VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", BELAGAVI-590 018

A PROJECT REPORT

on

"ENHANCED BRAILLE DISPLAY"

submitted in partial fulfilment of the requirements for the award of the degree of

Bachelor of Engineering

in

Electronics and Instrumentation Engineering

Submitted by:

ABHAY BHARAJWAJ G	**1RN17EI001**
KARAN S	**1RN16EI019**
PRAJWALMANOJ	**1RN14EI029**
SIDDARAJ H M	**1RN16EI042**

Under the Guidance of

Dr MADHURA G

Asst. Professor Department of EIE

RNSIT, Bengaluru.

Department of Electronics and Instrumentation Engineering

RNS Institute of Technology

(AICTE Approved, VTU Affiliated and NAAC 'A' Accredited)

(UG programs – CSE, ECE, ISE, EIE and EEE have been Accredited by NBA for the Academic Years 2018 - 19, 2019 – 20, 2020 – 21and 2021-22)

Channasandra, Dr. Vishnuvardhan Road, Bengaluru – 560 098

2020 – 21

RNS Institute of Technology

Department of Electronics and Instrumentation Engineering

(AICTE Approved, VTU Affiliated and NAAC 'A' Accredited)
(UG programs – CSE, ECE, ISE, EIE and EEE have been Accredited by NBA
for the Academic Years 2018 - 19, 2019 - 20 ,2020 – 21and 2021-22)
Channasandra, Dr. Vishnuvardhan Road, Bengaluru – 560 098

2020-21

CERTIFICATE

Certified that the project work entitled **"ENHANCED BRAILLE DISPLAY"** has been successfully carried out by name (1RN17EI001,1RN16EI019,1RN14EI029,1RN16EI042), presently VIII semester students of **RNS Institute of Technology** in partial fulfilment of the requirements for the award of degree in **Bachelor of Engineering** in **Electronics and Instrumentation Engineering** of **Visvesvaraya Technological University, Belagavi** during academic year 2020-21. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in it and satisfies the academic requirements in respect of Project work Phase II for the said degree.

.....	.....	.....
Dr. Madhura G	**Dr. Andhe Pallavi**	**Dr. M K Venkatesha**
Designation Department of EIE RNSIT, Bengaluru	Professor & Head Department of EIE RNSIT, Bengaluru	Principal RNSIT, Bengaluru

External Viva

Name of the Examiners	Signature with Date
1……...	……
2……...	……

RNS Institute of Technology

(AICTE Approved, VTU Affiliated and NAAC 'A' Accredited)
(UG programs – CSE, ECE, ISE, EIE and EEE have been Accredited by NBA for the Academic Years 2018 - 19, 2019 – 20, 2020 – 21and 2021-22)
Channasandra, Dr. Vishnuvardhan Road, Bengaluru – 560 098

Department of Electronics and Instrumentation Engineering

2020-21

DECLARATION

We, ABAHY BHARADWAJ G, KARAN S, PRAJWAL MANOJ,SIDDARAJ H M students of VIII Semester BE, in Electronics and Instrumentation Engineering, RNS Institute of Technology hereby declare that the Project work entitled "ENHANCED BRAILLE DISPLAY" has been carried out by us and submitted in partial fulfillment of the requirements for the VIII Semester degree of Bachelor of Engineering in Electronics and Instrumentation Engineering of Visvesvaraya Technological University, Belagavi during academic year 2020-21.

Name	USN	Signature
1. ABHAY BHARADWAJ G	1RN17EI001	
2. KARAN S	1RN16EI019	
3. PRAJWAL MANOJ	1RN14EI029	
4. SIDDARAJ H M	1RN16EI042	

ABSTRACT

There are total 37 million blind people across the world. Out of that 15 million are from India. These people are unable to access the documents or electronics media which are not available in Braille scripts. Here we are implementing enhanced Braille blind people to read text or content. We scanned image form camera, processed that image by image processing techniques and same will be converted into text using OCR. The detected text will be given to raspberry pi which recognize every character and convert it into Braille code. With the help of solenoid, we are displaying that Braille code on Braille**.**

ACKNOWLEDGEMENT

All the achievements, does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. We would like to take this opportunity to express enough thanks to all of them.

We would like to profoundly thank the Management of RNS Institute of Technology for providing such a healthy environment for the pursuing our Project work.

We express our special thanks of gratitude to the Principal, **Dr. M K Venkatesha** for giving us the opportunity to embark upon this Project Work Phase II work continued encouragement. A special thanks to **Dr. Andhe Pallavi**, Prof. & HoD, Department of EIE, RNSIT, for her motivation and invaluable support well through the development of this project.

We would like to thank our project guide **Dr Madhura G,** Asst. Prof, Dept. of EIE, RNSIT for her constant guidance, support, endurance and constructive suggestions for the betterment of this project work.

We thank all teaching and the non-teaching staff of the department of Electronics and Instrumentation Engineering, for all the help they provided. None of this would have been possible without our parents; their encouragement assisted us to do this work; our heartfelt thanks to them. We would also like to thank our dear classmates for their support.

ABHAY BHARADWAJ G	**1RN17EI001**
KARAN S	**1RN16EI019**
PRAJWAL MANOJ	**1RN14EI029**
SIDDARAJ H M	**1RN16EI042**

Table of Contents

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

As per World Health Organization (WHO), out of the 6737.5 million of world population, 39.365 million people are blind, 246.024 million people are suffering from lowvision, and 285.389 million people are visually impaired. New growing development in computer vision, portable computer and digital camera make it practical to verify or check these individuals by camera-based product that merges existing OCR system with computer vision. A Braille Display is a touch sensation device that takes a pdf or normal text (.txt) file from a computer which is converted from image that has been captured and generates Braille dots for the people who have weak eyesight. Several types of actuators are employed by different types of Braille display devices. With the help of plunger movement in the coil, devices like solenoid prints out Braille.

Braille is a system that enables blind and visually impaired people to read and write through touch. It was devised by Louis Braille in 1821 and consists of raised dots arranged in "cells." A cell is made up of six dots that fit under the fingertips, arranged in two columns of three dots each. Each cell represents a letter, a word, a combination of letters, a numeral or a punctuation mark. The number and arrangement of these dots distinguish one character from another. The pattern of raised bumps or dots can be read with the fingers by blinds.

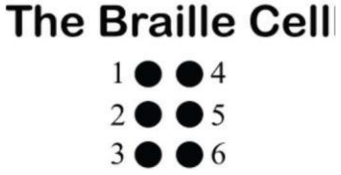


Fig 1.1 Braille Cell

The first ten letters of the alphabet are formed using the top four dots (1, 2, 4, 5). Adding a dot 3 makes the next ten letters and adding a dot 6 to that makes the last six letters (except "w" because it was not used very much in the French language at the time that Louis Braille devised this system). Punctuation is represented by its own unique set of dots, most often found in the lower part of the cell. In addition to the alphabet, the Braille Code includes many contractions, which are Braille cells that can stand for a combination of letters or entire words. Literary Braille numbers are formed by placing the Braille number sign (dots 3, 4, 5, and 6) before the Braille letters "a" through "j". There is also a code used for math and science notations called Nemeth.

