

**AIDING DEVICE FOR VISUALLY IMPAIRED AND BLIND
PEOPLE**



BY

NC AREEJ SHAHID

NC TASMIYA SHEIKH

NC M. AZEEM SARWAR

SUBMITTED TO THE FACULTY OF DEPARTMENT OF ELECTRICAL ENGINEERING

**MILITARY COLLEGE OF SIGNALS, NATIONAL UNIVERSITY OF SCIENCE AND
TECHNOLOGY, ISLAMABAD**

in partial fulfillment for the requirements of a B.E. Degree in Telecom Engineering

June 2016

CERTIFICATE

It is certified that the work contained in this thesis entitled **“Aiding System for Visually Impaired/Blind People”** carried out by Areej Shahid, Tasmiya Sheikh and M.Azeem Sarwar under the supervision of Lt Col. Dr. Abdul Ghafoor for the partial fulfillment of degree of Bachelors of Telecom (Electrical) Engineering is correct and approved.

Lt Col. Dr. Abdul Ghafoor
Project Supervisor

Dated:

ACKNOWLEDGEMENT

To begin with, there is no greater guide than ALLAH (SWT) Himself and we feel blessed that He gave us enough strength to complete this project well in time. We are thankful and would like to express our sincere gratitude to our project supervisor Lt. Col. Dr. Abdul Ghafoor, our co-supervisor Dr. Mohsin Riaz (COMSATS) and Mr. Saleh Usman (CAST) for their invaluable guidance, continuous encouragement and constant support in making this project possible. We are truly obliged for their guidance from the initial to the final level that enabled us to develop an understanding of this project. Without their advice and assistance it would not have been possible to complete this project.

Lastly we express our gratitude to each and every person who contributed to our final year project directly or indirectly. We would like to acknowledge their comments and suggestions, which were crucial for the successful completion of this project.

ABSTRACT

This device is portable, wearable, unique, low power, cost effective and user friendly. It is specially designed for blind people to help them perform their daily chores more efficiently and independently. It includes the concepts of digital image processing, circuit designing, coding/programming in python language and prototyping. Its processing unit is raspberry pi kit which works along with Arduino-Uno board, USB-camera, various sensors and headphones/speakers. It has multi functionalities which include obstacle detection in the path of the user using an array of ultrasonic sensors, face recognition, determining number of people in front of the user by face detection scheme, identifying paper currency using a camera and it also has health sensors mounted in it which includes pulse sensor and temperature sensor to monitor user's health conditions. All of these functionalities are accessed with the help of dedicated options on a keypad and the outputs of the respective functions are given to the user through an audio signal via headphones/speakers.

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATION

In the name of Allah, the Most Merciful, the Most Beneficent.

To our parents and faculty, without their unflinching support and unstinting cooperation,
a work of this magnitude would not have been possible.

Table of Contents

CHAPTER NO.1: INTRODUCTION.....	1
Overview.....	1
Background and Motivation	1
Scope and Deliverables.....	2
CHAPTER NO.2: LITERATURE REVIEW	3
Use of Background Study	4
CHAPTER NO.3: DESIGN AND SPECIFICATIONS.....	5
Project Description and Salient Features	5
Peripherals Interfacing	5
Interfacing Arduino with Raspberry Pi.....	7
Interfacing Keypad with Raspberry Pi.....	9
Speech output using Raspberry Pi	10
Interfacing Camera with Raspberry Pi	11
CHAPTER NO.4: OBSTACLE DETECTION	12
Ultrasonic Sensors	12
CHAPTER NO.5: HEALTH MONITORING SYSTEM	14
Temperature Measurement using Temperature Sensor	14
Heart Rate measurement using Pulse Sensor.....	16
CHAPTER NO.6: FACE DETECTION AND RECOGNITION SYSTEM	18
Determining number of people using face detection scheme	18
Face Recognition	20
Algorithm Description.....	20
CHAPTER NO.7: IMAGE TO SPEECH CONVERSION.....	25
CHAPTER NO.8: CURRENCY RECOGNITION	27
Extracting Templates	27
Testing on HD Images	29
Implementation on real-time video.....	32

CHAPTER NO.9: RECOMMENDATIONS AND CONCLUSION	35
Recommendations.....	35
Conclusion	35
APPENDIX A: APPROVED SYNOPSIS.....	36
APPENDIX B: PROJECT TIMELINE.....	37
APPENDIX C: COST BREAKDOWN.....	38
APPENDIX D: RASPBERRY-PI CODE	39
APPENDIX E: FINAL DELIVERABLE	36
BIBLIOGRAPHY	58

LIST OF FIGURES

- Figure 1: Raspberry Pi Kit and Its Peripherals
- Figure 2: Block Diagram
- Figure 3: Arduino And R-Pi Kit Interfacing
- Figure 4: Interfacing Results
- Figure 5: Keypad Interfacing
- Figure 6: Camera Interfacing
- Figure 7: Arduino Circuit Diagram
- Figure 8: HC-SR04
- Figure 9: Principle Of Ultrasonic Sensor
- Figure 10: Temperature Sensor Circuit Diagram
- Figure 11: Temperature Sensor
- Figure 12: Pulse Sensor Circuit Diagram
- Figure 13: Pulse Sensor
- Figure 14: Human Pulse
- Figure 15: Haar Features
- Figure 16: Result of Face Detection Module
- Figure 17: Results of Fisherfaces Algorithm
- Figure 18: Grey Scale Conversion
- Figure 19: Results of Face Recognition Module
- Figure 20: Results of Image to Text Conversion
- Figure 21: Results of Text To Speech Conversion

- Figure 22: Currency Templates
- Figure 23: Grayscale Conversion
- Figure 24: Input Image Edge Detection
- Figure 25: Template Image Edge Detection
- Figure 26: Detected Point
- Figure 27: Matched Template
- Figure 28: Results of Currency Recognition
- Figure 29: Final Deliverable

CHAPTER NO.1
INTRODUCTION

Overview

Performing daily routine work for blind people has always been a serious issue. They are not usually independent enough to do their work all by themselves. This issue is the main focus of our project and we have developed a device completely dedicated to them which makes their lives easier, is easy to use as well as portable and it is available in very affordable price.

Background and Motivation

285 million people are estimated to be visually impaired worldwide, out of which, 39 million are blind and 246 million have low vision. In Pakistan 2 million people have no sight while 6 million have low sight. About 90% of the worlds' visually impaired live in low income settings. Existing aids for visually impaired people are either outdated or unreliable with limited functionalities. The devices however, are available in international markets but they are too expensive for low income people of our country.

It is evident that existing devices for visually impaired people are very outdated, unreliable with limited functionalities or are too expensive. In the presence of such devices there are enormous chances for accidents or mishaps to occur while in case of expensive devices, masses of our country cannot afford them. Keeping such problems in consideration, we have designed a low cost, portable and reliable hardware along with the efficient and easy to use software for helping visually impaired people in such a way that they can move easily from one place to another by avoiding obstacles in their way and determining certain health parameters, that too without needing help from other people and also detect/recognize Pakistani currency as well as people around them. This will aid them in performing their daily chores easily, independently and more efficiently in an affordable manner.

Scope and Deliverables

This device is portable, wearable, unique, low power, cost effective and user friendly. It is specially designed for blind people to help them perform their daily chores more efficiently and independently. It includes the concepts of digital image processing, circuit designing, coding/programming in python language and prototyping. Its processing unit is raspberry pi kit which works along with Arduino-Uno board, USB-camera, various sensors and headphones/speakers. It has multi functionalities which include obstacle detection in the path of the user using an array of ultrasonic sensors, face recognition, determining number of people in front of the user by face detection scheme, identifying paper currency using a camera and it also has health sensors mounted in it which includes pulse sensor and temperature sensor to monitor user's health conditions. All of these functionalities are accessed with the help of dedicated options on a keypad and the outputs of the respective functions are given to the user through an audio signal via headphones/speakers.

So through this project we have integrated our theoretical knowledge with practicality and gained further insight and refined our skills in all the fields mentioned above. The major part of this project consists of concepts involving digital image processing and sensors.

CHAPTER NO.2

LITERATURE REVIEW

We learned our Digital image processing conceptual theory from the book Digital Image Processing (Third Edition) by Rafael C. Gonzalez (University of Tennessee) and Richard E. Woods (Med Data Interactive).[1]

Over the last decades, research has been conducted for new devices to design a good and reliable system for blind people to detect obstacles and warn them at danger places. There are some systems which have their benefits as well as deficiencies.

In 2001, a paper published by IEEE “Obstacle detection using adaptive color segmentation and color stereo homography” described an obstacle detection methodology which combines two complementary methods: adaptive color segmentation, and stereo-based color homography. This algorithm is particularly suited for environments in which the terrain is relatively flat and of roughly the same color.[2]

In 2007 Chang Gul Kim and Byung Seop Song designed a microprocessor and a PDA called “Design of a Wearable Walking - Guide System for the Blind”. All of the information about obstacle in front of user is checked by three ultrasound sensor pairs and delivered to the microprocessor which analyzes the information and generates the acoustic signal for alarm. This design does not have a very vast scope because of its very limited functionality.[3]

“An Application of Infrared Sensors for Electronic White Stick” published in 2008 introduced an obstacle avoidance alternative by using an electronic stick that employs an infrared sensor for detecting obstacles along the pathway. The stick detected obstacles in range of 80 cm which is the same as the length of white stick. The stick has different vibration modes for different range which is difficult for a blind to differentiate. It needs time for training which makes it complex and time wasting process. The stick informs the person clearly at dangerous stage which conveys less information and safety.[4]

Horace Josh et al. (2011) presented “A Real-time FPGA – Based Vision System for a Bionic Eye”, an FPGA implementation of a real-time vision system that is mobile and

consists of a CMOS camera, FPGA development board and a head-mounted display. But FPGA based systems are expensive and therefore may be unaffordable for many visually impaired people.[5]

In 2014, “Automatic Obstacle Detection using Image Segmentation” by Pankaj Jain and Dr. Mohan Awasthy implemented the method by dividing it into two parts: Segmenting the obstacle containing image, and then finding the obstacle from those obstacle containing image.[6]

Use of Background Study

These documents are about obstacle detection and object recognition, but each document focuses on a special aspect this project. Positives and negatives are discussed and then the most suitable approach or algorithm is suggested. These background studies not only highlight a lot of problems that occurred for the previous developers but they suggest what improvements can be done in the future in terms of functional and cost efficiency.

CHAPTER NO.3

DESIGN AND SPECIFICATIONS

Project Description and Salient Features

Our device performs various functions using camera; like currency, no of people and face recognition; to help blind people identify them without even touching. It also detects heart rate and temperature of the user using pulse sensor and temperature sensor and also detects obstacle coming in a Blind person's way with the help of an array of Ultrasonic sensors. These sensors send signal to the R-pi kit which will then convert this signal to the speech output via speakers/headphones.

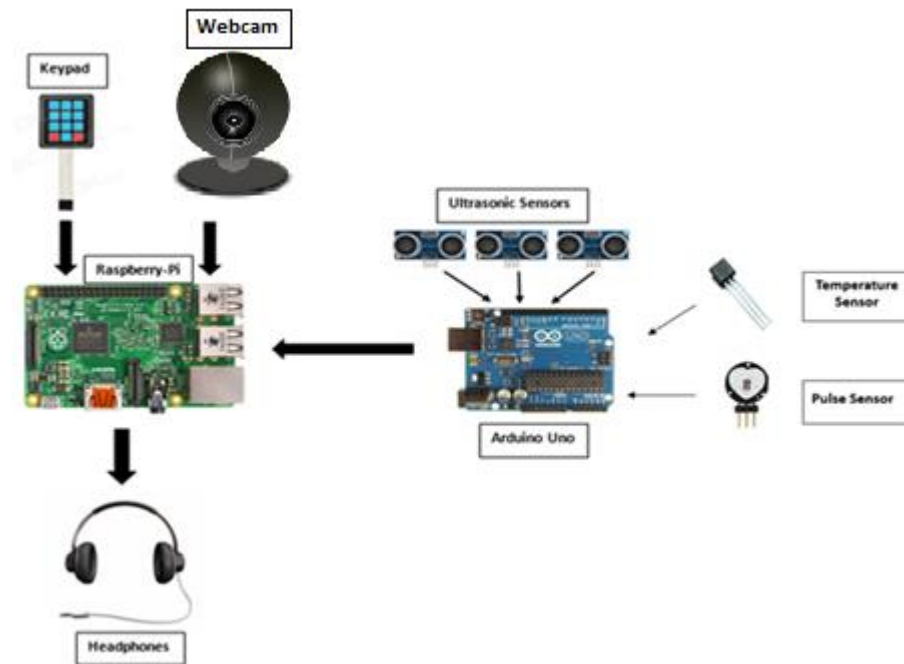


Figure 1: Raspberry pi kit and its peripherals

Peripherals Interfacing

The coding for interfacing camera and Arduino to Raspberry Pi is done using Python language. Image processing is also done using this language on a linux operating system on Raspberry Pi. The sensors send data through Arduino which in turn sends data to

Raspberry Pi. The coding for sensors and programming of Arduino is done using the Arduino IDE.

