

**TABLE OF CONTENTS:**

1. Synopsis/Abstract/Summary………………………………1
2. List of Illustrations………………………………………………2
3. Introduction……………….………………………………………5
4. Main text……………………………………………………………6

4A. Findings……………………………………………………………...9

4B. Future Scope/Future works……………………………….9

4C. Conclusions………………………………………………………...9

1. Attachments……………………………………………………….10
2. Glossary………………………………………………………………11

**SYNOPSIS:**

There are 37 million people across the globe who are fully and partially blind . Most of them lead there life by facing many hardships mainly while moving from place to place as they don’t know the directions and hurdles in there path. So, these people need a device that could continuously assist them with the path and hurdles in there path by providing on ground spontaneous assistance.

To the best of my knowledge there are no such products in the market that are designed purely for this use, But software’s like Google assistant and Google lens are useful to us in many purposes which could slightly solve the problem of blind, they cannot actually navigate a person through a desired location safely they can only estimate the path and distance.

My project consists of real time object detection system and object to person distance calculation system along its postioned direction and along with offline voice assistant and few additional features of providing temperature. The Google lens and google assist are only available in smart phones which costs around 15000-50000 INR which could not be a effiective use for this problem and even a poor man can’t afford it. Our project addresses these issues in a simple, efficient and cost- effective way

.



**ABSTRACT:**

We came up with a simple and cost-effective solution to solve this problem faced by the Blind people. We designed Artificial vision based voice assist system that has the following features

1. Conveys the info about hurdle in front of user by voice output.

2. Can detect the distance from the hurdle and assist in advance when suitable command given to voice assists.

3. Can assist the user with the path to move and conveys by voice out

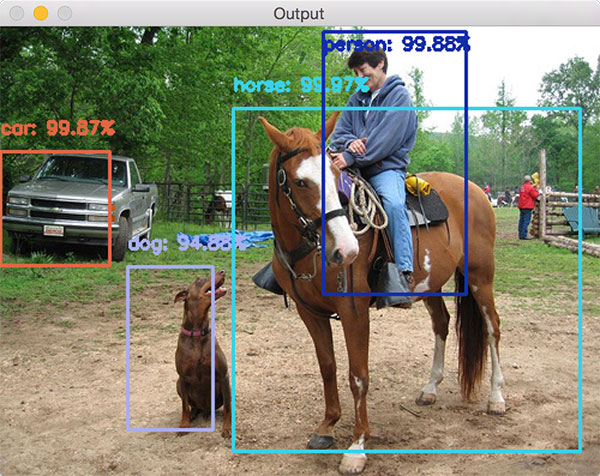
4. Can detect temperature and conveys by voice out when suitable command given to voice assist.

This process is done by using a good resolute camera(web cam) in which Open Computer vision is used up on the basic mobile netSSD data set which contain images of particular object stored in feasible fashion. We used the C-NN (congugational –neural network)algorithm for the feature excraction from the images and matching it to provided spontaneous image. An ultra sonic sensor is used to calculate the distance from the object infront of user .The clincic role of Open cv in the project is to provided a frame for work the image and detect. MPU6050 used to locate the objects exactly based on the head position .

By following above process a well efficient system that purely focused on blind people is developed in a cost effective and efficient manner.

**List of Illustrations:**

A.**Object detection System** :

****

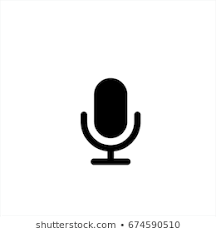
Object is detected with efficiency label and name of the object detected will be given voice out

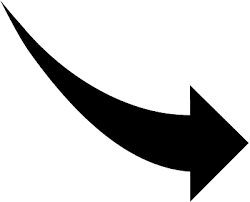


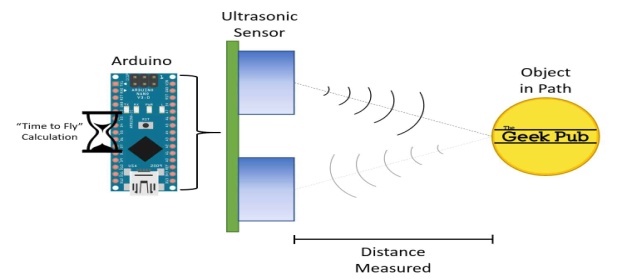
Person ahead distance 5 m

“AIRIS WHAT IS IN FRONT”