The Braille Alphabet

a	b	c	d	e	f	g	h	i	j
k	l	m	n	o	p	q	r	s	t
u	v	w	x	y	z				

Common Punctuation Marks

,	;	:	.	!	()	? "	*	"	'	-

The Braille Numbers

1	2	3	4	5	6	7	8	9	0

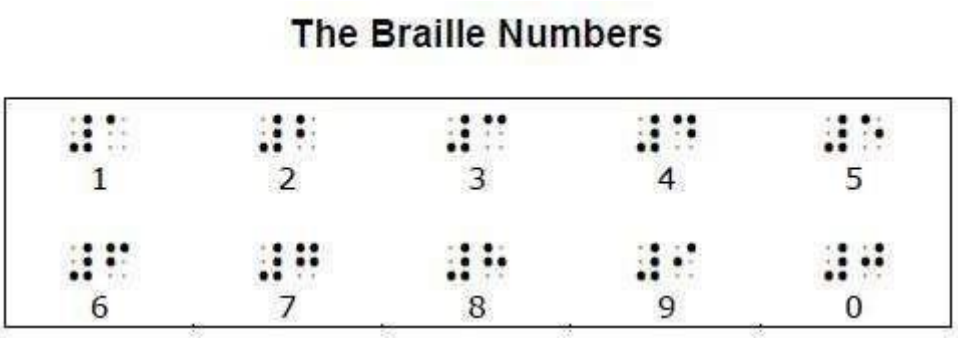


Fig 1.2 Braille numbers

1.2 MOTIVATION

Braille is vital for communication and education purposes for blind and visually challenged people. They face difficulties in interacting and gaining full advantage of computers.

Recently, and with the fast evolution in technology, researchers proposed to give the blinds the ability to take advantage of these advancements. Accordingly, designers and engineers started working on projects that relate input and output devices to the computers in order for the blind individual to have full control of the hi-tech machines. However, investments in these kinds of hardware presented complexity in the design, in addition to the high cost imposed by the devices used. In order to overcome the above said challenges our idea is to design a portable device called "Braille Display". The project's objective is to design and develop a Braille System output device for the visually impaired individuals that enable them to interact and communicate.

1.3 OBJECTIVE AND SCOPE

This project uses an algorithm which enables the user to convert the text that we normally have in our day to day usage into Braille Script and thus gives impetus for the visually impaired to read that text. The Product that we will create will be very intuitive and simplistic in design that will enable the end user to feel familiar and at home with the product. This project was conceived keeping in mind the day to day struggles in usage of laptops faced by the visually impaired people.

Braille Display is a device which helps the visually impaired to read a text file or access the internet. Braille is a tactile writing language of raised dots. It is developed for hap-tics perception, a combination of the sense of touch, movement and finger pressure. This product has got a plate which has holes. The text from the file or internet is converted to Braille and the keys present below the holes would pop up and down the hole based on the characters on the screen. These keys form a pattern of Braille alphabets which helps the visually impaired to read the text.

According to a recent survey by a national organisation for ophthalmologists India accounts for 20% of the total blind population of the world, with 7.8 million visually impaired out of the 39 million across the globe. India is now home to the world's largest number of blind people. 285 million people are estimated to be visually impaired worldwide: 39 million are blind and 246 have low vision. About 65 % of all people who are visually impaired are aged 50 and older, while this age group comprises about 20 % of the world's population. Thus, creating a low cost, intuitive text to braille converter is the main purpose of this paper.

CHAPTER 2

LITERATURE SURVEY

Enhanced Braille Display

Authors: Sangeeta Kumari, Akshay Akole, Pallavi Angnani, Yash Bhamare, Zaid Naikwadi

Publications: 2020 International Conference for Emerging Technology

Abstract:

Here we are implementing enhanced Braille system that helps blind people to read text or content. We scanned image form camera, processed that image by image processing techniques and same will be converted into text using OCR. The detected text will be given to raspberry pi which recognize every character and convert it into Braille code. With the help of solenoid, we are displaying that Braille code on Braille.

Tracing the effectiveness of braille reading patterns in individuals with blindness: Handedness and error analysis

Authors: Vassilios Papadimitriou and Vassilios Argyropoulos

Publications: British Journal of Visual Impairment

Abstract:

The main objective of the present study was to investigate the potential effects of handedness on braille reading patterns during braille text reading. Thirty-two Greek students (from Grades 3 to 12) with visual impairments, who used systematically the braille code as reading medium, participated in this study. Handedness was assessed through a modified version of the Edinburgh Handedness Inventory, while their reading level was estimated via a standardized test. In turn, participants read 18 texts, which were chosen randomly from their textbooks. Results indicated that handedness affected braille readers' selected reading patterns during text reading. A variety of reading patterns were recorded and the selected data were correlated with tactile reading strategies in terms of dominant hands and fingers. It seems that readers who selected one-hand braille reading patterns performed significantly more errors with the index of their dominant hand, whereas those who chose to read with both hands faced more difficulties toward the effective collaboration of the indices of their hands. Finally, the findings of the present study are discussed in relation to educational practice, relevant theory, and subsequent research.

Development of a Braille Display using Piezoelectric Linear Motors

Authors: Hyun-Cheol Cho, Byeong-Sang Kim, Jung-Jun Park, Jae-Bok Song

Publications: SICE-ICASE International Joint Conference

Abstract:

In this paper, the details of the proposed braille cell consisting of six piezoelectric linear motors are discussed. The electrical circuit to drive the motors is also introduced. Various tests have been conducted for seven visually impaired people. These tests for the prototype braille display system show that it can deliver the braille information which can be well recognized by visually impaired people.

Automatic system for text to Braille conversion

Authors: Adrian MOISE

Publications: International Conference

Abstract:

The authors of this paper present the development of an automatic system used to convert computer written text to the Braille language. The system uses a microcontroller connected to a special device that can be read by blind persons. For this system, a software based concept to implementing Finite State Machines (FSM) has been developed. Experimental results are shown and discussed.

Development of a text to braille interpreter for printed documents through optical image processing

Authors: Joshua L. Dela Cruz, Jonaida Angela D. Ebreo, Reniel Allan John P. Inovejas

Publications: IEEE

Abstract:

This paper presents the development of an optical text to braille converter device for aiding visually impaired individuals to read printed materials. This is a solution for the lag or even failure of translating or printing the braille version of everyday reading materials. The system utilized optical character recognition engine in which an image of the text to be translated into braille is captured. The digitized texts are then transferred electronically in a braille haptic device. This device are piezoelectric based haptic system which is composed of several haptic pins arranged in a way to resemble the braille writing system. Several experiments were conducted to determine the performance of the system. The overall system reliability obtained was 95.68%. The system is also capable of processing speed of 1 word in 2 seconds. The system performs at its best with a letter sized page reading material within the range of 15 to 20 cm from the camera, with the camera positioned at 0 degrees.

CHAPTER 3

REQUIREMENTS & SPECIFICATIONS

3.1 Functional Requirements

The functional requirements for a system describe what the system should do. These requirements depend on the type of software being developed, the general approach taken by the organization when writing requirements. The functional system requirements describe the system function in detail, its inputs and outputs, exceptions and so on.

Functional requirements are as follows:

- Raspberry Pi
- Web camera
- Keypad

3.2 Non-Functional Requirements

Non-functional requirements, as the name suggests, are requirements that are not directly concerned with the specific functions delivered by the system. They may relate to emergent system properties such as reliability, response time and store occupancy. Alternatively, they may define constraints on the system such as capabilities of I/O devices and the data representations used in system interfaces.