To control the selection of different options required, we are using a keypad to control different tasks. Pressing of one button perform a certain task while pressing another button begins another task.

The output for the user is given using speech output. The Raspberry Pi is programmed to generate specific commands in speech form through headphones/ speakers.

A block diagram is shown below.

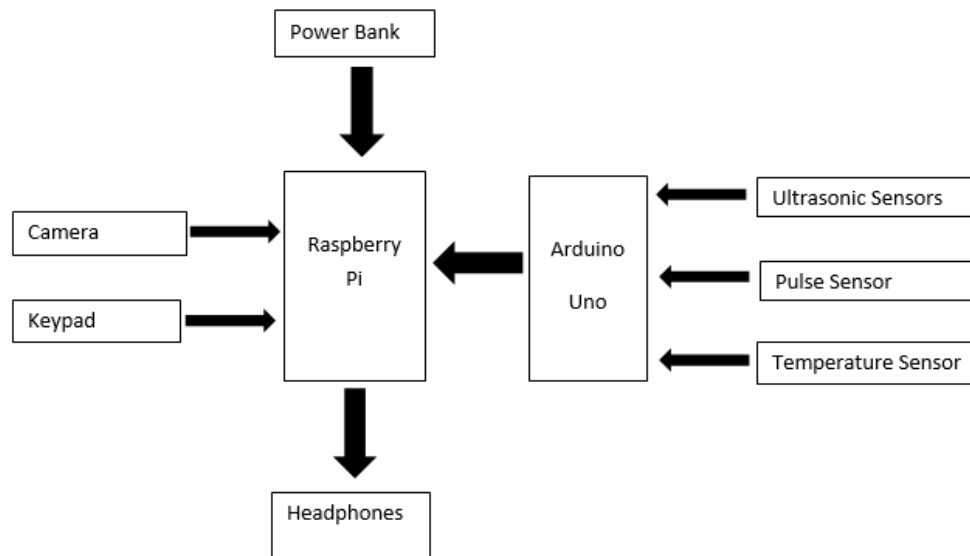


Figure 2: Block diagram

Interfacing Arduino with Raspberry Pi

Connection Diagram

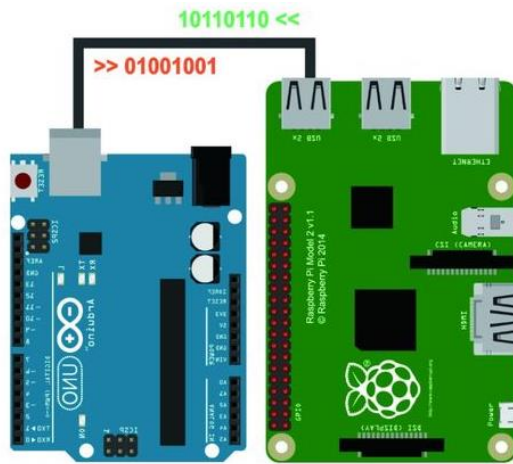


Figure 3: Arduino and R-pi kit interfacing

Our project requires that we are able to give speech output to the user when required. For this purpose we are using Raspberry Pi. In this stage, we send the data collected by the Arduino through sensors to the RPi for further analysis.

To send the data from Arduino to Raspberry Pi, we use serial communication. Serial communication on the Raspberry Pi can be done using either GPIO pins or the USB ports available depending upon the requirements. Since we are trying to make a compact product, we use the second method i.e we use the USB port to send and receive data.

The first thing required is a cable which can be connected to Arduino at one end and RPi at the other end. We are using the normal cable provided with the Arduino for uploading the code. Attaching this cable with the RPi also eliminates the need of external battery for Arduino as power will be provided through the USB port to the Arduino.

Then we establish serial connection with the help of following command in python

```
ser = serial.Serial('/dev/ttyACM0', 9600, timeout = 3)
```

The above command shows that we want to establish a connection with a device connected at “ttyACM0” port, at baud rate of 9600 and we want to check the reception of

data every 3 seconds. But this has an issue regarding the port assignment as it can change after boot or if we reconnect the Arduino.

This means that we would need to test connectivity and change the port assignment in your scripts and programs to match following every restart, and this is not really a practical and acceptable solution for something which is designed to be the heart of an automation system.

To address this we need to create something called a symbolic link which assigns a name to a port based on the device that is connected rather than the literal port assignments.

First, we determine the device parameters for Arduino by running the following command

```
udevadm info -a -p $(udevadm info -q path -n /dev/ttyACM0)
```

which provides information about the device connected to ttyACM0.

Now we need to setup some rules so that when the RPi restarts the port is identified based on the device connected to it. The following command creates the file required that will be read each time the RPi is restarted.

```
sudo nano /etc/udev/rules.d/10-local.rules
```

We enter the required data obtained from first step and set the name of our device as Arduino. After that we rename the port assignment as following.

```
ser = serial.Serial('/dev/ttyArduino', 9600, timeout = 3)
```

Now, we have established a fixed serial connection with Arduino and we can send and receive data using *ser.write* and *ser.read* commands.

```

File Edit Format Run Op
Python Shell
Python 2.7.3 (default, Mar 18 2014, 05:13:23)
[GCC 4.6.3] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
2
132_BFM
2
42_BFM
2
47BPM
2
49_BFM
2
48_BFM
3
70.7_degree_fahrenheit
3
71.04_degree_fahrenheit
3
71.04_degree_fahrenheit
3
71.15_degree_fahrenheit
3
71.15_degree_fahrenheit
1
obstacle at less than 1 ft
1
obstacle at less than 5 ft
1
no obstacle
1
obstacle at less than 1 ft
1
obstacle at less than 1 ft
|
Ln: 36 Col: 1

```

figure 4: interfacing results

Interfacing Keypad with Raspberry Pi

Connection Diagram

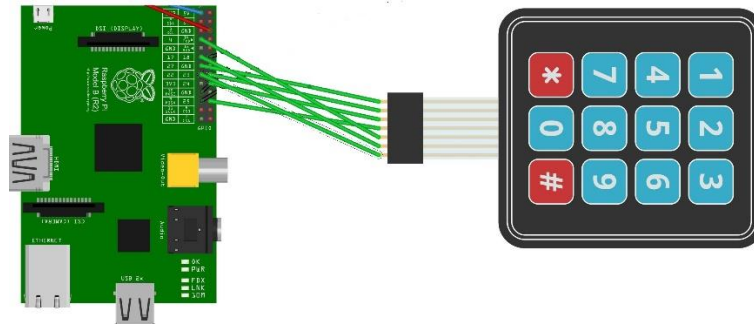


Figure 5: keypad interfacing

The project has a number of operations which can be performed on the request of user. The user needs to select the option so the appropriate script is executed and the required function is performed. For this purpose, we interface a keypad with Raspberry Pi so that the user can get the desired result by just selecting an option on the keypad.

The keypad is connected with the RPi with the help of GPIO pins on RPi. Different pins are present which include power pins, ground pins and I/O pins. We require 7 I/O pins

that needs to be attached with 4 row pins of keypad and 3 I/O pins that needs to be attached with 3 column pins of keypad.

4 GPIO pins as rows are pulled-up internally and initialized as input. Other 3 GPIO pins as columns are initialized as output low. First loop will scan for one pressed key being read as one of the rows pulled-low. After the loop breaks, all columns are set as input, then the row pin found in the loop is set as output-high. Second loop will scan for column being pulled-high by that row pin. The code reads row-column combination of the pressed key and determines which key is pressed.

Speech Output using Raspberry Pi

Speech output requires a few audio software packages to be installed on the R-Pi. We install them in the following way.

Firstly, we update the Raspbian distribution installed on our RPi. Next, we will need the alsa sound utilities. We install them by using the following command.

```
sudo apt-get install alsa-utils
```

and edit the file /etc/modules using the following command.

```
sudo nano /etc/modules
```

Add the following line if not already present in the file.

```
snd_bcm2835
```

Now, we need to select a package which can be used by us as a speech output program. Espeak is a more modern speech synthesis package than other free packages available. It sounds clearer but does wail a little. It is a good package with great customization options.

We install Espeak by using the following command.

```
sudo apt-get install espeak
```

With the help of package installed, we can now get a speech output from our RPi by giving commands or by setting pre-defined scripts.

Interfacing Camera with Raspberry Pi

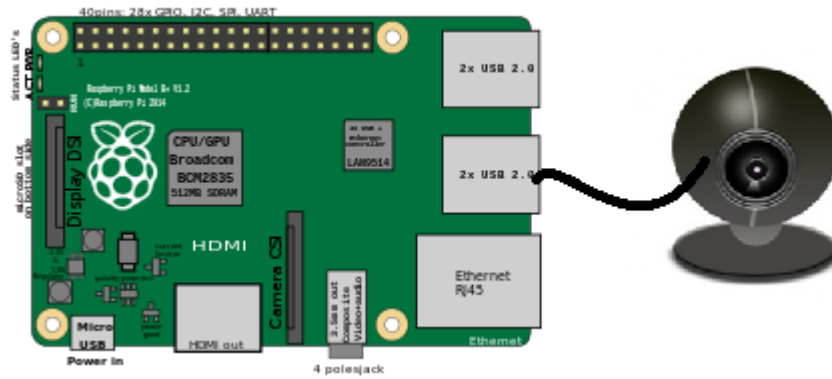


Figure 6: Camera interfacing

CHAPTER NO.4
OBSTACLE DETECTION

Circuit Diagram

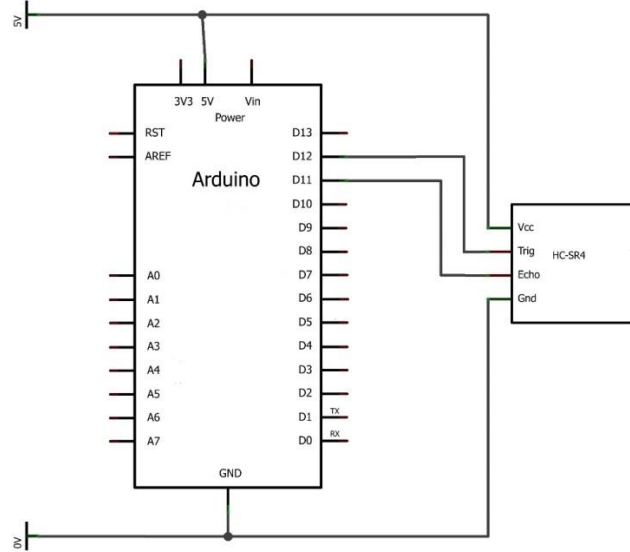


Figure 7: Arduino circuit diagram

Ultrasonic Sensor:

The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. It comes complete with ultrasonic transmitter and receiver module. It can measure distance up to 400 cm. Its effectual measuring angle is about 15 degrees. It operates on 5 volts. It is small in size with dimensions of 45mm x 20mm x 15mm.



Figure 8: HC-SR04

The basic principle of working is by using IO trigger for at least 10us high level signal, the module automatically sends eight 40 kHz and detect whether there is a pulse signal back. If the signal is back, through high level, time of high output IO duration is the time from sending ultrasonic to returning.

$$\text{Test distance} = (\text{high level time} \times \text{velocity of sound}) / 2$$

You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal.

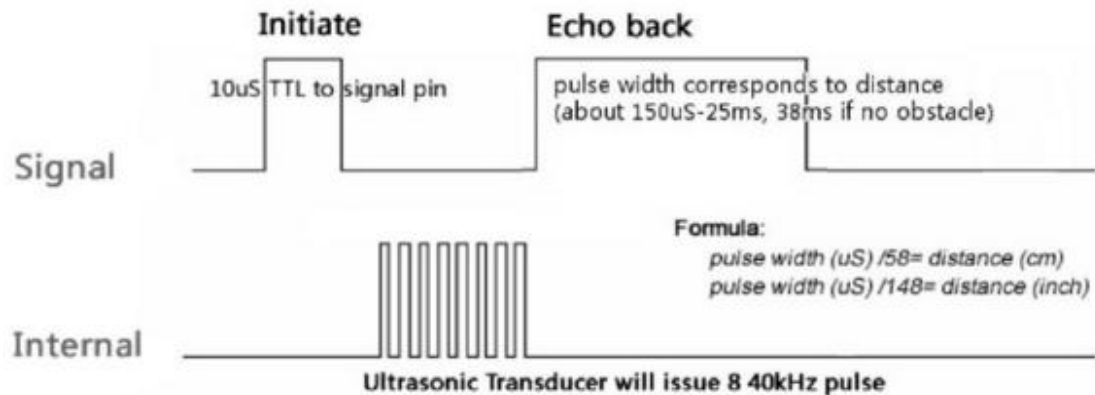


Figure 9: Principle of ultrasonic sensor

HC-SR04 has four pins which are power, ground, echo and trigger. We connect echo and trigger with two digital I/O pins and power and ground of sensor with the power and ground of Arduino. By attaching the 3 ultrasonic sensors, we will be using digital pins 6, 7, 8, 9, 10 and 11 of Arduino attached with the echo and trigger of ultrasonic sensors. Then with the help of our code, we will send a 10us high pulse on trigger and determine the duration of receiving pulse on echo which comes back after hitting an obstacle. Then we determine the distance between the obstacle and sensor by using the following formula:

$$distance = duration/58.2$$

This operation is done for all the three sensors one by one and if any one of the sensors detect an obstacle then the user is informed about it.