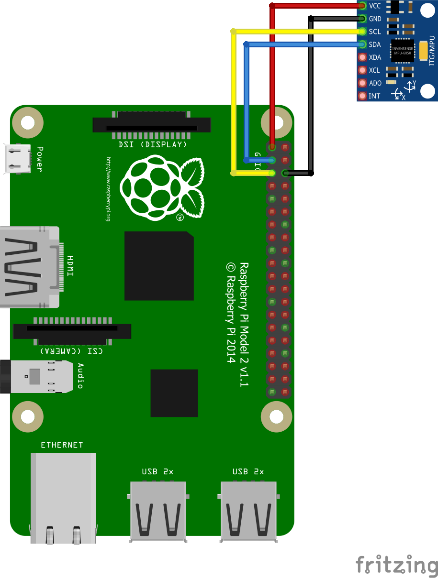
B.**Distance detection and voice assist**:

.

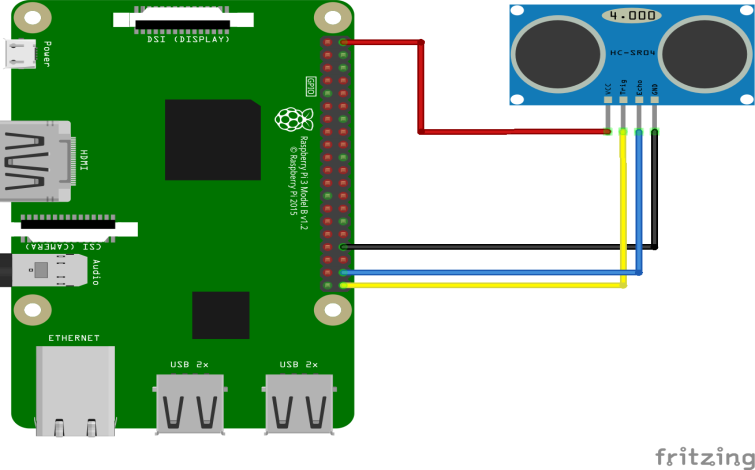
C . Wring and setting of Prototype:

MPU6050 connection with PI WEBCAM connection with pi



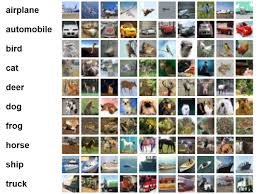
The MPU6050,ultra sonic and WEB cam are connected to the RaspberryPi 3B+ using Jumpers and a Power Bank is used as a Power Supply.

****

**Ultrasonic connection with pi**

**D.Image data set :**

The below image data set is the collection of different images affixed with a target c-nn ml algorithm is applied to the data set for feature exctration

****



**4.Introduction:**

There are many people across the globe who are fully and partially blind . Most of them lead there life by facing many hardships mainly while moving from place to place as they don’t know the directions and hurdles in there path. The solution will be to provide an artificial vision based assist for the blind people.This assist will be helping in detection of object and obstacles present near the blind person and convey themwith a voice out. Which could help them to even move from place to place by guiding them with voice.

**Main Text:**

The components used in this project are:

**HARDWARE:**

1. **RASPBERRYPi 3B+ MODEL** :

* The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range,boasting a 64-bit quad core processor running at 1.4GHz, dual-band 2.4GHz and 5GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT.
* The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced, wireless LAN compliance testing,improving both cost and time to market.
* The Raspberry Pi 3 Model B+ maintains the same mechanical footprint as both the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B.

**SPECIFICATIONS :**

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz

- 1GB LPDDR2 SDRAM

- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE

- Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)

- Extended 40-pin GPIO header

- Full-size HDMI

- 4 USB 2.0 ports

- CSI camera port for connecting a Raspberry Pi camera

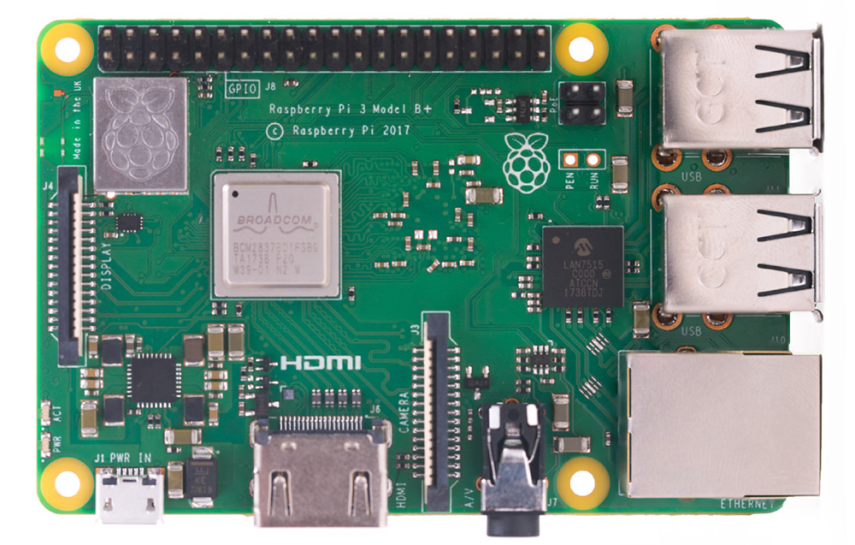
- DSI display port for connecting a Raspberry Pi touchscreen display