Easy to Operate : The captured image is converted to text using Tesseract OCR and save the text to file out1.txt.Open the text file and split the paragraph into sentences and save it. In OCR, the adaptive thresholding techniques are used to change the image into binary images and the are transferred to character outlines. The converted text is given to the Braille.

Portability: The important key factor of this project to facilitate these people and to fix them more confident to manage their sites by themselves. The primary advantage is that the device can be taken away easily and is of about less weight.

Cost Effective: Use of simple hardware modules makes the device cost effective.

Platform : Use of sensors makes the system useful in all environments

3.3 Hardware Requirements:

1. Raspberry Pi
2. Camera
3. SD Card
4. Braille Keypad
5. Monitor

3.4 Software Requirements:

1. Open CV
2. Python

Raspberry pi

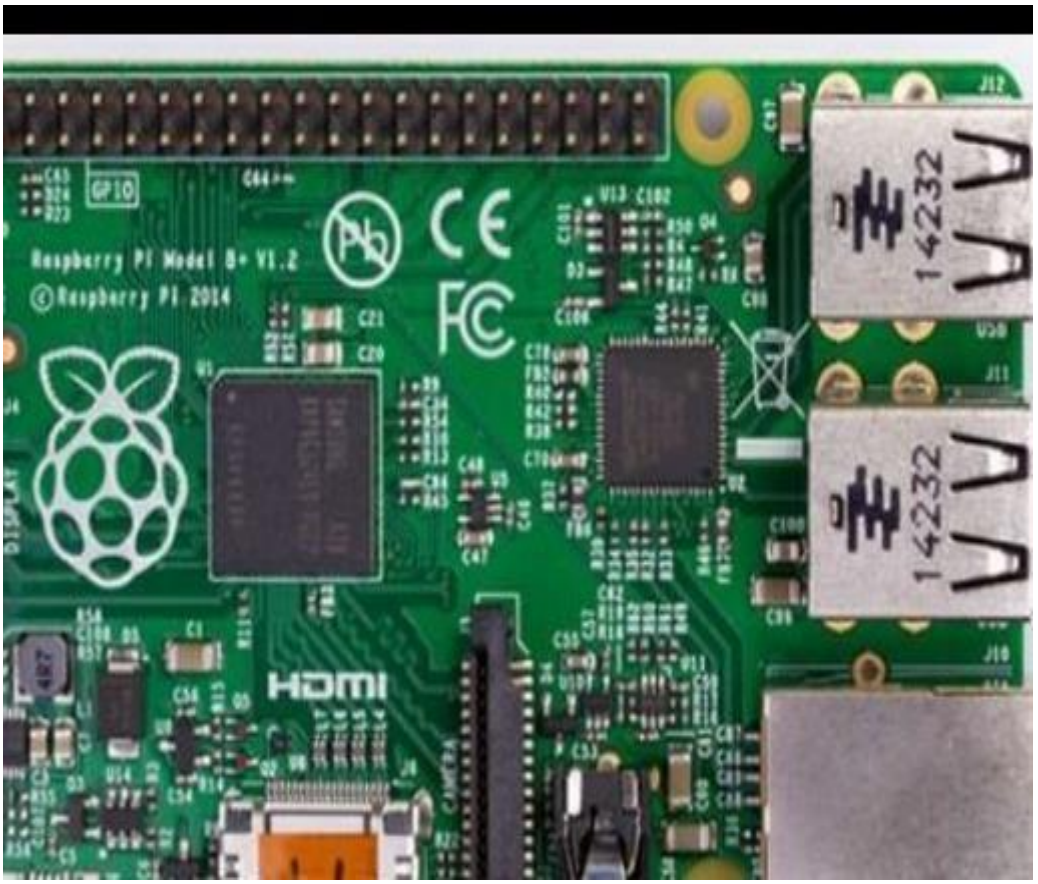


Fig 3.1 Raspberry PI

Low cost high performance computer which can be plugged in TV and monitor and can be used as computer which is very small as credit card.

- Its CPU is 700Mhz single core ARM1176JZF-S,
- It has 4 USB ports
- It has dual core video core iv multimedia coprocessor
- Size of its RAM is 512mb
- It has micro SDHC plot for storage
- Power rating of raspberry pi is 600mA i.e, 3.0W
- It has 17*GPIO plus the same specific functions

This raspberry pi works as the computer of the smart walking stick [4]. Raspberry Pi is a credit card sized single board, low cost computer [11]. It takes input from the GPIO pins, which can be attached to LEDs, switches, analog signals and other devices. For our proposed design, we connect the GPIO pins to the ultrasonic sensors. It requires a power source of 5V to be operational and we have to insert a Micro SD memory card in it, which acts as its permanent memory. For our design Raspberry Pi 1 Model B+ is used. It contains 4 USB ports, a HDMI port, an audio jack port and an Ethernet port. The Ethernet port helps the device connect to the Internet and install required driver APIs. It has a 700 MHz single core processor and supports programming languages such as Python, Java, C, and C++ etc. This mini computer runs our algorithm, which helps to calculate the distance from the obstacle based on the input it receives from the sensors. Then a Text-to-Speech driver API [12] is used to convert the text message (distance) to speech, which is relayed to the person wearing the earphone

F. GPIO pins

The raspberry pi board has 17 GPIO ins in it. These GPIO pins provide ability to connect directly to electronic devices. The inputs will be like sensors, buttons or other communication with chips or modules using low level protocols SPI and serial UART connections. It uses 3.3V logic levels. No analog input or output is available in this GPIO pins but we can use external chords for this analog connection. The above block diagram represents the working of the raspberry pi. Many inputs such as ultrasonic sensors, switch input and camera are given to the raspberry pi board through the GPIO pins.

Power Supply

There are many types of power supply. Most are designed to convert high voltage AC mains electricity to a suitable DC voltage supply for electronic circuits and other devices. A power supply can by broken down into a series of blocks, each of which performs a particular function. DC voltages are required to operate various electronic equipment . These voltages are 5V, 9V or 12V which cannot be obtained directly. Thus the input to the circuit is applied from the regulated power supply. A power supply takes the AC from the wall outlet, converts it to unregulated DC, and reduces the voltage using an input power transformer, typically stepping it down to the voltage required by the load. For safety reasons, the transformer also separates the output power supply from the mains input.

A power supply is an electrical device that supplies electric power to an electrical load. The primary function of a power supply is to convert electric current from a source to the correct voltage, current, and frequency to power the load. As a result, power supplies are sometimes referred to as electric power converters. Some power supplies are separate standalone pieces of equipment, while others are built into the load appliances that they power. Examples of the latter include power supplies found in desktop computers and consumer electronics devices. Other functions that power supplies may perform include limiting the current drawn by the load to safe levels, shutting off the current in the event of an electrical fault, power conditioning to prevent electronic noise or voltage surges on the input from reaching the load, power-factor correction, and storing energy so it can continue to power the load in the event of a temporary interruption in the source power.