CHAPTER NO.5

HEALTH MONITORING SYSTEM

Temperature measurement using temperature sensor

Circuit Diagram

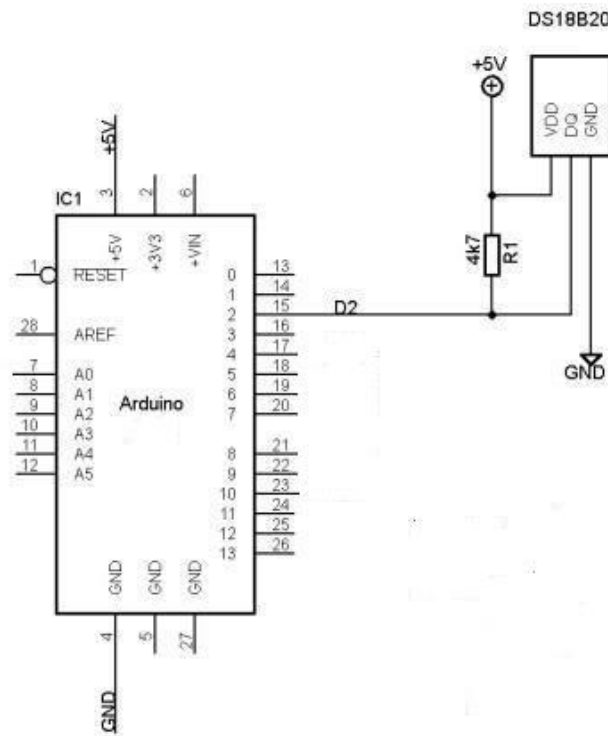


Figure 10: Temperature sensor circuit diagram

Temperature Sensor:

The DS18B20 digital thermometer provides 9-bit to 12-bit Celsius temperature measurements. The DS18B20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. It has an operating temperature range of -55°C to $+125^{\circ}\text{C}$ and is accurate to $\pm 0.5^{\circ}\text{C}$ over the range of -10°C to $+85^{\circ}\text{C}$.



Figure 11: Temperature Sensor

Temperature measurements are made using two bandgap-generated voltage sources. One of the voltage sources has a high-temperature coefficient and changes deterministically across temperature. The other voltage source has a low-temperature coefficient and does not change across temperature. An analog-to-digital converter (ADC) converts the difference between these two voltages to a digital value, representing the temperature of the device. The resolution of the temperature conversion can be selected from 9 to 12 bits.

These values are read by the microcontroller and useful data is extracted. Then the conversion to temperature value is performed to get the temperature.

DS18B20 has three pins which are power, signal and ground. We connect power and ground of sensor with the power and ground of Arduino and the signal pin of sensor to the digital I/O pin 2 of Arduino. We also connect a 4.7k resistor between power and signal pin of sensor. Then we calculate the temperature of user with the help of our code. To get a temperature measurement, we need to issue a command. When the sensor receive it, it will initiate data conversion. All that measurements are stored in the sensor's RAM. We can read it to get the data and we can also write to it to specify sensor resolution. To read the data we issue a command and receive the 9 bytes of data. Then we determine the temperature by using the following formula:

$$Temp = ((HighByte \ll 8) + LowByte) * 0.0625$$

This value is then converted into Fahrenheit and informed to the user.

Heart rate measurement using pulse sensor

Circuit Diagram

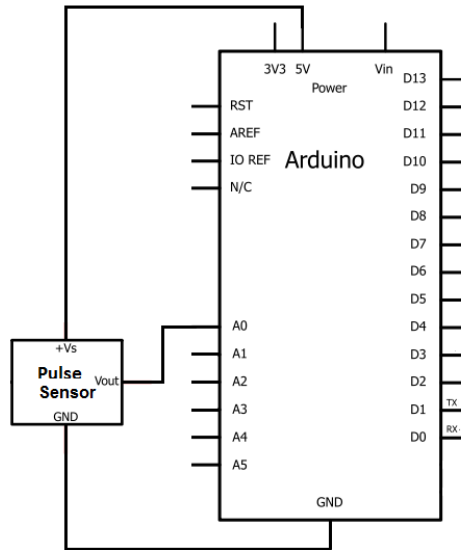


Figure 12: Pulse sensor circuit diagram

Pulse Sensor:

The Pulse Sensor Amped is a plug-and-play heart-rate sensor for Arduino. It essentially combines a simple optical heart rate sensor with amplification and noise cancellation circuitry making it fast and easy to get reliable pulse readings. Also, it sips power with just 4mA current draw at 5V so it's great for mobile applications.



Figure 13: Pulse Sensor

When the Pulse Sensor is just sitting there, not in contact with any finger, the analog signal hovers around the mid-point of the voltage i.e. $V/2$. When the Pulse Sensor is in close contact with your fingertip, the change in reflected light when the blood pumps through your tissues makes the signal fluctuate around that reference point.

Arduino watches the analog signal from Pulse Sensor and decides a pulse is found when the signal rises above mid-point. That is the moment when your capillary tissues gets slammed with a surge of fresh blood. Then, when the signal drops below the mid-point, Arduino sees this and gets ready to find the next pulse.

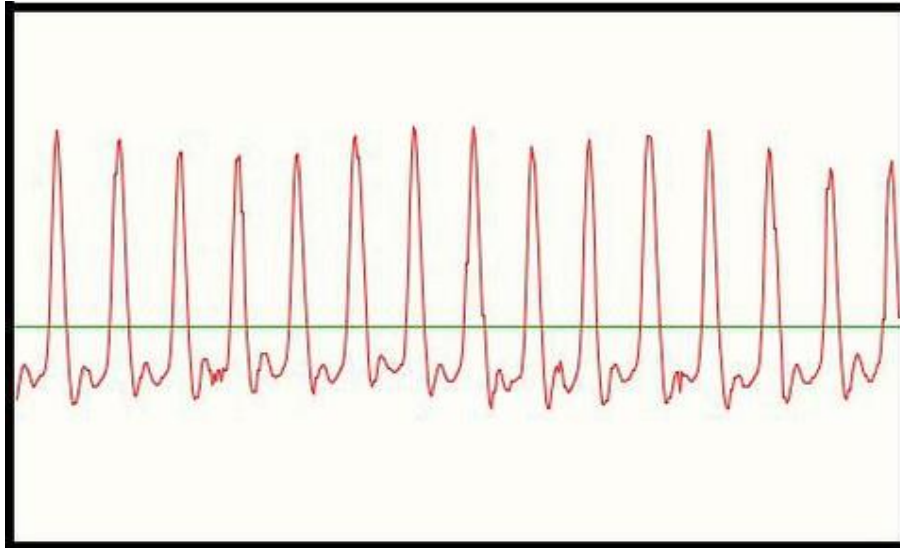


Figure 14: Human Pulse

Pulse sensor has three pins which are power, ground and signal. We connect power and ground of sensor with the power and ground of Arduino and the signal pin of sensor to the analog I/O pin 0 of Arduino. Timer2, an 8 bit hardware timer on the ATmega328 (UNO), is set up so that it throws an interrupt every other millisecond. That gives us a sample rate of 500Hz, and beat-to-beat timing resolution of 2mS. Therefore, an interrupt is generated every 2ms and an interrupt service routine is run where the signal value is recorded. The signal value received is checked if it lies in the range pre-set for a pulse value. If not, then it is discarded else it is used to find the pulse rate. We need to find successive moments of instantaneous heartbeat. We then measure the time between these moments, called the Inter Beat Interval (IBI). Heart Rate (HR) is the interval between successive heartbeats (IBI). IBI is inversely related to HR by the equation

$$HR = 60000 / IBI$$

where IBI is calculated in milliseconds.

CHAPTER NO.6

FACE DETECTION AND RECOGNITION SYSTEM

Determining Number of People Using Face Detection Scheme

In this module faces are detected using a camera mounted with the raspberry pi kit. It is done in order to find the number of people present in front of the user. When the user wishes to find the number of people in a specific location he presses a certain dedicated button on the keypad which evokes this function.

Face Detection in this module is achieved using Haar feature-based cascade classifiers in OpenCV, which is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, “Rapid Object Detection using a Boosted Cascade of Simple Features” in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, haar features shown in below image are used. They are just like convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle.

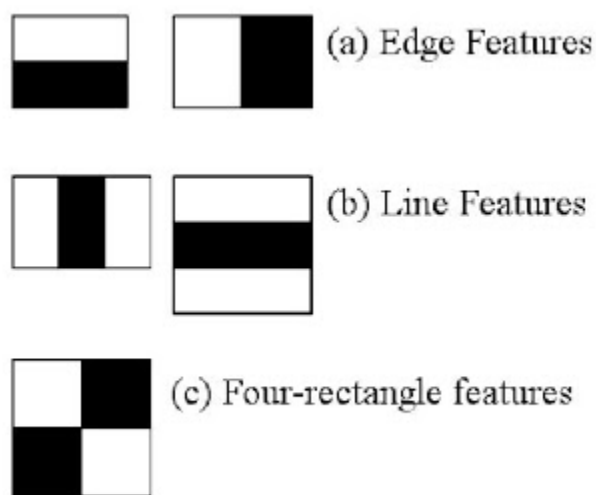


Figure 15: Haar Features

The relevant features are selected out of an image using the concept of Adaboost. For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. We select the

features with minimum error rate, which means they are the features that best classifies the face and non-face images.

In an image, most of the image region is non-face region. So we check if a window is not a face region. If it is not, it is discarded in a single shot and it is not processed again. We then focus on region where there can be a face. This way, we find more time to check a possible face region.

For this the concept of Cascade of Classifiers is used. Instead of applying all features on a window, we group the features into different stages of classifiers and apply one-by-one. If a window fails the first stage, it is discarded. We don't consider remaining features on it. If it passes, we apply the second stage of features and continue the process. The window which passes all stages is a face region. This algorithm has 6000+ features with 38 stages with 1, 10, 25, 25 and 50 features in first five stages. (Two features in the above image is actually obtained as the best two features from Adaboost). On an average, 10 features out of 6000+ are evaluated per sub-window. The results of testing of this module are shown in the following figure.

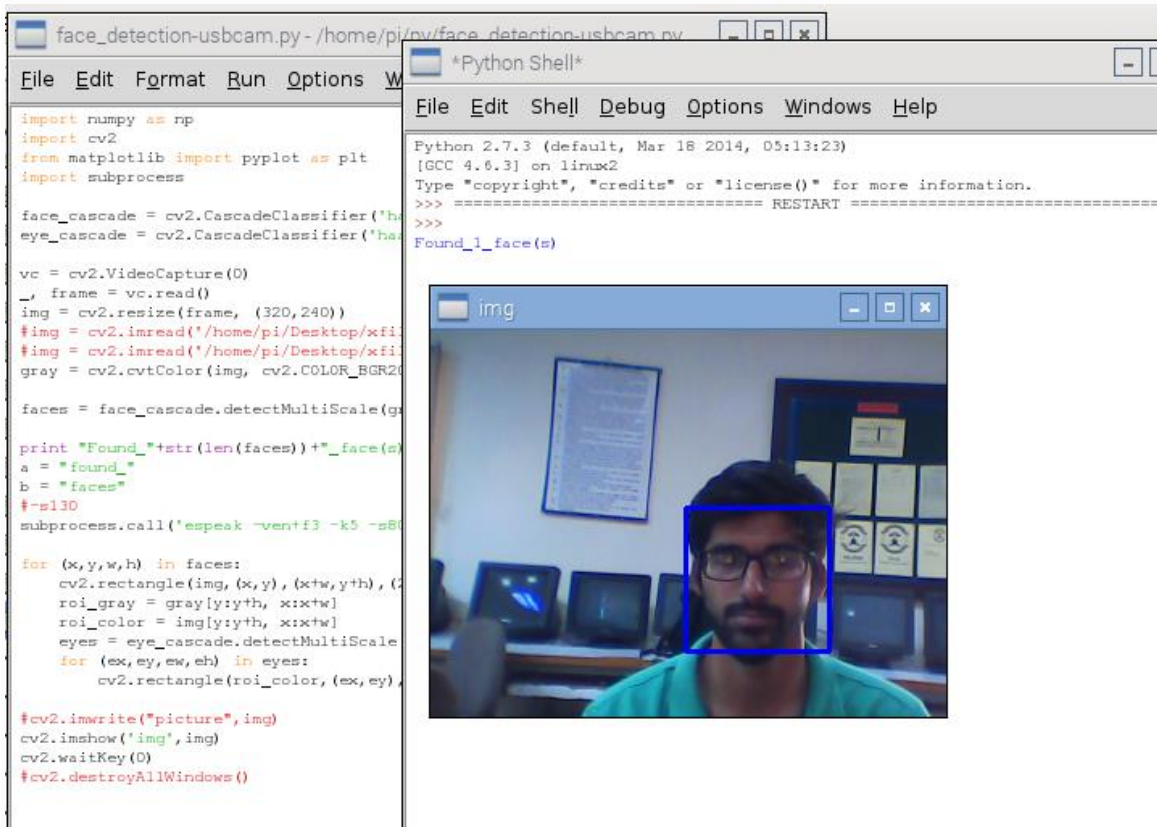


Figure 16: Results of face detection module

When the user evokes this function, the above mentioned scheme is processed and number of people is determined in the form of text which is then converted to speech and an output of this audio signal is sent to the user via headphones.