- 4-pole stereo output and composite video port

- Micro SD port for loading your operating system and storing data

- 5V/2.5A DC power input

- Power-over-Ethernet (PoE) support (requires separate PoE HAT)



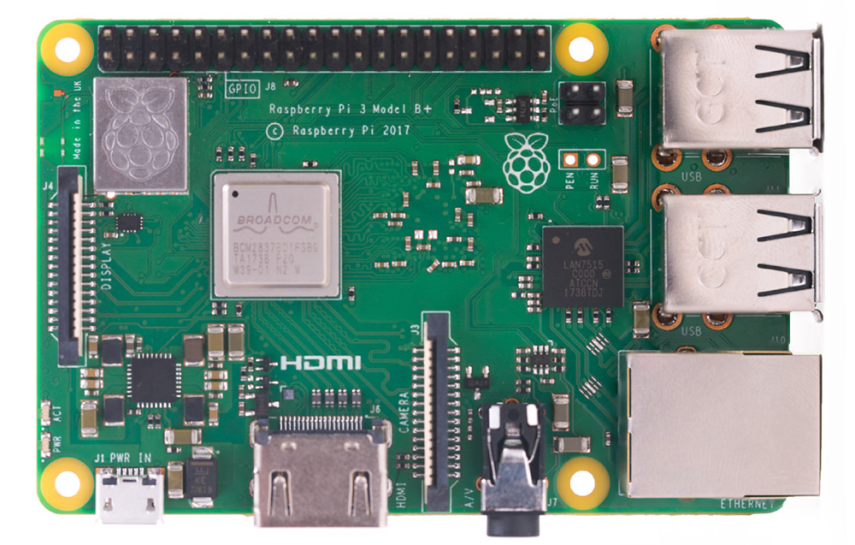


- 4-pole stereo output and composite video port

- Micro SD port for loading your operating system and storing data

- 5V/2.5A DC power input

- Power-over-Ethernet (PoE) support (requires separate PoE HAT)



1. **MPU6050(Acclerometer and Gyro Sensor):**

MPU6050 sensor module is complete 6-axis Motion Tracking Device. It combines 3-axis Gyroscope, 3-axis Accelerometer and Digital Motion Processor all in small package. Also, it has additional feature of on-chip Temperature sensor. It has I2C bus interface to communicate with the micro controllers or processors.

It has Auxiliary I2C bus to communicate with other sensor devices like 3-axis Magnetometer, Pressure sensor etc.

If 3-axis Magnetometer is connected to auxiliary I2C bus, then MPU6050 can provide complete 9-axis Motion Fusion output.

**3-Axis Gyroscope:**

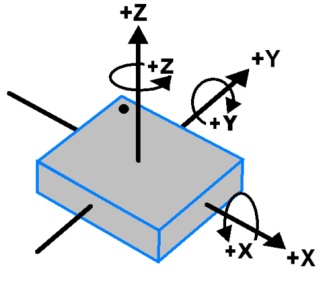
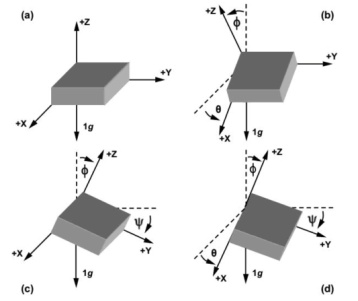
The MPU6050 consist of 3-axis Gyroscope with

Micro Electro Mechanical System(MEMS) technology.

It is used to detect rotational velocity along the X, Y, Z axes as shown in below fig.



**3-Axis Accelerometer:**



MPU6050 consist 3-axis Accelerometer with Micro Electro Mechanical (MEMs) technology. It used to detect angle of tilt or inclination along the X, Y and Z axes as shown in below figure.