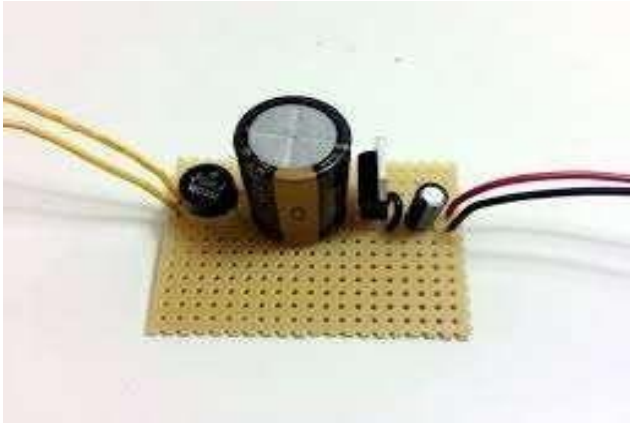


Fig 3.2 Power Supply

Camera

USB Cameras are imaging cameras that use USB 2.0 or USB 3.0 technology to transfer image data. USB Cameras are designed to easily interface with dedicated computer systems by using the same USB technology that is found on most computers. A camera is an optical instrument to capture still images or to record moving images, which are stored in a physical medium such as in a digital system or on photographic film. A camera consists of a lens which focuses light from the scene, and a camera body which holds the image capture mechanism. The still image camera is the main instrument in the art of photography and captured images may be reproduced later as a part of the process of photography, digital imaging, photographic printing. The accessibility of USB technology in computer systems as well as the 480 Mb/s transfer rate of USB 2.0 makes USB Cameras ideal for many imaging applications. An increasing selection of USB 3.0 Cameras is also available with data transfer rates of up to 5 Gb/s.

Fig 3.3 Camera Module

Software:

Tesseract OCR:

Python Tesseract is an optical character recognition (OCR) engine for various OS. Tesseract OCR is the process of electronically extracting1text from images and1reusing it in a variety of ways1such as document1editing, 1free-text1searches. OCR is a technology that is capable converting documents such as scanned papers, PDF files and captured image into editable data. Tesseract can be used for Linux, Windows and Mac OS. It can be used by programmers to extract typed, printed text from images using an API. Tesseract can use GUI from available 3rd party page. The installation process of tesseract OCR is a combination of two parts-The engine and training data for a language. For Linux OS. Tesseract was in the top three OCR engines in terms of character accuracy in
1995. It is available for Linux, Windows and Mac OS X. However, due to limited resources it is only rigorously tested by developers under Windows andUbuntu.
Tesseract up to and including version 2 could only accept TIFF images of simple one-column text as inputs. These early versions did not include layout analysis, and so inputting multi-columned text, images, or equations produced garbled output. Since version 3.00 Tesseract has supported output text formatting, hOCR positional
information and page-layout analysis. Support for a number of new image formats was added using the Leptonica library. Tesseract can detect whether text is monospaced or proportionally spaced.
Tesseract can process right-to-left text such as Arabic or Hebrew, many Indic scripts as well as CJK quite well. Accuracy rates are shown in this presentation for Tesseract tutorial at DAS 2016, Santorini by Ray Smith. Tesseract is suitable for use as a backend and can be used for more complicated OCR tasks including layout analysis by using a frontend such as OCRopus. Tesseract can be obtained directly from many Linux distributers. The latest stable version of tesseract OCR is 3.05.00. In our project Tesseract is used to convert the captured image text into text format. Tesseract Features: 1) Page layout analysis. 2) supported. 3) Improve forecast accuracy. 4) Add UI.

Open CV

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is crossplatform and free for use under the open-source BSD license. OpenCV supports deep learning frameworks TensorFlow, Torch/PyTorch and Cafe. It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation. Wrappers in other languages such as C#, Perl, Ch, Haskell and Ruby have been developed to encourage adoption by a wider audience. Since version 3.4, OpenCV.js is a JavaScript binding for selected subset of OpenCV functions

for the web platform. All of the new developments and algorithms in OpenCV are now developed in the C++ interface. If the library finds Intel's Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate itself. An OpenCL-based GPU interface has been in progress since October 2012, documentation for version 2.4.13.3 can be found at docs.opencv.org.

CHAPTER 4

SYSTEM DESIGN

4.1 System Architecture:

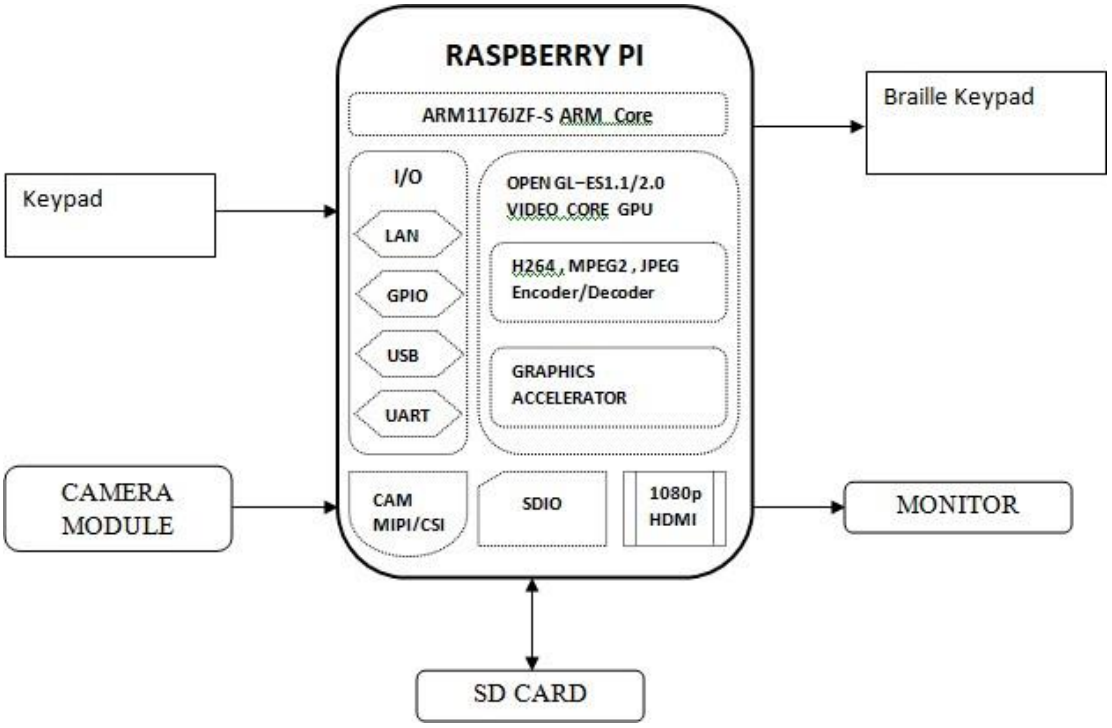


Fig 4.1 Architecture

4.2 System Description:

The boards functions as:

a) To produce Braille text, Raspberry pi receives text from the text file.

b) Send them to the control board via serial interface.

c) Raspberry pi then converts the ASCII characters into Braille symbol and then equivalent servo control signals to actuate the Braille pins.