Face Recognition

Faces can be recognized in a similar manner. First of all the faces are detected using the same scheme mentioned above and then the data bases of known people are made in the raspberry pi kit along with the training images of faces and their names. Whenever this function is required a specified dedicated key is pressed on the keypad and this function working in real time gives an audio signal to the user via headphones.

The face recognition is achieved in our project using the concepts of fisherfaces algorithm. The Principal Component Analysis (PCA), which is the core of the Eigenfaces method, finds a linear combination of features that maximizes the total variance in data. While this is clearly a powerful way to represent data, it doesn't consider any classes and so a lot of discriminative information may be lost when throwing components away. Imagine a situation where the variance in your data is generated by an external source, let it be the light. The components identified by a PCA do not necessarily contain any discriminative information at all, so the projected samples are smeared together and a classification becomes impossible.

The Linear Discriminant Analysis performs a class-specific dimensionality reduction and was invented by the great statistician Sir R. A. Fisher. He successfully used it for classifying flowers in his 1936 paper "The use of multiple measurements in taxonomic problems". In order to find the combination of features that separates best between classes the Linear Discriminant Analysis maximizes the ratio of between-classes to within-classes scatter, instead of maximizing the overall scatter. The idea is simple: same classes should cluster tightly together, while different classes are as far away as possible from each other in the lower-dimensional representation.

Algorithm Description:

Let \mathbf{X} be a random vector with samples drawn from \mathbf{c} classes:

$$\begin{aligned} \mathbf{X} &= \{X_1, X_2, \dots, X_c\} \\ X_i &= \{x_1, x_2, \dots, x_n\} \end{aligned}$$

The scatter matrices \mathbf{S}_B and \mathbf{S}_W are calculated as:

$$S_B = \sum_{i=1}^c N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

$$S_W = \sum_{i=1}^c \sum_{x_j \in X_i} (x_j - \mu_i)(x_j - \mu_i)^T$$

where μ is the total mean:

$$\mu = \frac{1}{N} \sum_{i=1}^N x_i$$

And μ_i is the mean of class $i \in \{1, \dots, c\}$:

$$\mu_i = \frac{1}{|X_i|} \sum_{x_j \in X_i} x_j$$

Fisher's classic algorithm now looks for a projection W , that maximizes the class separability criterion:

$$W_{\text{opt}} = \arg \max_W \frac{|W^T S_B W|}{|W^T S_W W|}$$

Following is a solution for this optimization problem is given by solving the General Eigenvalue Problem:

$$\begin{aligned} S_B v_i &= \lambda_i S_W v_i \\ S_W^{-1} S_B v_i &= \lambda_i v_i \end{aligned}$$

There's one problem left to solve: The rank of S_W is at most $(N - c)$, with N samples and c classes. In pattern recognition problems the number of samples N is almost always smaller than the dimension of the input data (the number of pixels), so the scatter matrix S_W becomes singular. This was solved by performing a Principal Component Analysis on the data and projecting the samples into the $(N - c)$ -dimensional space. A Linear Discriminant Analysis was then performed on the reduced data, because S_W isn't singular anymore.

The optimization problem can then be rewritten as:

$$W_{pca} = \arg \max_W |W^T S_T W|$$

$$W_{fld} = \arg \max_W \frac{|W^T W_{pca}^T S_B W_{pca} W|}{|W^T W_{pca}^T S_W W_{pca} W|}$$

The transformation matrix W , that projects a sample into the $(c - 1)$ -dimensional space is then given by:

$$W = W_{fld}^T W_{pca}^T$$

Following are the results of an experiment performed in order to show the results of this algorithm. Each Fisherface has the same length as an original image, thus it can be displayed as an image.

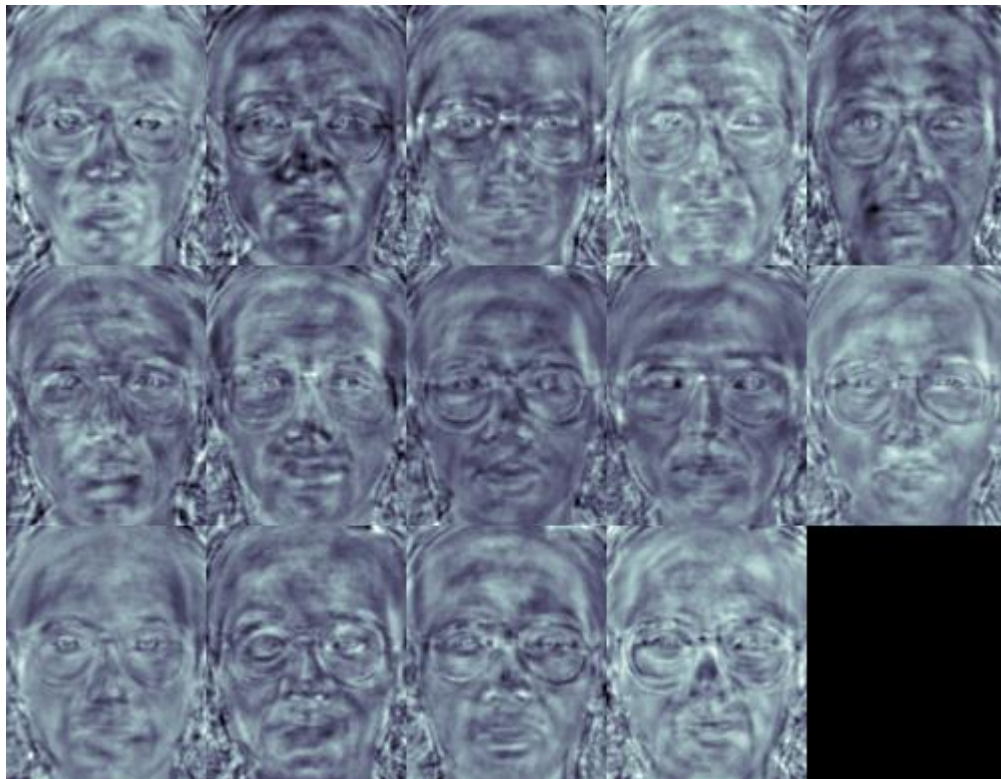


Figure 17: results of Fisherfaces algorithm

The Fisherfaces method learns a class-specific transformation matrix, so they do not capture illumination as obviously as the Eigenfaces method. The Discriminant Analysis instead finds the facial features to discriminate between the persons. It's important to mention, that the performance of the Fisherfaces heavily depends on the input data as well. Practically said: if you learn the Fisherfaces for well-illuminated pictures only and you try to recognize faces in bad-illuminated scenes, then method is likely to find the wrong components (just because those features may not be predominant on bad

illuminated images). This is somewhat logical, since the method had no chance to learn the illumination.

OpenCV comes with a trainer as well as detector. It already contains many pre-trained classifiers for face, eyes, smile etc. Those XML files are stored in opencv's haar-cascades folder. We used face and eyes XML files for this module. First we load the required XML classifiers. Then load our input image taken by the camera. Once it is loaded, we convert it into grey scale for reduced manipulation complexity.

Colored image → Grey scaled image

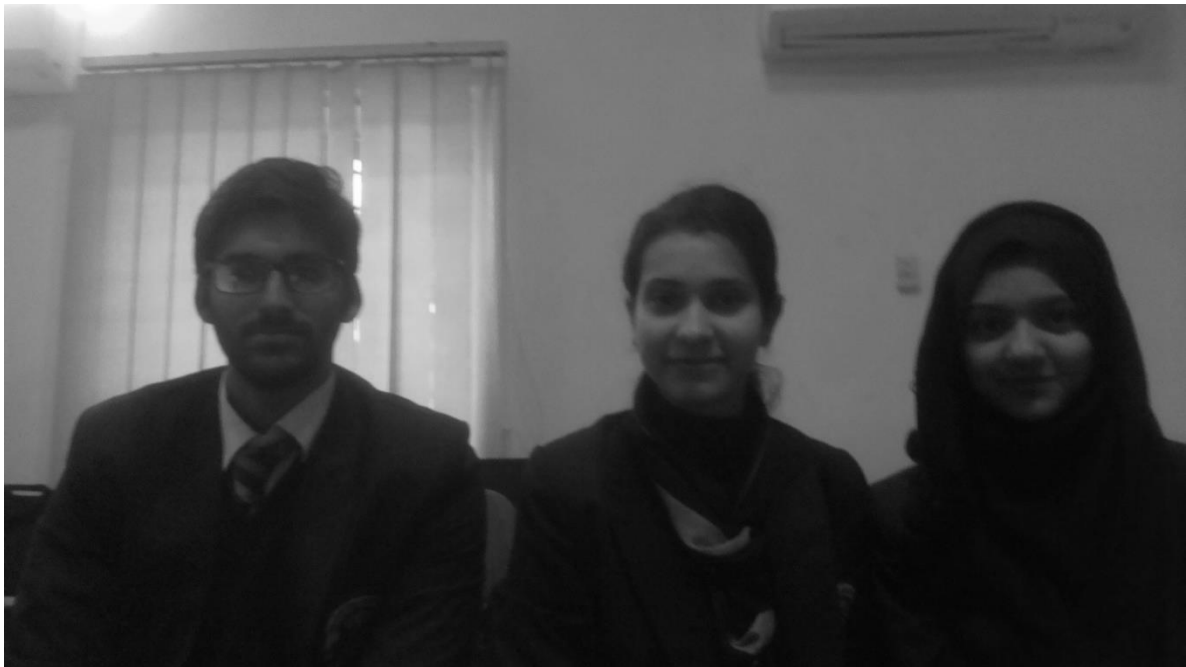


Figure 18: grey scale conversion

Once the XML file is loaded and the image is converted into grey scale we find the faces in the image. For that purpose we run a loop in which all the pixels of the image are analyzed and matched one by one with the face xml file. If faces are found, the coordinates or the location of those pixels are returned. Once we get these locations, we create a region of interest for the face and apply eye detection on this area. Same scheme is implemented in this phase and if the eyes are matched in the location of detected faces, rectangle is created using a build in function present in python.

The faces extracted in the above process are then compared with the images stored in the data base of the kit. This process is done in real time and once the images match with the

frames taken by the camera, the results are displayed in the form of names and a speech signal is sent to the headphone in order to inform the user about the recognized person. Following image is the result of the scheme applied on the above picture.