**Calculations:**

Gyroscope and accelerometer sensor data of MPU6050 module consists of 16-bit raw data in 2’s complement form.

Temperature sensor data of MPU6050 module consists of 16-bit data (not in 2’s complement form).

Now suppose we have selected,

- Accelerometer full scale range of +/- 2g with Sensitivity Scale Factor of 16,384 LSB(Count)/g.

- Gyroscope full scale range of +/- 250 °/s with Sensitivity Scale Factor of 131 LSB (Count)/°/s.

then,

To get sensor raw data, we need to first perform 2’s complement on sensor data of Accelerometer and gyroscope.

After getting sensor raw data we can calculate acceleration and angular velocity by dividing sensor raw data with their sensitivity scale factor as follows,

Accelerometer values in g (g force):

Acceleration along the X axis = (Accelerometer X axis raw data/16384) g.

Acceleration along the Y axis = (Accelerometer Y axis raw data/16384) g.



Acceleration along the Z axis = (Accelerometer Z axis raw data/16384) g.

gyroscope values in °/s (degree per second):

Angular velocity along the X axis = (Gyroscope X axis raw data/131) °/s.

Angular velocity along the Y axis = (Gyroscope Y axis raw data/131) °/s.

Angular velocity along the Z axis = (Gyroscope Z axis raw data/131) °/s

### ULTRASONIC SENSOR:

### HC-SR04 Sensor Features

* Operating voltage: +5V
* Theoretical  Measuring Distance: 2cm to 450cm
* Practical Measuring Distance: 2cm to 80cm
* Accuracy: 3mm
* Measuring angle covered: <15°
* Operating Current: <15mA
* Operating Frequency: 40Hz

**Equivalent distance measuring Sensors:**

US transmitter Receiver pair, IR sensor module, IR sensor pair, IR Analog distance sensor,

### HC-SR04 Ultrasonic Sensor - Working

As shown above the **HC-SR04 Ultrasonic (US) sensor** is a 4 pin module, whose pin names are Vcc, Trigger, Echo and Ground respectively. This sensor is a very popular sensor used in many applications where measuring distance or sensing objects are required. The module has two eyes like projects in the front which forms the Ultrasonic transmitter and Receiver. The sensor works with the simple high school formula that

**Distance = Speed × Time**

The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when it gets objected by any material it gets reflected back toward the sensor this reflected wave is observed by the Ultrasonic receiver module as shown in the picture below





Now, to calculate the distance using the above formulae, we should know the Speed and time. Since we are using the Ultrasonic wave we know the universal speed of US wave at room conditions which is 330m/s. The circuitry inbuilt on the module will calculate the time taken for the US wave to come back and turns on the echo pin high for that same particular amount of time, this way we can also know the time taken. Now simply calculate the distance using a microcontroller or microprocessor.

**SOFTWARE:**

1. **PYTHON3 IDLE:**

Python is an open source programming language that was made to be easy-to-read and powerful. Python is an interpreted language. Interpreted languages do not need to be compiled to run. A program called an interpreter runs Python code on almost any kind of computer. This means that a programmer can change the code and quickly see the results. This also means Python is slower than a compiled language like C, because it is not running machine code directly.

Python is a good programming language for beginners. It is a high-level language, which means a programmer can focus on what to do instead of how to do it. Writing programs in Python takes less time than in some other languages.

Python drew inspiration from other programming languages like C, C++, Java, Perl, and Lisp.

Python is used by hundreds of thousands of programmers and is used in many places. Sometimes only Python code is used for a program, but most of the time it is used to do simple jobs while another programming language is used to do more complicated tasks.

Its standard library is made up of many functions that come with Python when it is installed. On the Internet there are many other libraries available that make it possible for the Python language to do more things. These libraries make it a powerful language it can do many different things





Some things that Python is often used for are:

Web development

Scientific programming

Desktop GUIs

Network programming

Game programming

**WORKING OF THE PROTOTYPE:**

This project uses RaspberryPi as a processor and Accelerometer and Gyro sensor MPU6050 sensor. It also uses a WEB cam for object detection Ultrasonic for distance caluclation.