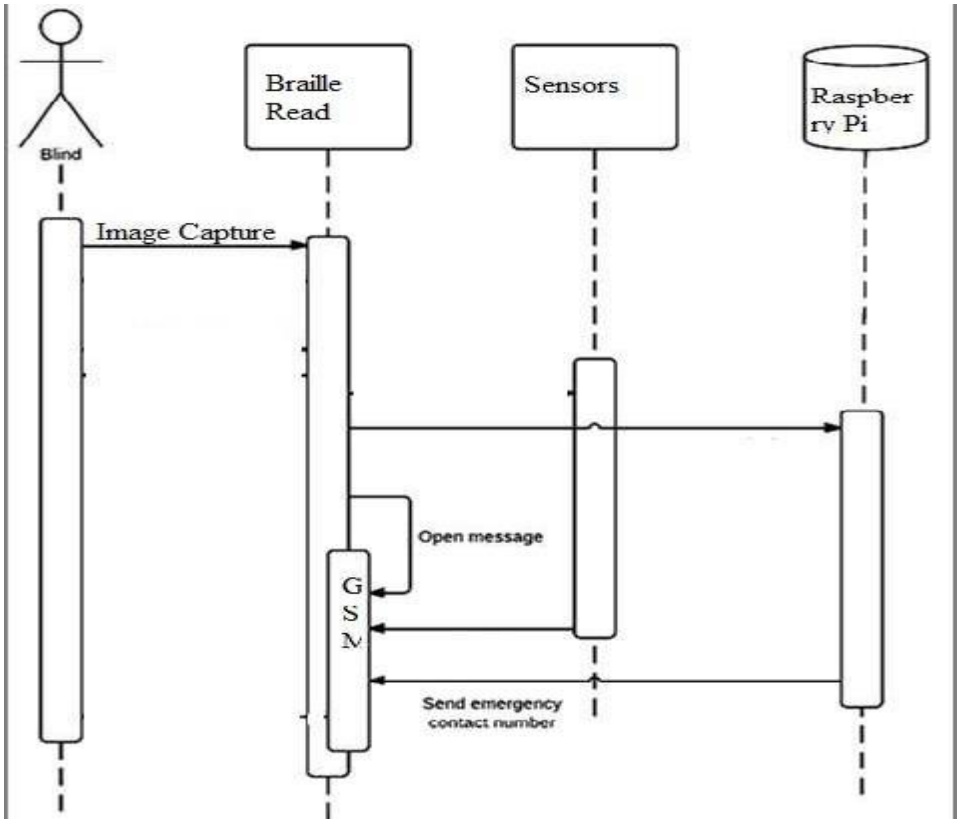
d) The blinds can feel the sense of touch of the Braille pins that are popped up according to the input letter and they recognize the letter accordingly.

The above control mechanism is been programmed in Python language in Raspbian operating system.

Braille text popper

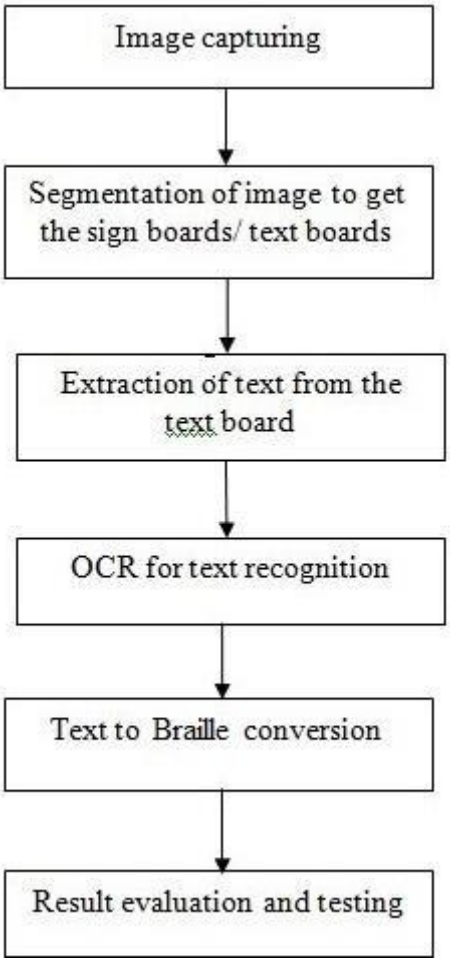
Braille text popper is a key setup which has a solenoid/ servo motors below each key. Each servo motor/ solenoid is responsible for lowering and raising a pin, which will emerge through perforations on the top plate to form a Braille dot. The perforations serve as guides for the pins and form six dots which is equivalent to one Braille cell.

Sequence Diagram:



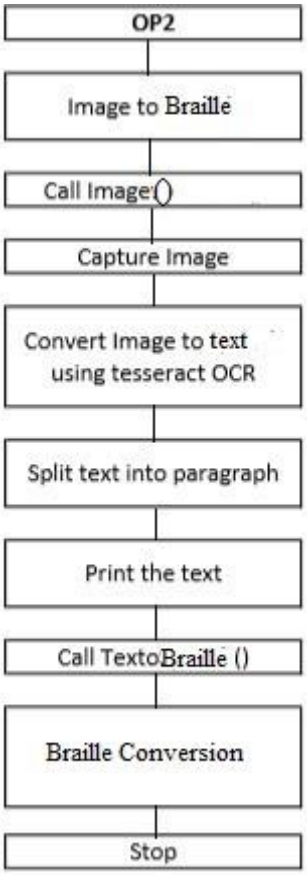
Sequence Diagram explains the sequence of a particular operation that takes place. As we can see in the diagram the blind would Want to read a particular text. The process begins with capturing the image the blind wants to read, we make use of the text read application to read the text from the image, which is then processed to the sensors and which the help of raspberry pi we convert the text retrieved to braille which can be given as the output.

Flow Chart



It basically describes the sequences of operations that takes places with respect to the project. Our project striking feature would be the image capturing for reading a particular text. The flowchart starts with capturing an image that the blind wants to read. Segmentation of the image takes place in order to retrieve the text from the image, which then makes use of the sign board to get the text identified. Extraction of the text from the text board converts the image to the text that the blind will want to read. We make use of OCR for the text recognition. This text is then converted into Braille which helps the blind person to read anything he desires. This feature keeps the blind informed with respect to the surrounding.

Data flow Diagram



A Data-flow diagram (DFD) is a way of representing a flow of a data of a process or a system (usually an information system) The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart. There are several notations for displaying data-flow diagrams. The notation presented above was described in 1979 by Tom DeMarco as part of Structured Analysis. For each data flow, at least one of the endpoints (source and / or destination) must exist in a process. The refined representation of a process can be done in another data-flow diagram, which subdivides this process into sub-processes. The data-flow diagram is part of the structured-analysis modelling tools. When using UML, the activity diagram typically takes over the role of the data-flow diagram. A special form of data-flow plan is a site-oriented data-flow plan. Data-flow diagrams can be regarded as inverted Petri nets, because places in such networks correspond to the semantics of data memories. Analogously, the semantics of transitions from Petri nets and data flows and functions from data-flow diagrams should be considered equivalent.

Data flow (flow, dataflow) shows the transfer of information (sometimes also material) from one part of the system to another. The symbol of the flow is the arrow.

The flow should have a name that determines what information (or what material) is being moved. Exceptions are flows where it is clear what information is transferred through the entities that are linked to these flows. Material shifts are modeled in systems that are not merely informative. Flow should only transmit one type of information (material). The arrow shows the flow direction (it can also be bi- directional if the information to/from the entity is logically dependent - e.g. question and answer). Flows link processes, warehouses and terminators.

CHAPTER 5

IMAGE PROCESSING

IMAGE PROCESSING:

Introduction to image processing

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image. Nowadays, image processing is among rapidly growing technologies. It forms core research area within engineering and computer science disciplines too.