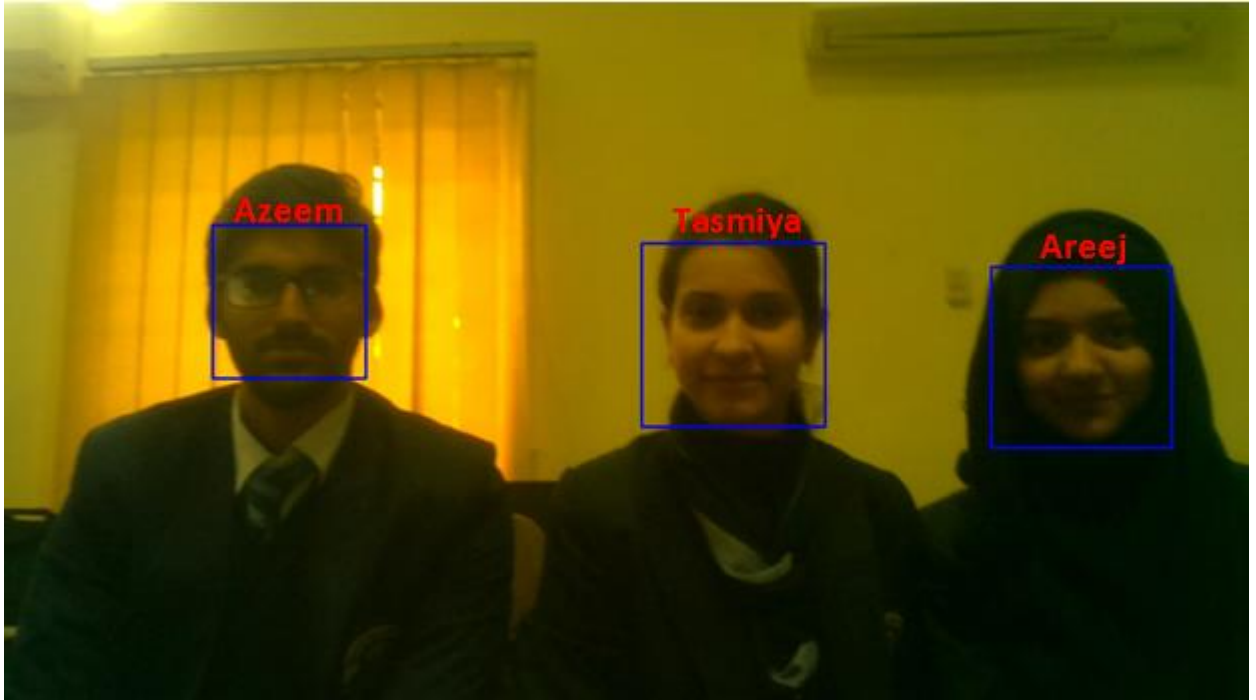


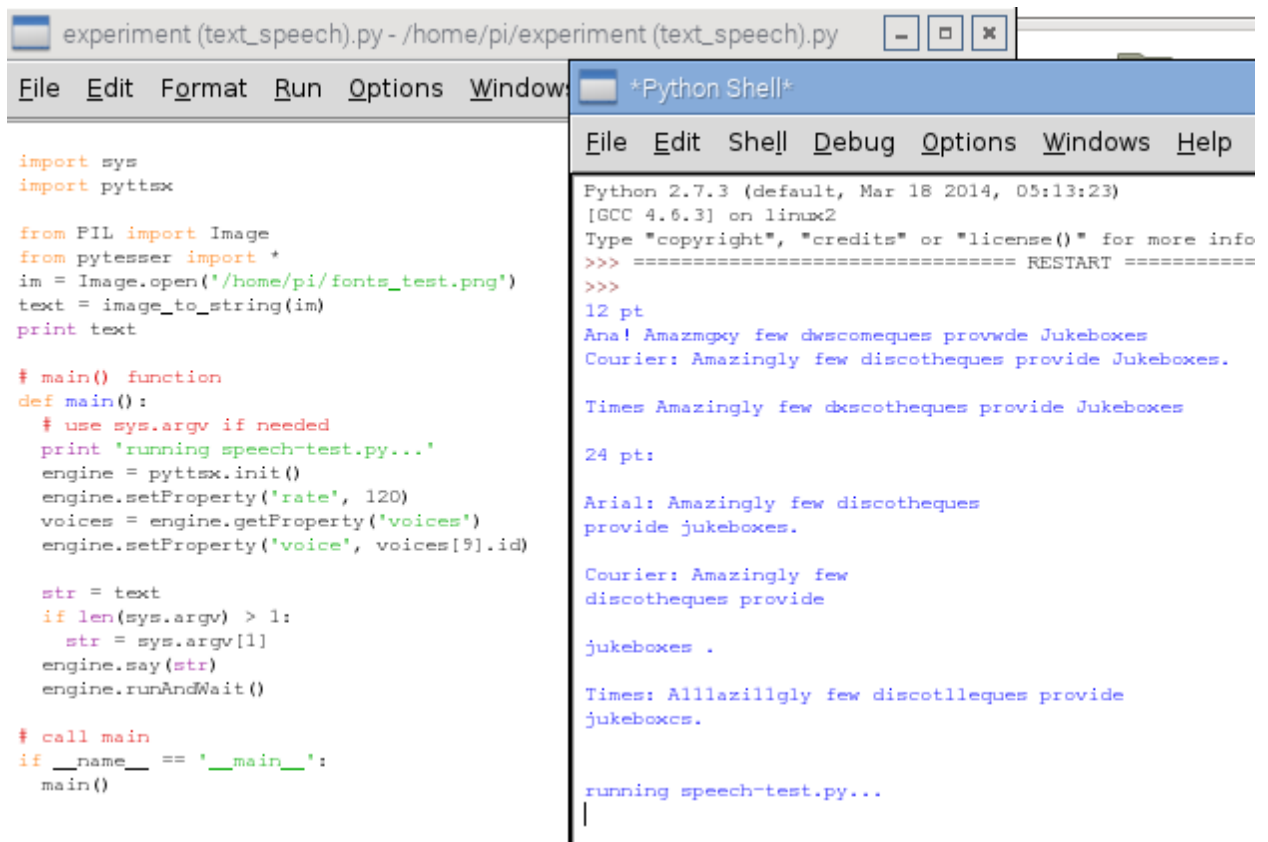
Figure 19: Results of face recognition module

CHAPTER NO.7

IMAGE TO SPEECH CONVERSION

This module is further divided into two parts.

In first part the image is taken using the camera and it is converted into text. This function loads the image, selects the digits/letters/symbols by contour finding and applying constraints on area and height of letters to avoid false detections and draws the bounding rectangle around one letters. Once this is done, it resizes this box to 10x10 and saves 100 pixel values in an array. Then this array is saved in separate txt files. The results of this part are shown below.



```
experiment (text_speech).py - /home/pi/experiment (text_speech).py
File Edit Format Run Options Window *Python Shell*
File Edit Shell Debug Options Windows Help
Python 2.7.3 (default, Mar 18 2014, 05:13:23)
[GCC 4.6.3] on linux2
Type "copyright", "credits" or "license()" for more info
>>> ===== RESTART =====
>>>
12 pt
Ana! Amazmgxy few dwscomeques provwde Jukeboxes
Courier: Amazingly few discotheques provide Jukeboxes.

Times Amazingly few dxscotheques provide Jukeboxes

24 pt:
Arial: Amazingly few discotheques
provide jukeboxes.

Courier: Amazingly few
discotheques provide

jukeboxes .

Times: Alllazillgly few discotlleques provide
jukeboxcs.

running speech-test.py...
|
```

Figure 20: Results of image to text conversion

In second part this text is converted into speech using eSpeak command that is present in OpenCV and Python. Its libraries are open source and can directly be downloaded and installed in your system. 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. Google has integrated eSpeak, an open source software speech

synthesizer for English and other languages, in its online translation service Google Translate. The move allow users of Google Translate to hear translations spoken out loud (text-to-speech) by clicking the speaker icon beside some translations.

The results of this part are shown below where the text that was converted from image is then converted to speech using the above scheme and it is then audible to the user with the help of headphones/speakers.

```
Type "copyright", "credits" or "license()" for more info
>>> ===== RESTART =====
>>>
12 pt
Ana! Amazmgxy few dwscomeques provwde Jukeboxes
Courier: Amazingly few discotheques provide Jukeboxes.

Times Amazingly few dxscotheques provide Jukeboxes

24 pt:

Arial: Amazingly few discotheques
provide jukeboxes.

Courier: Amazingly few
discotheques provide

jukeboxes .

Times: Alllazillgly few discotlleques provide
jukeboxcs.

running speech-test.py...
|
```

Figure 21: Results of text to speech conversion

CHAPTER NO.8

CURRENCY RECOGNITION

Currency is identified via image processing in OpenCV/Python using template matching [7]. Template matching is a technique in digital image processing for finding small parts of an image which match a template image. The two primary components needed for this technique are

- Source image: The image in which we expect to find a match to the template image acquired from the USB camera attached with Raspberry-Pi
- Template image: The patch image which will be compared to the template image within source image

A basic method of template matching uses a filter mask called template, tailored to a specific feature of the search/source image, which we want to detect. It is normally implemented by picking out a part of the source image. This technique can be easily performed on grey images or edge images.

Let source image be denoted by $S(x, y)$, where (x, y) represent the coordinates of each pixel in it and template be denoted by $T(x_t, y_t)$, where (x_t, y_t) represent the coordinates of each pixel in the template.

The correlation of this template with input image is then calculated. For this purpose, center (or the origin) of the template is moved over each point in the source image and sum of products between the coefficients in $S(x, y)$ and $T(x_t, y_t)$ is calculated over the whole area spanned by the template. As all possible positions of the template with respect to the source image are considered, the output will be highest at places where the image structure matches the mask structure.

Extracting Templates

The template kernels/masks are manually extracted from the HD currency images available. For every currency note, four templates are extracted i.e. one from front side,

one from the back side and other two being the inverted versions of these for making identification possible from any side and inversion too.

	Front side upright	Front side inverted	Back side upright	Back side inverted
10 Rupees				
20 Rupees				
50 Rupees				
100 Rupees				
500 Rupees				
1000 Rupees				
5000 Rupees				

Figure 22: currency templates

Testing on HD Images

As an initial testing of algorithm static, high definition (HD) and perfectly aligned input/source images were used and template matching was applied on grey images. For this purpose, both source and template images were subjected to following three processes:

Preprocessing

The sample image of the object and the input image taken from the R-Pi camera are preprocessed by conversion to grayscale which reduces the complexity of manipulating 3D (RGB images) and then de-noising. The results of this preprocessing are shown:

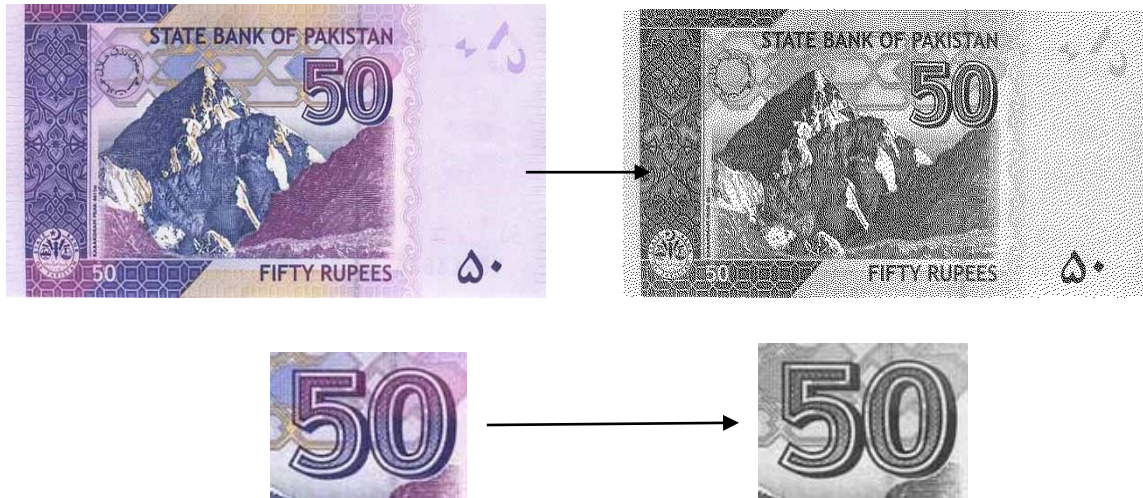


Figure 23: grayscale conversion

Edge Detection

It is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness. Edge detection is used for image segmentation and data extraction. Subjecting our static image to this technique enhances the efficiency of code as using *edges* rather than the *raw image* gives a substantial boost in accuracy for template matching. For this purpose, images are first converted to black and white by defining a threshold value which makes edge detection easier and more efficient. The results of testing are shown below:

