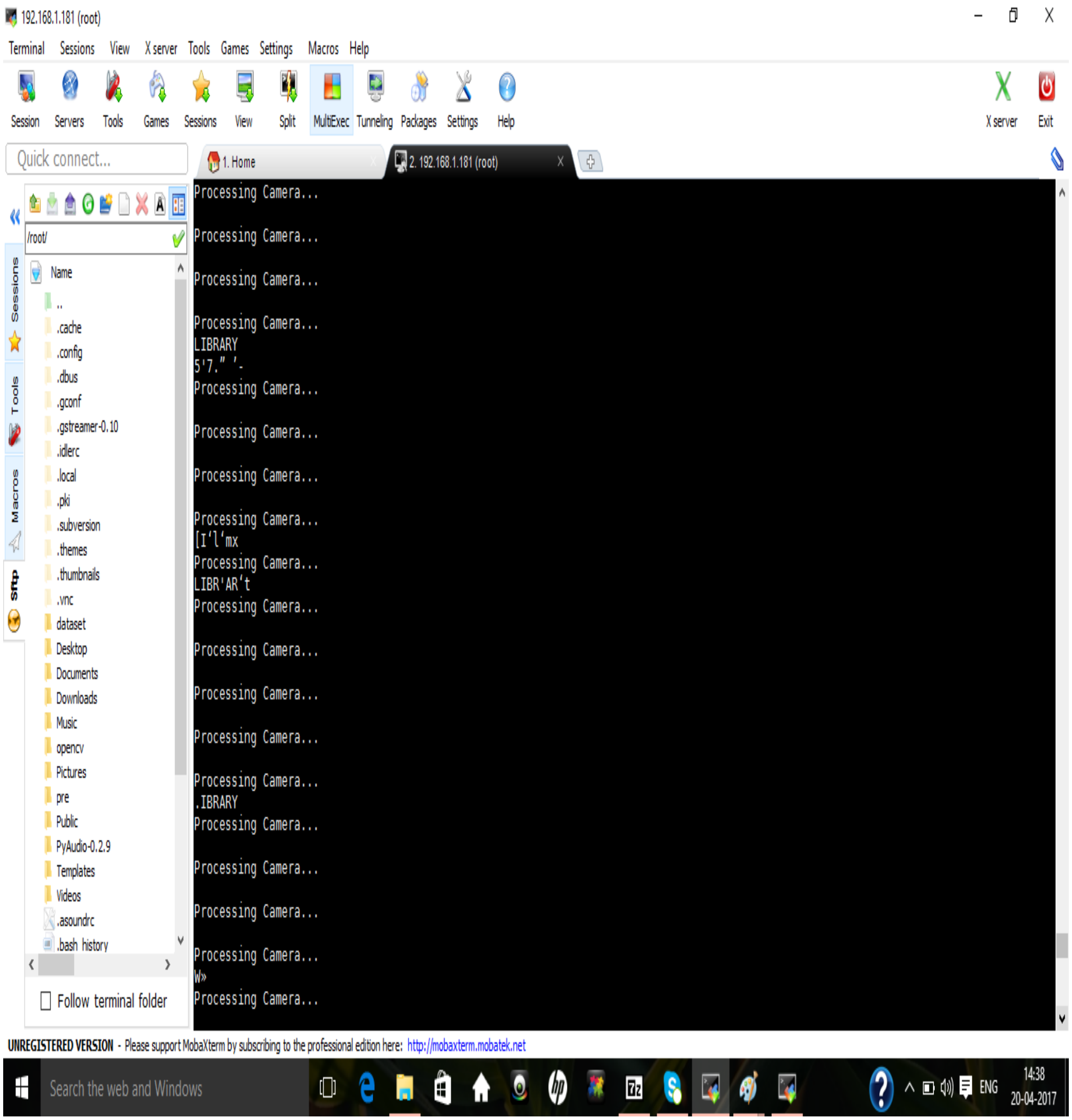
OBSTACLE DETECTION:



```
ALSA lib pulse.c:243:(pulse_connect) PulseAudio: Unable to connect: Connection refused

Obstacle Detected
Distance ..: 2.73 cm
Out Of Range
Out Of Range
Out Of Range
Distance ..: 202.1 cm
Out Of Range
Out Of Range
Distance ..: 17.23 cm
Out Of Range
Distance ..: 18.71 cm
Out Of Range
Distance ..: 17.37 cm
^[[BOut Of Range
Distance ..: 17.39 cm
^[[BDistance ..: 16.56 cm
Distance ..: 139.68 cm
Out Of Range
Out Of Range
Out Of Range
Out Of Range
Out Of Range
Obstacle Detected
Distance ..: 4.2 cm
Obstacle Detected
Distance ..: 6.86 cm
Obstacle Detected
Distance ..: 6.6 cm
Obstacle Detected
Distance ..: 7.32 cm
Obstacle Detected
Distance ..: 3.81 cm
Obstacle Detected
Distance ..: 5.26 cm
Obstacle Detected
```

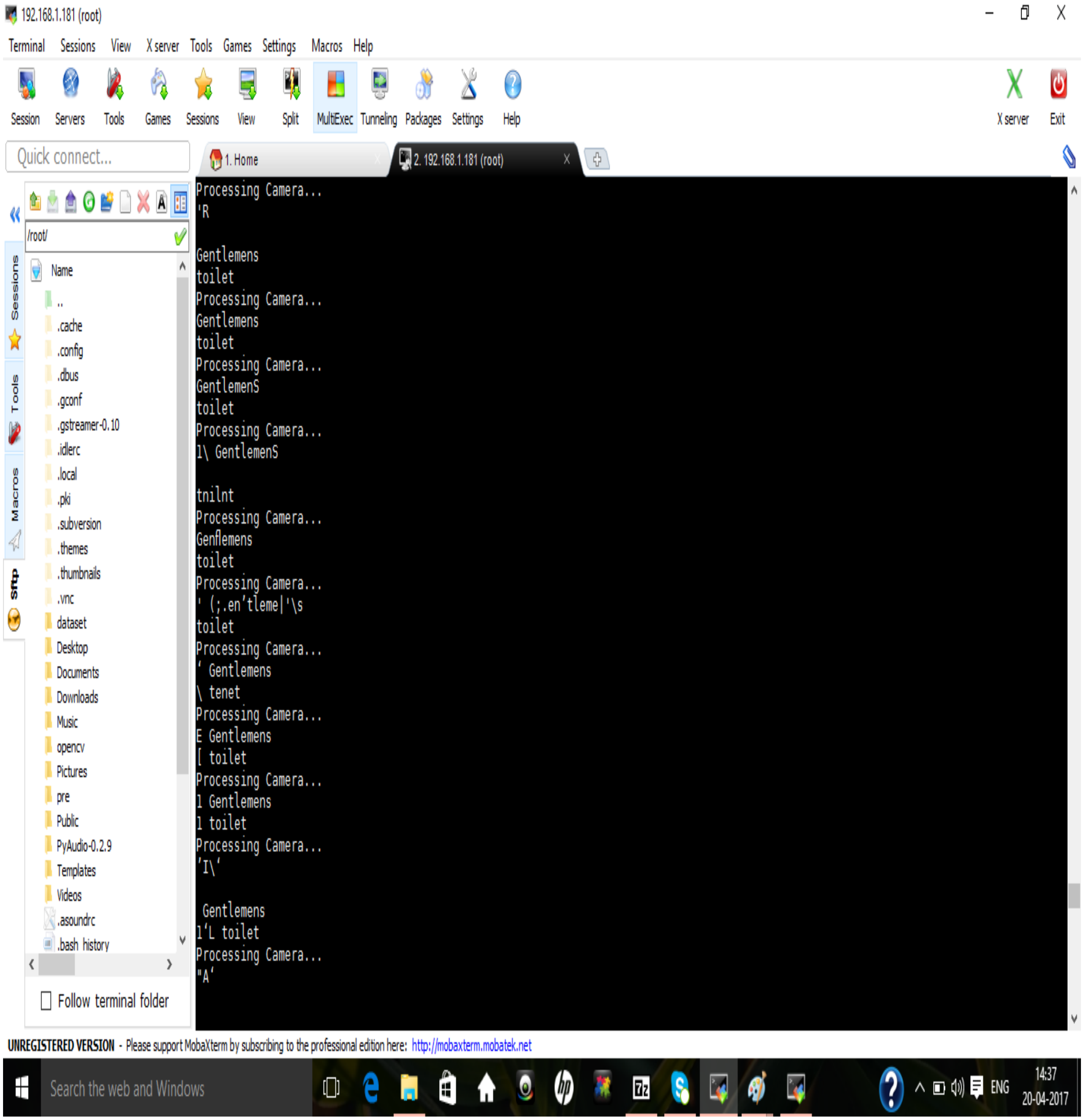
SIGN DETECTION:

LIBRARY



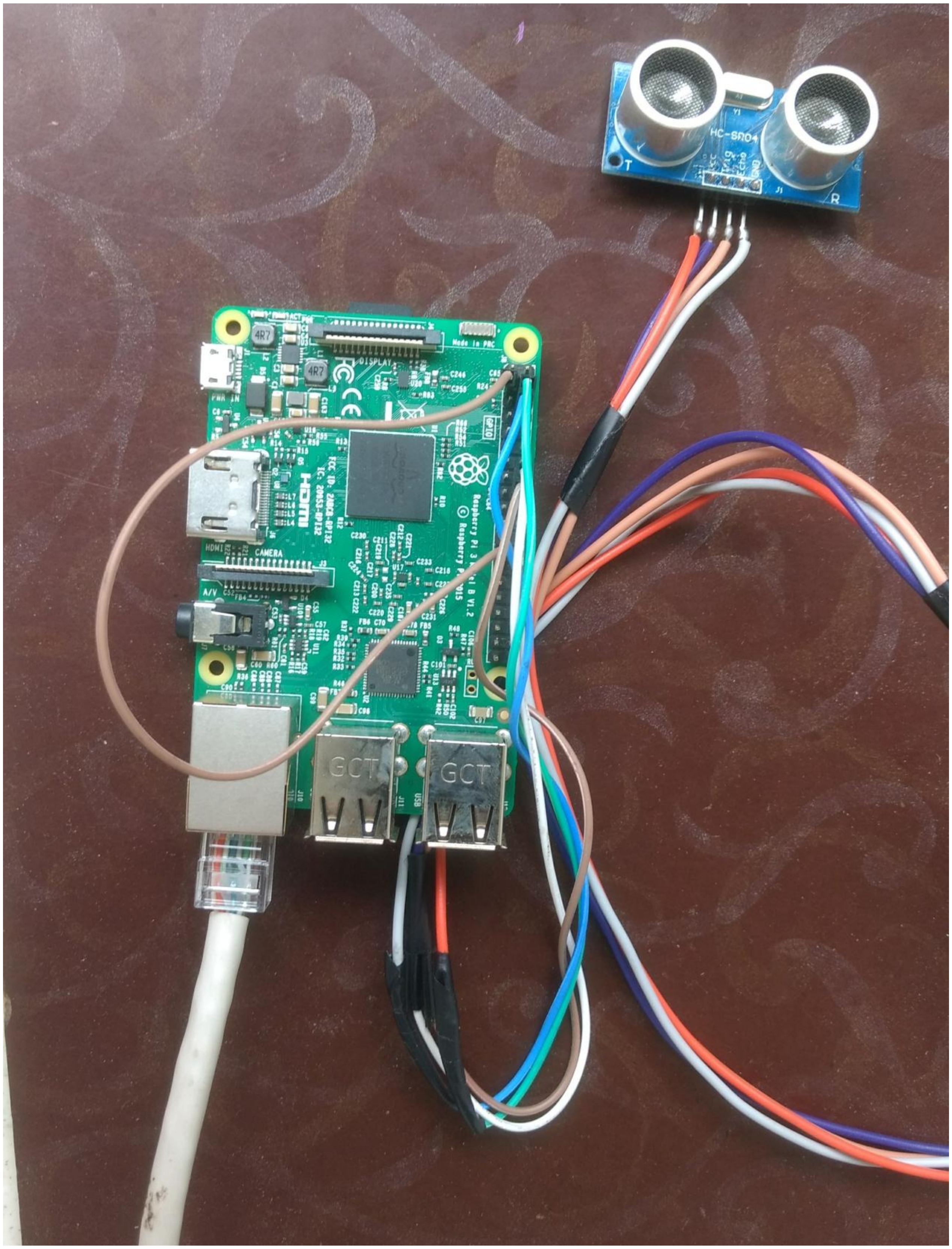
```
Processing Camera...

Processing Camera...

Processing Camera...

Processing Camera...
LIBRARY
5'7." '-
Processing Camera...

Processing Camera...

Processing Camera...