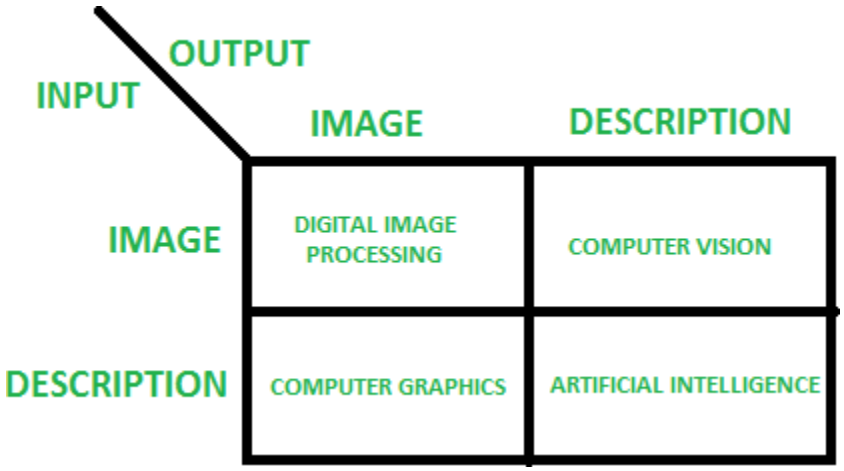
Image processing basically includes the following three steps:

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

There are two types of methods used for image processing namely, analogue and digital image processing. Analogue image processing can be used for the hard copies like printouts and photographs. Image analysts use various fundamentals of interpretation while using these visual techniques. Digital image processing techniques help in manipulation of the digital images by using computers. The three general phases that all types of data have to undergo while using digital technique are pre-processing, enhancement, and display, information extraction.

In this project we will talk about a few fundamental definitions such as image, digital image, and digital image processing. Different sources of digital images will be discussed and examples for each source will be provided. The continuum from image processing to computer vision will be covered in this lecture. Finally we will talk about image acquisition

and different types of image sensors.



Digital image processing consists of the manipulation of images using digital computers. Its use has been increasing exponentially in the last decades. Its applications range from medicine to entertainment, passing by geological processing and remote sensing. Multimedia systems, one of the pillars of the modern information society, rely heavily on digital image processing.

The discipline of digital image processing is a vast one, encompassing digital signal processing techniques as well as techniques that are specific to images. An image can be regarded as a function $f(x, y)$ of two continuous variables x and y. To be processed digitally, it has to be **sampled** and transformed into a matrix of numbers. Since a computer represents the numbers using finite precision, these numbers have to be **quantized** to be represented digitally. Digital image processing consists of the manipulation of those finite precision numbers. The processing of digital images can be divided into several classes: **image enhancement, image restoration, image analysis,** and image compression. In image enhancement, an image is manipulated, mostly by heuristic techniques, so that a human viewer can extract useful information from it. Image restoration techniques aim at processing corrupted images from which there is a statistical or mathematical description of the degradation so that it can be reverted. Image analysis techniques permit that an image be processed so that information can be automatically extracted from it. Examples of image

analysis are image segmentation, edge extraction, and texture and motion analysis. An important characteristic of images is the huge amount of information required to represent them. Even a gray-scale image of moderate resolution, say 512 × 512, needs 512 × 512 × 8 ≈ 2×10^6 bits for its representation. Therefore, to be practical to store and transmit digital images, one needs to perform some sort of image compression, whereby the redundancy of the images is exploited for reducing the number of bits needed in their representation.

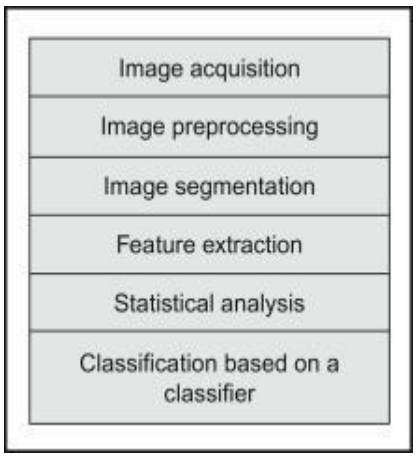


Fig. 5.1: Image processing techniques

Advantages:

- It removes noise from the digital image.
- It has ability to manipulate the pixel shades to correct image density and contrast.
- Images can be stored in a computer memory and easily retrieved on the same computer screen.
- Digital imaging allows the electronic transmission of images to third party providers.

Applications:

- Medical applications.
- Restoration and enhancement.
- Digital cinema.
- Image transmission and coding.

- Processing of colour images.
- Image and video processing architectures.

How Image processing is used in our project:

Image processing is a technology to perform the operations on image, in order to get the enhanced image and to extract the useful information from it. It basically includes the three steps they are

- Importing the image via image acquisition tools
- Analysing and manipulating the image
- Output in which result can be altered image.

There are two types of image processing, they are analogue image processing and digital image processing. In analogue image processing photographs and printouts are used as an image. Digital image processing technique help in manipulation of the digital images by using computers. The digital techniques are used to undergo all types of data, they are pre-processing, enhancement and display, information extraction. Image processing technique can be obtained by using various programming language. In our project we are using python programming language in image processing technique, since it is easy to program and flexible. The installed system should be safe and to make it safe we are using several sensors to protect the system. We are using health monitoring sensors to detect the health condition of the child. Whenever the installed system is touched the camera gets on, there we are using image processing. That it pre-process the image captured by the webcam in the laptop by improving the image data that suppresses unwilling distortions or enhances some image features important for the further processing. After the pre-processing of the image the next technique is the image enhancement, it improves the quality of the image and information content of original data before processing. Once the enhancement of the image is done the next technique is to display the image. After enhancement of the image the webcam displays the image in the screen.

IMPLEMENTATION

Implementation is the important phase where the development of the proposed system is based on the decisions made previously in the design and system requirement phase. Selecting the platform to implement the system and the guidelines to develop the code are also discussed in this section. The decisions made on the selection of the languages, code and other aspects are based on the environment the system works on. implementation refers to post-sales process of guiding a client from purchase to use of the software or hardware that was purchased. This includes requirements analysis, scope analysis, customizations, systems integrations, user policies, user training and delivery. These steps are often overseen by a project manager using project management methodologies. Software Implementations involve several professionals that are relatively new to the knowledge based economy such as business analysts, technical analysts, solutions architects, and project managers. To implement a system successfully, a large number of inter-related tasks need to be carried out in an appropriate sequence. Utilizing a well- proven implementation methodology and enlisting professional advice can help but often it is the number of tasks, poor planning and inadequate resourcing that causes problems with an implementation project, rather than any of the tasks being particularly difficult. Similarly, with the cultural issues it is often the lack of adequate consultation and two- way communication that inhibits achievement of the desired results. Usually, the implementation consists of the following steps:

- Planning
- Investigating the present system and the requirements on implementations.
- Providing training for the users about the newly developed system.

5.1 Implementation

This is the phase where the theoretical system designed is turned into the actual working system. Thus, this phase is considered as the trivial phase that yields the required results making the users confident enough about the system to use effectively. As mentioned in the above section, the steps of the implementation phase are designed with care to achieve the expected results. There are three major decisions made on the project before implementation, they are as follows:

- Platform selection.
- Selecting programming language.
- Coding guidelines.

5.2 Selecting Platform

The platform is the one on which a program actually runs. Most platforms are a combination of the OS and the hardware. The details of the platforms are as follows:

5.2.1 Raspbian OS

Although the Raspberry Pi's operating system is closer to the Mac than Windows, it's the latter that the desktop most closely resembles It might seem a little alien at first glance, but using Raspbian is hardly any different to using Windows (barring Windows 8 of course). There's a menu bar, a web browser, a file manager and no shortage of desktop shortcuts of pre-installed applications. Raspbian is an unofficial port of Debian Wheezy armhf with compilation settings adjusted to produce optimized "hard float" code that will run on the Raspberry Pi. This provides significantly faster performance for applications that make heavy use of floating point arithmetic operations.