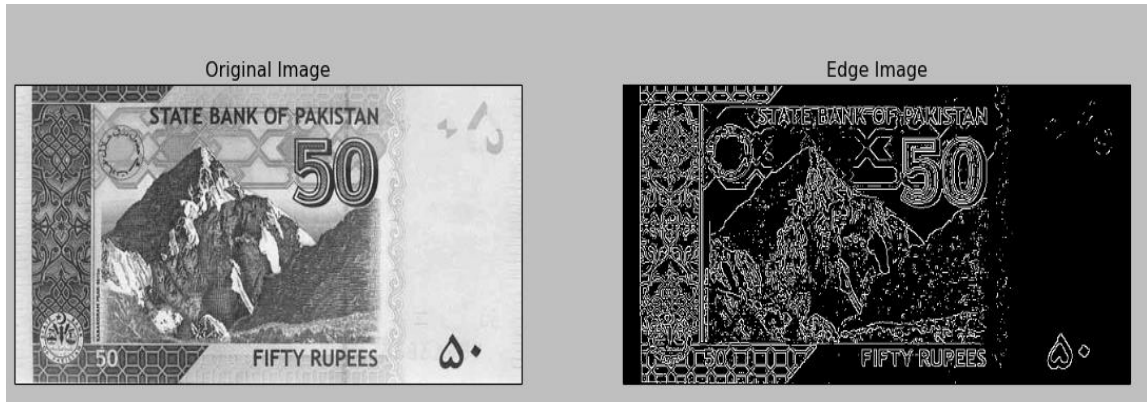


Figure 24: Input image edge detection

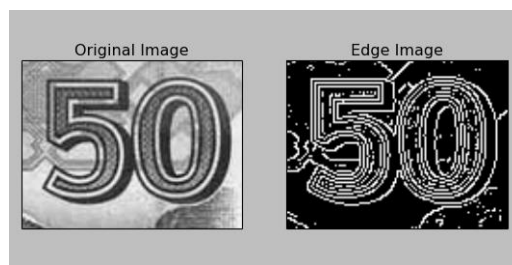


Figure 25: Template Image Edge Detection

Template matching

To identify the matching area, the template image is compared against the source image by sliding it over it and detecting the highest matching area. The patch is moved one pixel at a time (left to right, up to down). At each location, a metric is calculated so it represents quality of the match at that location (or how similar the patch is to that particular area of the source image). For each location the metric is stored in the *result matrix* (\mathbf{R}). Each location (x, y) in \mathbf{R} contains the match metric

OpenCV has a function `cv2.matchTemplate()` [8] for this purpose. It simply slides the template image over the input image (as in 2D convolution) and compares the template and patch of input image under the template image. There are six methods of finding correlation using this particular function listed as:

- `cv2.TM_CCOEFF`
- `cv2.TM_CCOEFF_NORMED`
- `cv2.TM_CCORR`

- cv2.TM_CCORR_NORMED
- cv2.TM_SQDIFF
- cv2.TM_SQDIFF_NORMED

cv2.TM_CCOEFF_NORMED is used in this particular algorithm because of its accuracy and efficiency. This method calculates correlation result value $R(x,y)$ of template image T and source/input image I as:

$$R(x, y) = \frac{\sum_{x',y'} (T'(x', y') \cdot I'(x + x', y + y'))}{\sqrt{\sum_{x',y'} T'(x', y')^2 \cdot \sum_{x',y'} I'(x + x', y + y')^2}}$$

The code slides T over I to apply cv2.matchTemplate() function while keeping track of the match with the largest correlation coefficient (along with the x, y-coordinates of that region). The coordinates of template and the matched region (given by resultant image and cv2.minMaxLoc) are used to draw a rectangle indicating matched result.

Result: Matched Region's Coordinates:

min_val, max_val, min_loc, max_loc= 0.862194478512 0.999967873096 (146, 79)
(317, 42)

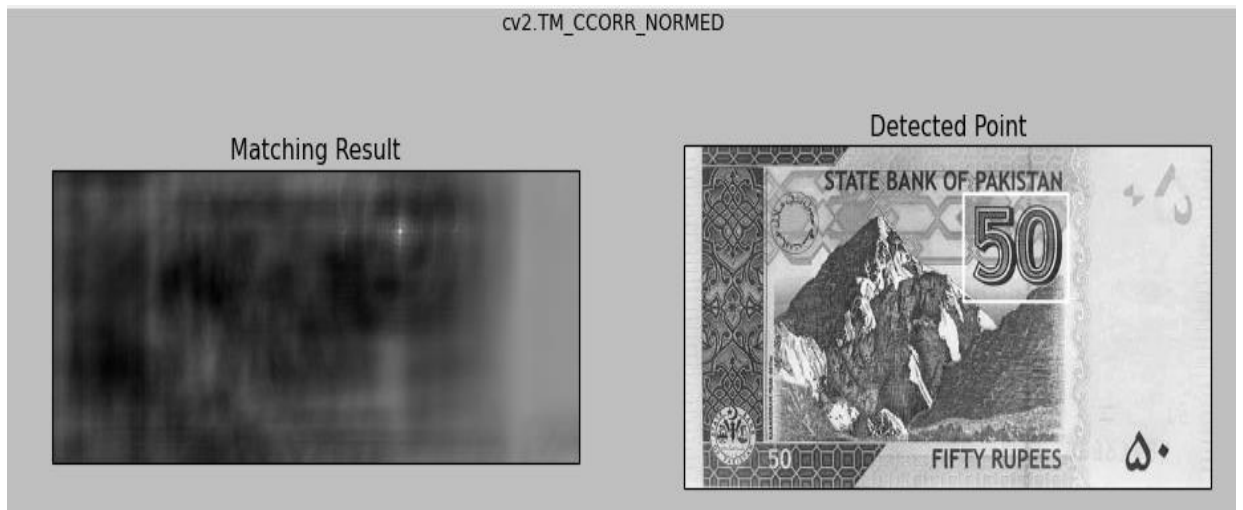


Figure 26: Detected point

A more generalized form of above is attained by setting a minimum threshold value that decides matching and a more robust rectangle plotting accordingly. To save the original image, a copy is made with a separate name and that is saved as a result image.

The results of template matching using 0.8 as minimum value of correlation result are shown below:



Figure 27: Matched Template

Implementation on Real-Time Video

Implementation of above algorithm in real time requires processing on multiple frames per second taken from the video input. Although input image quality is not ideal and the

orientation of currency note may not be perfectly aligned with the USB camera, but since a while loop continuously reads frames, therefore just a slight movement results in the correct orientation with detection of the matched region. Hence edge detection and its prerequisite i.e. de-noising are not required in real time processing.

Following important modifications were done to identify any Pakistani currency note in real time processing:

- Video capturing from USB camera instead of reading saved image
- Input of all currency templates (four per currency note) i.e. front side, inverted front side, back side, inverted back side
- Reading frames from the video
- A continuous loop to read frames and compare it with each template stored to find the best match
- Highlight matched region by plotting a rectangle and labeling
- Print output for every reading to check error
- Speech output to the user indicating amount of the currency

Results

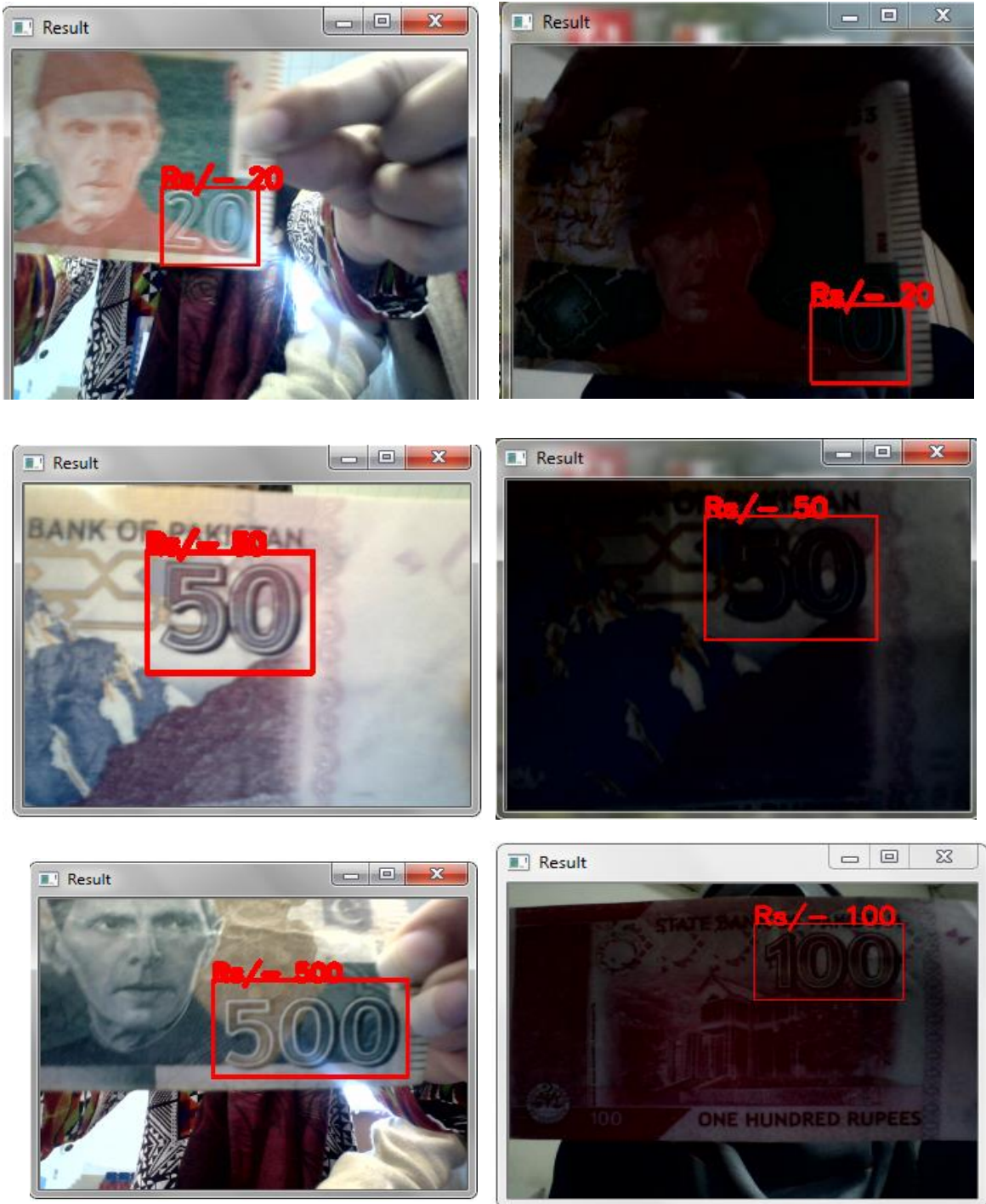


Figure 28: Results of currency recognition

CHAPTER NO.9

RECOMMENDATIONS AND CONCLUSION

Recommendations

In future work following features can be accommodated:

- Instead of a wearable jacket, device can be made as so it can be simply attached with the clothes.
- Further features and functionalities can be added in order to make it more efficient e.g. stairs, doors, windows and car detection.
- Instead of Raspberry pi 2 kit, the latest version i.e. Raspberry pi 3 can be used for faster processing.
- GSM system can be mounted in order to send message about the location of user to his relatives.
- Parallel implementation of functions can be done rather than serial implementation.
- Voice recognition can be added as another feature.

Conclusion

We have been able to make this device portable, wearable, unique, low power, cost effective and easy to use. It is able to help blind people perform their daily chores more efficiently and independently. We have successfully implemented the concepts of digital image processing, circuit designing, coding/programming in python language and prototyping. It is a multi-functional system which successfully performs the function of obstacle detection in the path of the user using an array of ultrasonic sensors, face recognition, determines number of people in front of the user by face detection scheme, identifies paper currency using a camera and it also monitors user's health conditions using pulse and temperature sensors. The outputs of the respective functions are given to the user through an audio signal via headphones/speakers.

So through this project we have integrated our theoretical knowledge with practicality and gained further insight and refined our skills in all the fields mentioned above. The major part of this project consists of concepts involving digital image processing and sensors.