Processing Camera...
[I'l'mx
Processing Camera...
LIBR'AR't
Processing Camera...

Processing Camera...

Processing Camera...

Processing Camera...

Processing Camera...
.IBRARY
Processing Camera...

Processing Camera...

Processing Camera...

Processing Camera...
W»
Processing Camera...
```

GENTLE MENS TOILET



```
Processing Camera...
'R

Gentlemens
toilet
Processing Camera...
Gentlemens
toilet
Processing Camera...
GentlemenS
toilet
Processing Camera...
1\ GentlemenS

tnilnt
Processing Camera...
Genflemens
toilet
Processing Camera...
' (;.en'tleme|'\s
toilet
Processing Camera...
' Gentlemens
\ tenet
Processing Camera...
E Gentlemens
[ toilet
Processing Camera...
1 Gentlemens
1 toilet
Processing Camera...
'I\'

 Gentlemens
1'L toilet
Processing Camera...
"A'
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net


HC-SR04

CHAPTER 9
CONCLUSION

Thus the text reader device module developed helps both the low vision and vision less people to easily access the library with the indicator signs placed on the way. In this way they can access restrooms, living rooms and class rooms by pre defining the signs or room numbers. The voice stick can capture any printed image and the output will be obtained through headphones instantly. This project is rather a proposal of smart reading device which can be optimised with further features like face recognition, new currency recognition etc can be added under deep research and development.

CHAPTER 10

REFERENCE

1) Text to speech conversion technology.

   M.H.O'Malley, Berkeley speech technol.,

   CA,USA .

   http://ieeexplore.ieee.org/document/56867/

2) Text to speech conversion system for Brazilian

   Portuguese using a format based synthesis technique.

   L.De C.T. Gomes ,LPS,Campinas Univ.,Brazi8l.

   http://ieeexplore.ieee.org/document/713120/

3) D. Keating, M. A. Hersh, M. A. Johnson, Assistive technology for visually impaired and blind people, Berlin:Springer London, 2008.

4) "Visual impairment and blindness 2010", 2013, [online] Available: http://www.who.int/blindness/data_maps/VIFActsheetglodat2010full.pdf/