Although Raspbian is primarily the efforts of Mike Thompson (mpthompson) and Peter Green (plugwash), it has also benefited greatly from the enthusiastic support of Raspberry Pi community members who wish to get the maximum performance from their device.

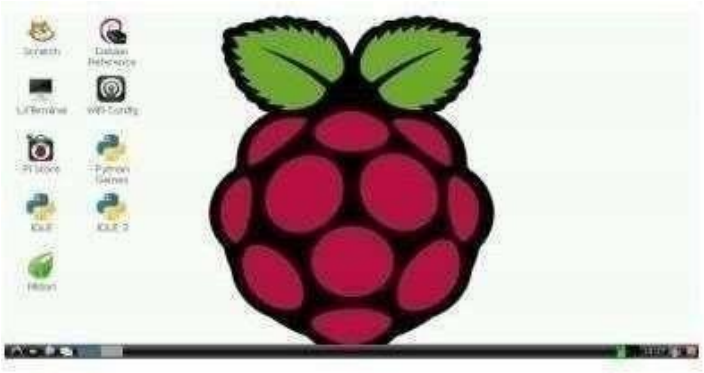


Fig.5.2: Raspberian OS

Raspbian is a Debian-based computer operating system for Raspberry Pi. There are several versions of Raspbian including Raspbian Stretch and Raspbian Jessie. Since 2015 it has been officially provided by the Raspberry Pi Foundation as the primary operating system for the family of Raspberry Pi single-board computers. Raspbian was created by Mike Thompson and Peter Green as an independent project. The initial build was completed in June 2012. The operating system is still under active development. Raspbian is highly optimized for the Raspberry Pi line's low-performance ARMCPUs.

Raspbian uses PIXEL, Pi Improved X-Window Environment, Lightweight as its main desktop environment as of the latest update. It is composed of a modified LXDE desktop environment and the Openbox stacking window manager with a new theme and few other changes. The distribution is shipped with a copy of computer algebra program Mathematica and a version of Minecraft called Minecraft Pi as well as a lightweight version of Chromium as of the latest version.

5.2.2 Tesseract OCR

Python Tesseract is an optical character recognition (OCR) engine for various OS. Tesseract OCR is the process of electronically extracting1text from images and1reusing it in a variety of ways1such as document1editing, 1free-text1searches. OCR is a technology that is capable converting documents such as scanned papers, PDF files and captured image into editable data. Tesseract can be used for Linux, Windows and Mac OS. It can be used by programmers to extract typed, printed text from images using an API. Tesseract can use GUI from available 3rd party page. The installation process of tesseract OCR is a combination of two parts-The engine and training data for a language. For Linux OS. Tesseract was in the top three OCR engines in terms of character accuracy in 1995. It is available for Linux, Windows and Mac OS X. However, due to limited resources it is only rigorously tested by developers under Windows andUbuntu.

Tesseract up to and including version 2 could only accept TIFF images of simple one- column text as inputs. These early versions did not include layout analysis, and so inputting multi-columned text, images, or equations produced garbled output. Since version 3.00 Tesseract has supported output text formatting, hOCR positional information and page-layout analysis. Support for a number of new image formats was added using the Leptonica library. Tesseract can detect whether text is monospaced or proportionally spaced.

Tesseract can process right-to-left text such as Arabic or Hebrew, many Indic scripts as well as CJK quite well. Accuracy rates are shown in this presentation for Tesseract tutorial at DAS 2016, Santorini by Ray Smith. Tesseract is suitable for use as a backend and can be used for more complicated OCR tasks including layout analysis by using a frontend such as OCRopus. Tesseract can be obtained directly from many Linux distributers. The latest stable version of tesseract OCR is 3.05.00. In our project Tesseract is used to convert the captured image text into text format. Tesseract Features: 1) Page layout analysis. 2) supported. 3) Improve forecast accuracy. 4) Add UI.

5.2.3 Open CV

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross platform and free for use under the open-source BSD license. OpenCV supports deep learning frameworks TensorFlow, Torch/PyTorch and Cafe. It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation. Wrappers in other languages such as C#, Perl, Ch, Haskell and Ruby have been developed to encourage adoption by a wider audience. Since version 3.4, OpenCV.js is a JavaScript binding for selected subset of OpenCV functions for the web platform. All of the new developments and algorithms in OpenCV are now developed in the C++ interface. If the library finds Intel's Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate itself. An OpenCL-based GPU interface has been in progress since October 2012, documentation for version 2.4.13.3 can be found at docs.opencv.org.

5.2.4 Espeak

Originally known as speak and originally written for Acorn/RISC_OS computers starting in 1995. This version is an enhancement and re-write, including a relaxation of the original memory and processing power constraints, and with support for additional languages. It's a GUI program used to prepare and compile phoneme data. It is now available for download. Documentation is currently sparse, but if you want to use it to add or improve language support.

It is a compact open source software speech synthesizer for English and 11other languages for Linux and Windows platform. It is used to convert text to voice. It supports many languages in a small size. The programming for Espeak software is done using rule files with feedback. It supports SSML. It can be modified by voice variant. These are text files which can change1characteristics such as1pitch range, add effects1such as echo, whisper and croaky voice, or make systematic adjustments to formant frequencies to change the sound of the voice. The default speaking speed of 180 words per minute is too fast to be intelligible. In our project, Espeak is used to convert the text to voice signal. The eSpeak speech synthesizer supports several languages, however in many cases these are initial drafts and need more work to improve them. Assistance from native speakers is welcome for these, or other new languages. Please contact me if you want to help. Espeak does text to speech synthesis for the following languages: Afrikaans, Albanian, Aragonese, Armenian, Bulgarian, Cantonese, Catalan, Croatian, Czech, Danish, Dutch, English, Esperanto, Estonian, Farsi, Finnish, French, Georgian, German, Greek, Hindi, Hungarian, Icelandic, Indonesian, Irish, Italian, Kannada, Kurdish, Latvian, Lithuanian, Lojban, Macedonian, Malaysian, Malayalam, Mandarin, Nepalese, Norwegian, Polish, Portuguese, Punjabi, Romanian, Russian, Serbian, Slovak, Spanish, Swahili, Swedish, Tamil, Turkish, Vietnamese, Welsh.

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

5.3 Python:

Fig. 5.3: Python logo

Python is an interpreted high-level, general-purpose programming language. Created by Guido van Rossum and first released on 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much python 2code does not run unmodified on python 3. Due to concern about the amount of code written for python 2, support for python 2.7 was extended to 2020.

Features:

- Python is easy to learn and use. It is developer-friendly and high level programming language.
- Python language is more expressive means that it is more understandable and readable.
    - Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginner

- Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language
- Python language is freely available at official web address. The source-code is also available. Therefore it is open source.
- Python supports object oriented language and concepts of classes and objects come into existence.
- It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.
- Python has a large and broad library and provides rich set of module and functions for rapid application development.

- Graphical user interfaces can be developed using Python.
- It can be easily integrated with languages like C, C++, JAVA

5.4 TESTING

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

Testing Principle

Before applying methods to design effective test cases, a software engineer must understand the basic principle that guides software testing. All the tests should be traceable to customer requirements.