APPENDICES

APPENDIX A

Aiding System for Visually Impaired People

Extended Title: A machine vision system that helps visually impaired people in daily life activities including navigation, object identification and obstacle handling.
Brief Description of The Project / Thesis with Salient Specifications: This project is designed to ease blind people in navigation, avoiding obstacles and identifying different objects. The project will include concepts of digital signal and image processing, programming and prototyping using Raspberry-pi kit.
Scope of Work: This project has a very vast scope in the medical field. It will help blind people to perform their daily activities in a better and easier way. The project will be completed within very limited cost to make it easily accessible for all segments of the society.
Academic Objectives: Digital signal and image processing, programming, circuit designing and prototyping using Raspberry-pi kit.
Application / End Goal Objectives : The end of this project is to develop a low cost, portable and reliable hardware along with efficient and easy to use software for helping visually impaired people in navigation, object detection and obstacle handling.
Previous Work Done on The Subject: There are few products available in internal markets to help blind people. However, they are costly and are generally application specific.
Material Resources Required: Raspberry-pi kit, Headphones/Speaker, Camera, sensors etc
No of Students Required: 3
Group Members: NC Areej Shahid, NC Tasmiya Sheikh, NC M. Azeem Sarwar
Special Skills Required: Nil

APPENDIX B

Week No	Work done during the week
Week 1-2	Literature Review
Week 3-4	Project layout/design
Week 5	Raspberry Pi installation, basic understanding of operations and config.
Week 6-7	Installation of opencv, Basic examples of coding like blinking an LED, Learning python(through tutorials)
Week 8	Camera integration and its basic working, Testing of ultrasonic sensors and Integration with Arduino
Week 9	Image acquisition, Testing of ultrasonic sensors and coding
Week 10-11	Image enhancement and filtering, Testing of ultrasonic sensors and coding
Week 12-13	Segmentation of image, Integration of Arduino with R-Pi
Week 14-15	Edge Detection & Template Matching for currency recognition, Face Detection, Testing of health sensors (B.P, pulse etc)
Week 16-18	Currency Recognition, Face Detection testing, Testing Health Sensors
Week 19	Currency Rec. code enhance, Coding for Book reading, Final Integration of Health Sensors
Week 20-21	Real Time Currency Recognition, Coding for Book reading
Week 22-23	Final Testing: Currency Recognition, Face Detection, Book Reading
Week 24-25	Final Testing: Obstacle Detection, Health Sensors
Week 26-27	Code for audio output (gadget response) for all Applications
Week 28-29	Integration of all Apps- using Keypad
Week 30-32	Code optimization (speed issues)
Week 33-34	Hardware Integration- Final Wearable Device
Week 35-36	Testing and Documentation

APPENDIX C

Name of Equipment	Cost-estimate
Raspberry Pi	6500 PKR
Microsd Card	1350 PKR
Arduino	1250 PKR
Pulse Sensor	1350 PKR
Temperature Sensor	550 PKR
Ultrasonic Sensor x3	350 x3 PKR
Keypad	150 PKR
Camera	1000 PKR
Headphones	750 PKR
Power Bank	3000 PKR
Jacket	300 PKR
Total	17200 PKR

APPENDIX D

#Raspberry-Pi Code

```
import RPi.GPIO as GPIO

import time

import serial

import subprocess

import sys, os

from facerec.feature import Fisherfaces

from facerec.classifier import NearestNeighbor

from facerec.model import PredictableModel

from PIL import Image

import numpy as np

import cv2

from matplotlib import pyplot as plt

import multiprocessing

import sys

import pyttssx

from pytesser import *

ser = serial.Serial('/dev/Arduino', 9600, timeout = 3)

time.sleep(2)

GPIO.setmode (GPIO.BOARD)

MATRIX = [ [1,2,3], [4,5,6], [7,8,9], ['*',0,'#']]

ROW = [31,33,35,37]

COL = [36,38,40]

for j in range(3):

    GPIO.setup(COL[j], GPIO.OUT)

    GPIO.output(COL[j], 1)
```

```

for i in range (4):
    GPIO.setup(ROW[i], GPIO.IN, pull_up_down = GPIO.PUD_UP)
a = 0
s1 = 'aiding_system_for_visually_impaired_people'
s2 = 'select_option'
a1 = 'obstacle_at_less_than_1_feet'
a2 = 'obstacle_at_less_than_5_feet'
a3 = 'no_obstacle'
b1 = 'bpm'
c = 'fahrenheit'
v1 = 'hold_the_sensor_for_10_seconds'
v2 = 'press_the_sensor_for_5_seconds'
v3 = 'taking_picture'
t1 = 'test'
cur10='10_Ruppees'
cur20='20_Ruppees'
cur50='50_Ruppees'
cur100='100_Ruppees'
cur500='500_Ruppees'
cur0=0
#subprocess.call('espeak -ven+f3 -k5 -s130 '+s1, shell=True)
#time.sleep(2)
#subprocess.call('espeak -ven+f3 -k5 -s130 '+s2, shell=True)
#a=6
try:
    while(True):
        for j in range (3):

```

```

GPIO.output(COL[j],0)
for i in range(4):
    if GPIO.input (ROW[i]) == 0:
        #print MATRIX[i][j]
        a = MATRIX[i][j]
        print a
        if a == '*':
            ser.write(b'1')
            print a
            time.sleep(1)
            y = ser.readline()[:-2]
            if y == '1' :
                print("obstacle at less than 1 ft")
                subprocess.call('espeak -ven+f3 -k5 -s130 '+a1, shell=True)
            elif y == '2' :
                print("obstacle at less than 5 ft")
                subprocess.call('espeak -ven+f3 -k5 -s130 '+a2, shell=True)
            elif y == '3' :
                print("no obstacle")
                subprocess.call('espeak -ven+f3 -k5 -s130 '+a3, shell=True)
            ser.write(b'0')
        elif a == 2:
            ser.write(b'2')
            print a
            subprocess.call('espeak -ven+f3 -k5 -s130 '+v2, shell=True)
            time.sleep(2)
            hb = ser.readline()

```

```
print(hb)
ser.write(b'2')
time.sleep(1)
hb = ser.readline()
print(hb)
ser.write(b'2')
time.sleep(1)
hb = ser.readline()
print(hb)
ser.write(b'2')
time.sleep(1)
hb = ser.readline()
print(hb)
y=len(hb)
if y < 10:
    if y > 2:
        subprocess.call('espeak -ven+f3 -k5 -s130 '+hb, shell=True)
ser.write(b'0')
elif a == 3:
ser.write(b'3')
print a
subprocess.call('espeak -ven+f3 -k5 -s130 '+v1, shell=True)
time.sleep(6)
tmp = ser.readline()
print(tmp)
y=len(tmp)
ser.write(b'3')
```

```

time.sleep(1)
tmp = ser.readline()
print(tmp)
y=len(tmp)
ser.write(b'3')
time.sleep(1)
tmp = ser.readline()
print(tmp)
y=len(tmp)
ser.write(b'3')
time.sleep(1)
tmp = ser.readline()
print(tmp)
y=len(tmp)
ser.write(b'3')
time.sleep(1)
tmp = ser.readline()
print(tmp)
y=len(tmp)
if y > 10:
    subprocess.call('espeak -ven+f3 -k5 -s130 '+tmp, shell=True)
ser.write(b'0')
elif a == 4:
    vc = cv2.VideoCapture(0)
    threshold = 0.68
    temp10a = cv2.imread('./images/temp10a.jpg',0)
    w0, h0 = temp10a.shape

```

```
temp10b = cv2.imread('./images/temp10b.jpg',0)
w1, h1 = temp10b.shape
temp10c = cv2.imread('./images/temp10c.jpg',0)
w2, h2 = temp10c.shape
temp10d = cv2.imread('./images/temp10d.jpg',0)
w3, h3 = temp10d.shape
temp20a = cv2.imread('./images/temp20a.jpg',0)
w4, h4 = temp20a.shape
temp20b = cv2.imread('./images/temp20b.jpg',0)
w5, h5 = temp20b.shape
temp20c = cv2.imread('./images/temp20c.jpg',0)
w6, h6 = temp20c.shape
temp20d = cv2.imread('./images/temp20d.jpg',0)
w7, h7 = temp20d.shape-
temp50a = cv2.imread('./images/temp50a.jpg',0)
w8, h8 = temp50a.shape
temp50b = cv2.imread('./images/temp50b.jpg',0)
w9, h9 = temp50b.shape
temp50c = cv2.imread('./images/temp50c.jpg',0)
w10, h10 = temp50c.shape
temp50d = cv2.imread('./images/temp50d.jpg',0)
w11, h11 = temp50d.shape
temp100a = cv2.imread('./images/temp100a.jpg',0)
w12, h12 = temp100a.shape
temp100b = cv2.imread('./images/temp100b.jpg',0)
w13, h13 = temp100b.shape
temp100c = cv2.imread('./images/temp100c.jpg',0)
```



```

w14, h14 = temp100c.shape
temp100d = cv2.imread('./images/temp100d.jpg',0)
w15, h15 = temp100d.shape
temp500a = cv2.imread('./images/temp500a.jpg',0)
w16, h16 = temp500a.shape
temp500b = cv2.imread('./images/temp500b.jpg',0)
w17, h17 = temp500b.shape
temp500c = cv2.imread('./images/temp500c.jpg',0)
w18, h18 = temp500c.shape
temp500d = cv2.imread('./images/temp500d.jpg',0)
w19, h19 = temp500d.shape
while 1:
    frame = vc
    frame = cv2.resize(frame, (320,240))
    img_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    res = cv2.matchTemplate(img_gray,temp10a,cv2.TM_CCOEFF_NORMED)
    loc = np.where( res >= threshold)
    for pt in loc[::-1]
        print "10/-"
        subprocess.call('espeak -ven+f3 -k5 -s130 '+cur10, shell=True)
        cur0=10
        cv2.rectangle(frame, pt, (pt[0] + w0, pt[1] + h0), (0,0,255), 1)
        cv2.putText(frame,'Rs/- 10', (pt[0],
pt[1]),cv2.FONT_HERSHEY_SIMPLEX,0.6,2,1)
        res = cv2.matchTemplate(img_gray,temp10b,cv2.TM_CCOEFF_NORMED)
        loc = np.where( res >= threshold)
        for pt in loc[::-1]
            print "10/-"

```

```

subprocess.call('espeak -ven+f3 -k5 -s130 '+cur10, shell=True)

cur0=10

cv2.rectangle(frame, pt, (pt[0] + w1, pt[1] + h1), (0,0,255), 1)

cv2.putText(frame,'Rs/- 10', (pt[0], pt[1]),
cv2.FONT_HERSHEY_SIMPLEX,0.6,2,1)

res = cv2.matchTemplate(img_gray,temp10c,cv2.TM_CCOEFF_NORMED)

loc = np.where( res >= threshold)

for pt in loc[::-1]

    print "10/-"

    subprocess.call('espeak -ven+f3 -k5 -s130 '+cur10, shell=True)

    cur0=10

    cv2.rectangle(frame, pt, (pt[0] + w2, pt[1] + h2), (0,0,255), 1)

    cv2.putText(frame,'Rs/- 10', (pt[0], pt[1]),
cv2.FONT_HERSHEY_SIMPLEX,0.6,2,1)

res = cv2.matchTemplate(img_gray,temp10d,cv2.TM_CCOEFF_NORMED)

loc = np.where( res >= threshold)

for pt in loc[::-1]

    print "10/-"

    subprocess.call('espeak -ven+f3 -k5 -s130 '+cur10, shell=True)

    cur0=10

    cv2.rectangle(frame, pt, (pt[0] + w3, pt[1] + h3), (0,0,255), 1)

    cv2.putText(frame,'Rs/- 10', (pt[0], pt[1]),
cv2.FONT_HERSHEY_SIMPLEX,0.6,2,1)

res = cv2.matchTemplate(img_gray,temp20a,cv2.TM_CCOEFF_NORMED)

loc = np.where( res >= threshold)

for pt in loc[::-1]

    print "20/-"

    cv2.rectangle(frame, pt, (pt[0] + w4, pt[1] + h4), (0,0,255), 1)

```

```

subprocess.call('espeak -ven+f3 -k5 -s130 '+cur20, shell=True)

cur0=20

cv2.putText(frame,'Rs/- 20', (pt[0], pt[1]),
cv2.FONT_HERSHEY_SIMPLEX,0.6,2,1)

res = cv2.matchTemplate(img_gray,temp20b,cv2.TM_CCOEFF_NORMED)

loc = np.where( res >= threshold)

for pt in loc[::-1]

    print "20/-"

    subprocess.call('espeak -ven+f3 -k5 -s130 '+cur20, shell=True)

    cur0=20

    cv2.rectangle(frame, pt, (pt[0] + w5, pt[1] + h5), (0,0,255), 1)

    cv2.putText(frame,'Rs/- 20', (pt[0], pt[1]),
cv2.FONT_HERSHEY_SIMPLEX,0.6,2,1)

res = cv2.matchTemplate(img_gray,temp20c,cv2.TM_CCOEFF_NORMED)

loc = np.where( res >= threshold)

for pt in loc[::-1]

    subprocess.call('espeak -ven+f3 -k5 -s130 '+cur20, shell=True)

    cur0=20

    print "20/-"

    cv2.rectangle(frame, pt, (pt[0] + w6, pt[1] + h6), (0,0,255), 1)

    cv2.putText(frame,'Rs/- 20', (pt[0], pt[1]),
cv2.FONT_HERSHEY_SIMPLEX,0.6,2,1)

res = cv2.matchTemplate(img_gray,temp20d,cv2.TM_CCOEFF_NORMED)

loc = np.where( res >= threshold)

for pt in loc[::-1]

    print "20/-"

    subprocess.call('espeak -ven+f3 -k5 -s130 '+cur20, shell=True)

    cur0=20