The prototype works by detecting which object or obstacle before the blind

person based on objects shape ,colour, by the camera and that could also

calculate the distance between obstacle and the person and assists the

blind person with voice commands to make him safe from the obstacle. The device will be made portable,compartable to work efficiently. The

device will be developed using vision based image recognition techniques

which includes camera ,microcontrollers. The current temperature

surrounding him is conveyed . This device also conveys the direction along

with the name of the object and distance between him and the object. We use the combination of the mobile net ,single shot detector(SSD)and convolutional neural networks (CNN) for detecting the object , Here the CNN comes under the evolution of the deep learning , basically the input image is processed and classified from the data set consisting of the number of images using the above algorithms ,later the output and the distance from the object is converted to speech and conveyed through the micro phone. We use ultrasonic sensor for calculating distance and MPU sensor for calculating direction and temperature surrounding him and this is conveyed to him along with the object name.



**The code of our project in PYTHON3:**

# python pi\_object\_detection.py --prototxt MobileNetSSD\_deploy.prototxt.txt --model MobileNetSSD\_deploy.caffemodel

# import the necessary packages  
from imutils.video import VideoStream  
from imutils.video import FPS  
from multiprocessing import Process  
from multiprocessing import Queue  
import numpy as np  
import argparse  
import imutils  
import time  
import cv2

Import os

Import robot

mport RPi.GPIO as GPIO #Import GPIO library

import time #Import time library

GPIO.setmode(GPIO.BCM) #Set GPIO pin numbering

TRIG = 23 #Associate pin 23 to TRIG

ECHO = 24 #Associate pin 24 to ECHO

print "Distance measurement in progress"

GPIO.setup(TRIG,GPIO.OUT) #Set pin as GPIO out

GPIO.setup(ECHO,[GPIO.IN](http://GPIO.IN)) #Set pin as GPIO in

def robot(text) :  
    os.system("espeak ' " + text + " ' ")

font = cv2.FONT\_HERSHEY\_SIMPLEX

# Capture from the default deivce

cap = cv2.VideoCapture(0)

def classify\_frame(net, inputQueue, outputQueue):  
# keep looping  
 while True:  
     # Caputure a single frame  
    ret, huge\_frame = cap.read()  
    frame = cv2.resize(huge\_frame, (0,0), fx=0.5, fy=0.5, interpolation=cv2.INTER\_NEAREST)  
            # check to see if there is a frame in our input queue  
    if not inputQueue.empty():

        # grab the frame from the input queue, resize it, and  
        # construct a blob from it  
        frame = inputQueue.get()  
        frame = cv2.resize(frame, (300, 300))  
        blob = cv2.dnn.blobFromImage(frame, 0.007843,(300, 300), 127.5)



        # set the blob as input to our deep learning object  
        # detector and obtain the detections  
        net.setInput(blob)  
        detections = net.forward()  
        # write the detections to the output queue  
        outputQueue.put(detections)  
        # construct the argument parse and parse the arguments  
        ap = argparse.ArgumentParser()  
        ap.add\_argument("-p", "--prototxt", required=True,help="path to Caffe 'deploy' prototxt file")  
        ap.add\_argument("-m", "--model", required=True,help="path to Caffe pre-trained model")  
        ap.add\_argument("-c", "--confidence", type=float, default=0.2,help="minimum probability to filter weak detections")  
        args = vars(ap.parse\_args())  
        # initialize the list of class labels MobileNet SSD was trained to  
        # detect, then generate a set of bounding box colors for each class  
        CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat","bottle", "bus", "car", "cat", "chair", "cow", "diningtable","dog", "horse", "motorbike", "person", "pottedplant", "sheep","sofa", "train", "tvmonitor"]  
        COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))  
        # load our serialized model from disk  
        print("[INFO] loading model...")  
        net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])  
        # initialize the input queue (frames), output queue (detections),  
        # and the list of actual detections returned by the child process  
        inputQueue = Queue(maxsize=1)  
        outputQueue = Queue(maxsize=1)  
        detections = None  
        # construct a child process \*indepedent\* from our main process of  
        # execution  
        print("[INFO] starting process...")  
        p = Process(target=classify\_frame, args=(net, inputQueue,outputQueue,))  
        p.daemon = True  
        p.start()  
        # initialize the video stream, allow the cammera sensor to warmup,  
        # and initialize the FPS counter  
        print("[INFO] starting video stream...")  
        vs = VideoStream(src=0).start()  
        # vs = VideoStream(usePiCamera=True).start()  
        time.sleep(2.0)  
        fps = FPS().start()  
        # loop over the frames from the video stream  
 while True:

GPIO.output(TRIG, False) #Set TRIG as LOW

print "Waitng For Sensor To Settle"



time.sleep(2) #Delay of 2 seconds

GPIO.output(TRIG, True) #Set TRIG as HIGH

time.sleep(0.00001) #Delay of 0.00001 seconds

GPIO.output(TRIG, False) #Set TRIG as LOW

while GPIO.input(ECHO)==0: #Check whether the ECHO is LOW

pulse\_start = time.time() #Saves the last known time of LOW pulse

while GPIO.input(ECHO)==1: #Check whether the ECHO is HIGH

pulse\_end = time.time() #Saves the last known time o fH f HIGH pulse

pulse\_duration = pulse\_end - pulse\_start #Get pulse duration to a variable

distance = pulse\_duration \* 17150 #Multiply pulse duration by 17150 t to get distancedistance = round(distance, 2) #Round to two decimal points

if distance > 2 and distance < 400: #Check whether the distance is w Within rangeprint "Distance:",distance - 0.5,"cm" #Print distance with 0.5 cm c a alibration

else:

print "Out Of Range"

# grab the frame from the threaded video stream, resize it, and  
# grab its imensions  
  frame = vs.read()  
  frame = imutils.resize(frame, width=400)  
(fH, fW) = frame.shape[:2]

# if the input queue \*is\* empty, give the current frame to  
# classify  
if inputQueue.empty():  
 inputQueue.put(frame)

# if the output queue \*is not\* empty, grab the detections  
if not outputQueue.empty():  
 detections = outputQueue.get()

# check to see if our detectios are not None (and if so, we'll  
# draw the detections on the frame)  
if detections is not None:  
    # loop over the detections  
    for i in np.arange(0, detections.shape[2]):  
        # extract the confidence (i.e., probability) associated  
        # with the prediction  
        confidence = detections[0, 0, i, 2]  
        # filter out weak detections by ensuring the `confidence`  
        # is greater than the minimum confidence  
        if confidence < args["confidence"]:  
            continue  
        # otherwise, extract the index of the class label from  
        # the `detections`, then compute the (x, y)-coordinates  
# of the bounding box for the object  
            idx = int(detections[0, 0, i, 1])



|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | |  |                dims = np.array([fW, fH, fW, fH])             box = detections[0, 0, i, 3:7] \* dims             (startX, startY, endX, endY) = box.astype("int")  # draw the prediction on the frame             label = "{}: {:.2f}%".format(CLASSES[idx],confidence \* 100)             cv2.rectangle(frame, (startX, startY), (endX, endY),COLORS[idx], 2)             y = startY - 15 if startY - 15 > 15 else startY + 15             cv2.putText(frame, label, (startX, y),cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, COLORS[idx], 2)  # show the output frame             cv2.imshow("Frame", frame)             key = cv2.waitKey(1) & 0xFF  # if the `q` key was pressed, break from the loop  If d==”ok airis”: # voice assist  robot(label)  If d==”distance”:  robot(“distance”)     if key == ord("q"):                 break  # update the FPS counter fps.update()  # stop the timer and display FPS information fps.stop() print("[INFO] elapsed time: {:.2f}".format(fps.elapsed())) print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))  # do a bit of cleanup cv2.destroyAllWindows() vs.stop()  Code of the project is explained by commenting its description.   1. **Findings:**   It is found that this sort of devices as awide market in the outer world .people are ready to buy if we could make the prototype effient.For perfect object detection the data set should be effcient.According to many surveys blind people have strong listing power which could used as a aspect of technology develeopment in that area. We feel that our device was effecient enough to reach the market needs with simple upliftments.Its the cost effient and best way possible to solve the problem .     |  |  | | --- | --- | |  |  | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | |  | |  |  |



**b.Future Upgrades for our Project:**

* Object detection: Development of dataset is makes product more effective and finding exacts positions and 360 degree vision could uplift the project.
* Making the device embedded and making it fit into a googl.

**c. Conclusion :**

We found out that our project is a lot cost effective than any of the other ones. Along with that we have a distance and object detecting system and an Offline Voice Assistant. By this project we sought out to solve the main problems faced by the blind people making their life simple and happy. We would further like to develop the device by solving further problems of the blind people and would try to make it a product in the coming future.

**6. REFERENCES:**

[**https://link.springer.com/chapter/10.1007/978-3-642-36124-1\_23**](https://link.springer.com/chapter/10.1007/978-3-642-36124-1_23)

**https://ieeexplore.ieee.org/document/6141394**

**WORKING PROTOTYPE:**





**USECASE DIAGRAM:**



**8.GLOSSARY**:

* MPU6050 – An Accelerometer and Gyro Sensor
* ML – Machine Learning
* RaspberryPi – A processor
* WEB cam .
* Ultrasonic sensor