Testing Methods

There are different methods that can be used for software testing. They are,

1. Black-Box Testing

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a

tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

2. White-Box Testing

White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called glass testing or open-box testing. In order to perform white-box testing on an application, a tester needs to know the internal workings of the code. The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

Levels of Testing

There are different levels during the process of testing. Levels of testing include different methodologies that can be used while conducting software testing. The main levels of software testing are:

- **Functional Testing:**

This is a type of black-box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. Functional testing of software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. There are five steps that are involved while testing an application for functionality.

- The determination of the functionality that the intended application is meant to perform.
- The creation of test data based on the specifications of the application.
- The output based on the test data and the specifications of the application.
- The writing of test scenarios and the execution of test cases.
- The comparison of actual and expected results based on the executed test cases.

- **Non-functional Testing**

This section is based upon testing an application from its non-functional attributes. Non-functional testing involves testing software from the requirements which are non-functional in nature but important such as performance, security, user interface, etc. Testing can be done in different levels of SDLC. Few of them are

Unit Testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit testing is often automated but it can also be done manually. The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality. Test cases and results are shown in the Tables.

Unit Testing Benefits

- Unit testing increases confidence in changing/ maintaining code.
- Codes are more reusable.
- Development is faster.
- The cost of fixing a defect detected during unit testing is lesser in comparison to that of defects detected at higher levels.
- Debugging is easy.
- Codes are more reliable.

Sl # Test Case : -	UTC-1
Name of Test: -	Camera Test
Items being tested: -	Capture Image
Sample Input: -	Input image
Expected output: -	Should Capture the image
Actual output: -	Image Capture Success full
Remarks: -	Pass.

Integration Testing:

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing. Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. It occurs after unit testing and before validation testing. Integration testing can be done in two ways: Bottom-up integration testing and Top-down integration testing.

1. Bottom-up Integration

This testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.

2. Top-down Integration

In this testing, the highest-level modules are tested first and progressively, lower-level modules are tested thereafter.

In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic actual situations. Table below shows the test cases for integration testing and their results.

Sl # Test Case : -	UTC-2
Name of Test: -	Raspberry Pi and solenoids Test
Items being tested: -	Power on Test
Sample Input: -	Tested for Power on
Expected output: -	Raspberry Pi should Turned on and Boot Up and solenoids should turn on
Actual output: -	Raspberry Pi booted up
Remarks: -	Pass

System testing:

System testing of software or hardware is testing conducted on a complete, integrated system tevaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic. System testing is important because of the following reasons:

- System testing is the first step in the Software Development Life Cycle, where the application is tested as a whole.
- The application is tested thoroughly to verify that it meets the functional and technical specifications.
- The application is tested in an environment that is very close to the production environment where the application will be deployed.
- System testing enables us to test, verify, and validate both the business requirements as well as the application architecture.

System Testing is shown in below tables

Sl # Test Case : -	STC-1
Name of Test: -	System testing
Item being tested: -	Camera, solenoids
Sample Input: -	Input power on
Expected output: -	Should detect
Actual output: -	Synchronization Worked Successfully
Remarks: -	Pass

Acceptance Testing

Acceptance testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Include recovery testing crashes, security testing for unauthorized user, etc.

Acceptance testing is sometimes performed with realistic data of the client to demonstrate that the software is working satisfactorily. This testing in FDAC focuses on the external behavior of the system.

Test Case ID	System Test Case 1
Description	Application Should Work in All Platforms
Input	Input from Different people
Expected output	Functionality should be according to given criteria
Actual Result/Remarks	Working as expected output.
Passed (?)	Yes

Validation Testing

At the culmination of integration testing, software is completely assembled as a packages; interfacing errors have been covered and corrected, and final series of software tests-validating testing may begin. Validation can be defined in many ways, but a simple definition is that validation succeeds when software functions in a manner that can be reasonably expected by customers. Reasonable expectation is defined in the software requirement specification- a document that describes all users' visible attributes of the software. The specification contains a section title "validation criteria". Information contained in that section forms the basis for validation testing approach.

6.CONCLUSION

This device will surely help the visually impaired to be independent and flourish in this fast-developing world. Access to communication in its broadest sense is access to knowledge, and that is vital for us to achieve the highest degree of personal autonomy and be treated as equals.

7.REFERENCES

- Hyun-Cheol Cho et.al, “Development of a Braille Display using Piezr” in SICE-ICASE International Joint Conference: Bexco, Busan, Korea: October 2006
- https://i.pinimg.com/736x/cc/ba/79/ccba79264c60d6c3c1a2de4e19c2 0497--braille-tattoo-tatouage- braille.jpg
- http://www.iaeme.com/IJECET/issues.asp?JType=IJECET&VType= 7&IType=4 Journal Impact Factor (2016): 8.2691 (Calculated by GISI) www.jifactor.com ISSN Print: 0976-6464 and ISSN Online: 0976-6472
- Lester E. Krueger, “A word-superiority effect with print and braille characters” . DOI: 10.3758/bf03202658
- Bruynooghe, J et.al, “Development of a wireless portable Braille organiser for visually impaired2” in Conf. and Workshop on Assistive Technologies for Vision and Hearing Impairment (CVHI 2001), Castelvecchio Pascoli, Italy.
- Néstor Falcón et.al, “Image Processing Techniques for Braille Writing Recognition” in International Conference on Computer Aided Systems Theory. DOI:10.1007/11556985_49
- Vrushabh S. Dharme, et.al, “Designing of English text to braille conversion system”, in 2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS). DOI: 10.1109/ICIIECS.2015.7193267.
- Adrian Moise et.al, “Automatic system for text to Braille conversion”, 2017 in 9th International Conference on Electronics, Computers an Artificial Intelligence (ECAI). DOI: 10.1109/ECAI.2017.8166391
- Joshua L. Dela Cruz et.al, “Development of a text to braille interpreter for printed documents through optical image processing”, in 2017IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM). DOI: 10.1109/HNICEM.2017.8269523
- Vineeth Kartha et.al, “A gesture controlled text to braille converter”, in 2012 Annual IEEE Ind Conference (INDICON). DOI: 10.1109/INDCON.2012.6420639
- Xuan Zhang et.al, “Text-to-Braille Translator in a Chip”, in 2006 International Conference on Electrical and Computer Engineering. DOI: 10.1109/ICECE.2006.355685
- Zhiming Liu et.al, “Finger-eye: A wearable text reading assistive system for the blind and visually impaired”, in 2016 IEEE International Conference on Real-time Computing and Robotics (RCAR).. DOI: 10.1109/RCAR.2016.7784012

Conference on Inventive Systems and Control (ICISC).. DOI: 10.1109/ICISC.2018.8399091

- Joshi Kumar A. V., “PENPAL - Electronic Pen Aiding Visually Impaired in Reading and Visualizing Textual Contents”, in 2011 IEEE International Conference on Technology for Education. DOI: 10.1109/T4E.2011.34
- Ajinkya Domale, “Printed Book to Audio Book Converter for Visually Impaired”, in 2013 Texas Instruments India Educators' Conference. DOI: 10.1109/TIIEC.2013.27
- Jacob George, “Optical Character Recognition Based Hand-Held Device for Printed Text to Braille Conversion”, in 2011 3rd International Conference on Machine Learning and Computing (ICMLC 2011). DOI: 10.13140/RG.2.1.2534.5440