```

```

cv2.rectangle(frame, pt, (pt[0] + w7, pt[1] + h7), (0,0,255), 1)
cv2.putText(frame,'Rs/- 20', (pt[0], pt[1]),
cv2.FONT_HERSHEY_SIMPLEX,0.6,2,1)

res = cv2.matchTemplate(img_gray,temp50a,cv2.TM_CCOEFF_NORMED)
loc = np.where( res >= threshold)
for pt in loc[::-1]
    print "50/-"
    subprocess.call('espeak -ven+f3 -k5 -s130 '+cur50, shell=True)
    cur0=50
    cv2.rectangle(frame, pt, (pt[0] + w8, pt[1] + h8), (0,0,255), 1)
    cv2.putText(frame,'Rs/- 50', (pt[0], pt[1]),
cv2.FONT_HERSHEY_SIMPLEX,0.6,2,1)

res = cv2.matchTemplate(img_gray,temp50b,cv2.TM_CCOEFF_NORMED)
loc = np.where( res >= threshold)
for pt in loc[::-1]
    print "50/-"
    subprocess.call('espeak -ven+f3 -k5 -s130 '+cur50, shell=True)
    cur0=50
    cv2.rectangle(frame, pt, (pt[0] + w9, pt[1] + h9), (0,0,255), 1)
    cv2.putText(frame,'Rs/- 50', (pt[0], pt[1]),
cv2.FONT_HERSHEY_SIMPLEX,0.6,2,1)

res = cv2.matchTemplate(img_gray,temp50c,cv2.TM_CCOEFF_NORMED)
loc = np.where( res >= threshold)
for pt in loc[::-1]
    print "50/-"
    subprocess.call('espeak -ven+f3 -k5 -s130 '+cur50, shell=True)
    cur0=50
    cv2.rectangle(frame, pt, (pt[0] + w10, pt[1] + h10), (0,0,255), 1)

```

```

    cv2.putText(frame,'Rs/- 50', (pt[0], pt[1]),
cv2.FONT_HERSHEY_SIMPLEX,0.6,2,1)

    res = cv2.matchTemplate(img_gray,temp50d,cv2.TM_CCOEFF_NORMED)

    loc = np.where( res >= threshold)

    for pt in loc[::-1]

        print "50/-"

        subprocess.call('espeak -ven+f3 -k5 -s130 '+cur50, shell=True)

        cur0=50

        cv2.rectangle(frame, pt, (pt[0] + w11, pt[1] + h11), (0,0,255), 1)

        cv2.putText(frame,'Rs/- 50', (pt[0], pt[1]),
cv2.FONT_HERSHEY_SIMPLEX,0.6,2,1)

    res = cv2.matchTemplate(img_gray,temp100a,cv2.TM_CCOEFF_NORMED)

    loc = np.where( res >= threshold)

    for pt in loc[::-1]

        print "100/-"

        subprocess.call('espeak -ven+f3 -k5 -s130 '+cur100, shell=True)

        cur0=100

        cv2.rectangle(frame, pt, (pt[0] + w12, pt[1] + h12), (0,0,255), 1)

        cv2.putText(frame,'Rs/- 100', (pt[0], pt[1]),
cv2.FONT_HERSHEY_SIMPLEX,0.6,2,1)

    res =
cv2.matchTemplate(img_gray,temp100b,cv2.TM_CCOEFF_NORMED)

    loc = np.where( res >= threshold)

    for pt in loc[::-1]

        print "100/-"

        subprocess.call('espeak -ven+f3 -k5 -s130 '+cur100, shell=True)

        cur0=100

        cv2.rectangle(frame, pt, (pt[0] + w13, pt[1] + h13), (0,0,255), 1)

```

```

        cv2.putText(frame,'Rs/- 100', (pt[0], pt[1]),
cv2.FONT_HERSHEY_SIMPLEX,0.6,2,1)

        res =
cv2.matchTemplate(img_gray,temp100c,cv2.TM_CCOEFF_NORMED)

        loc = np.where( res >= threshold)

        for pt in loc[::-1]

            print "100/-"

            subprocess.call('espeak -ven+f3 -k5 -s130 '+cur100, shell=True)

            cur0=100

            cv2.rectangle(frame, pt, (pt[0] + w14, pt[1] + h14), (0,0,255), 1)

            cv2.putText(frame,'Rs/- 100', (pt[0], pt[1]),
cv2.FONT_HERSHEY_SIMPLEX,0.6,2,1)

        res = cv2.matchTemplate(img_gray,temp100d,cv2.TM_CCOEFF_NORMED)

        loc = np.where( res >= threshold)

        for pt in loc[::-1]

            print "100/-"

            subprocess.call('espeak -ven+f3 -k5 -s130 '+cur100, shell=True)

            cur0=100

            cv2.rectangle(frame, pt, (pt[0] + w15, pt[1] + h15), (0,0,255), 1)

            cv2.putText(frame,'Rs/- 100', (pt[0], pt[1]),
cv2.FONT_HERSHEY_SIMPLEX,0.6,2,1)

        res = cv2.matchTemplate(img_gray,temp500a,cv2.TM_CCOEFF_NORMED)

        loc = np.where( res >= threshold)

        for pt in loc[::-1]

            print "500/-"

            subprocess.call('espeak -ven+f3 -k5 -s130 '+cur500, shell=True)

            cur0=500

            cv2.rectangle(frame, pt, (pt[0] + w16, pt[1] + h16), (0,0,255), 1)

```

```

        cv2.putText(frame,'Rs/- 500', (pt[0], pt[1]),
cv2.FONT_HERSHEY_SIMPLEX,0.6,2,1)

    res =
cv2.matchTemplate(img_gray,temp500b,cv2.TM_CCOEFF_NORMED)

    loc = np.where( res >= threshold)

    for pt in loc[::-1]

        print "500/-"

        subprocess.call('espeak -ven+f3 -k5 -s130 '+cur500, shell=True)

        cur0=500

        cv2.rectangle(frame, pt, (pt[0] + w17, pt[1] + h17), (0,0,255), 1)

        cv2.putText(frame,'Rs/- 500', (pt[0], pt[1]),
cv2.FONT_HERSHEY_SIMPLEX,0.6,2,1)

    res =
cv2.matchTemplate(img_gray,temp500c,cv2.TM_CCOEFF_NORMED)

    loc = np.where( res >= threshold)

    for pt in loc[::-1]

        print "500/-"

        subprocess.call('espeak -ven+f3 -k5 -s130 '+cur500, shell=True)

        cur0=500

        cv2.rectangle(frame, pt, (pt[0] + w18, pt[1] + h18), (0,0,255), 1)

        cv2.putText(frame,'Rs/- 500', (pt[0], pt[1]),
cv2.FONT_HERSHEY_SIMPLEX,0.6,2,1)

    res =
cv2.matchTemplate(img_gray,temp500d,cv2.TM_CCOEFF_NORMED)

    loc = np.where( res >= threshold)

    for pt in loc[::-1]

        print "500/-"

        subprocess.call('espeak -ven+f3 -k5 -s130 '+cur500, shell=True)

        cur0=500

        cv2.rectangle(frame, pt, (pt[0] + w19, pt[1] + h19), (0,0,255), 1)

```

```

        cv2.putText(frame,'Rs/- 500', (pt[0], pt[1]),
cv2.FONT_HERSHEY_SIMPLEX,0.6,2,1)

    cv2.imshow('Result', frame)

    cv2.waitKey(1)
print "end code"
cv2.destroyAllWindows()

elif a == 5:
    vc = cv2.VideoCapture(0)
    ret , frame = vc.read()
    frame=cv2.resize(frame,(320,240))
    cv2.imwrite("picture.tif", frame)
    #cv2.imshow('image',frame)
    im = Image.open('picture.tif')
    text = image_to_string(im)
    print text

# main() function
def main():
    # use sys.argv if needed
    print 'running speech-test.py...'
    engine = pyttsx.init()
    engine.setProperty('rate', 120)
    voices = engine.getProperty('voices')
    engine.setProperty('voice', voices[9].id)
    str = text

    if len(sys.argv) > 1:
        str = sys.argv[1]

    engine.say(str)
    engine.runAndWait()

```



```

# call main

if __name__ == '__main__':
    main()

elif a == 6:
    subprocess.call('espeak -ven+f3 -k5 -s130 '+v3, shell=True)

    face_cascade =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

    eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

    vc = cv2.VideoCapture(0)

    _, frame = vc.read()

    img = cv2.resize(frame, (320,240))

    #img = cv2.imread('/home/pi/Desktop/xfiles4.jpg')

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    faces = face_cascade.detectMultiScale(gray, 1.3, 5) #(gray, 1.3, 5)

    print "Found_"+str(len(faces))+"_face(s)"

    x1 = "found_"

    x2 = "faces"

    subprocess.call('espeak -ven+f3 -k5 -s130 '+x1+str(len(faces))+x2,
shell=True)

elif a == 0:
    pathdir='prove/'

    model = PredictableModel(Fisherfaces(), NearestNeighbor())

    vc=cv2.VideoCapture(0)

    face_cascade = cv2.CascadeClassifier('face.xml')

    a="0"

    def read_images(path, sz=(256,256)):

        c = 0

        X,y = [], []

```

```

folder_names = []
for dirname, dirnames, filenames in os.walk(path):
    for subdirname in dirnames:
        folder_names.append(subdirname)
        subject_path = os.path.join(dirname, subdirname)
        for filename in os.listdir(subject_path):
            try:
                im = cv2.imread(os.path.join(subject_path, filename),
cv2.imread_grayscale)
                if (sz is not None):
                    im = cv2.resize(im, sz)
                    X.append(np.asarray(im, dtype=np.uint8))
                    y.append(c)
            except IOError, (errno, strerror):
                print "I/O error({0}): {1}".format(errno, strerror)
            except:
                print "Unexpected error:", sys.exc_info()[0]
                raise
            c = c+1
    return [X,y,folder_names]
[X,y,subject_names] = read_images(pathdir)
list_of_labels = list(xrange(max(y)+1))
subject_dictionary = dict(zip(list_of_labels, subject_names))
model.compute(X,y)
timeout_start = time.time()
timeout=4
while (time.time() < timeout_start + timeout):
    rval, frame = vc.read()

```

```

img = cv2.resize(frame, (320,240))
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.2, 3)
for (x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),1)
    sampleImage = gray[y:y+h, x:x+w]
    sampleImage = cv2.resize(sampleImage, (256,256))
    [ predicted_label, generic_classifier_output] =
model.predict(sampleImage)
    if int(generic_classifier_output['distances']) <= 620:
        a= str(subject_dictionary[predicted_label])
        print(a)
        print int(generic_classifier_output['distances']), " ", predicted_label
    if cv2.waitKey(10) == 27:
        break
    if len(a) > 2:
        subprocess.call('espeak -ven+f3 -k5 -s80
'+str(dictionary[predicted_label]), shell=True)
    else :
        print "a"
    cv2.destroyAllWindows()
    vc.release()
#elif a == 8:
#elif a == 9
elif a == 7:
subprocess.call('espeak -ven+f3 -k5 -s130 '+t1, shell=True)
time.sleep(2)
cur0=0

```

```
y = ser.readline()[:-2]
hb = ser.readline()
tmp = ser.readline()
cv2.destroyAllWindows()
while (GPIO.input(ROW[i]) == 0):
    pass
    GPIO.output(COL[j],1)
except KeyboardInterrupt:
GPIO.cleanup()
```

APPENDIX E



Figure 29: Final Deliverable

BIBLIOGRAPHY

BIBLIOGRAPHY

- [¹] Visual impairment and blindness. (n.d.). Retrieved October 18, 2015, from <http://www.who.int/mediacentre/factsheets/fs282/en/>
- [²] Batavia, P., & Singh, S. (2001). Obstacle detection using adaptive color segmentation and color stereo homography. Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164), 1, 705-710. <http://dx.doi.org/10.1109/ROBOT.2001.932633>
- [³] Kim, C., & Song, B. (2007). Design of a wearable walking-guide system for the blind. Proceedings of the 1st International Convention on Rehabilitation Engineering & Assistive Technology in Conjunction with 1st Tan Tock Seng Hospital Neurorehabilitation Meeting - I-CREATe '07, 118-122. doi:10.1145/1328491.1328523
- [⁴] Innet, S.; Ritnoom, N., "An application of infrared sensors for electronic white stick," in *Intelligent Signal Processing and Communications Systems, 2008. ISPACS 2008. International Symposium on*, vol., no., pp.1-4, 8-11 Feb. 2008
doi: 10.1109/ISPACS.2009.4806716
URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4806716&isnumber=4806653>
- [⁵] Josh, H., Yong, B., & Kleeman, L. (2011). A Real-Time and Portable Bionic Eye Simulator. Biomedical Engineering Systems and Technologies Communications in Computer and Information Science, 51-67. Retrieved October 18, 2015, from <http://www.araa.asn.au/acra/acra2011/papers/pap166.pdf>
- [⁶] Jain, P., & Awasthy, M. (2014). Automatic Obstacle Detection using Image Segmentation. International Journal of Emerging Technology and Advanced Engineering, 4(3). Retrieved October 18, 2015, from www.ijetae.com

[7] Template Matching. (n.d.). Retrieved June 19, 2016, from http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html

[8] OpenCV: Template Matching. (n.d.). Retrieved June 19, 2016, from http://docs.opencv.org/master/d4/dc6/tutorial_py_template_matching.html#gsc.tab=0