భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Biomedical Informatics Lab
Department of
Biomedical Engineering

Virtual Vision

Project Report

Version 1

Datum: 20.10.2022

Project submitted by-

Ramanathan Annamalai	BM20BTECH11011
Himanshu Singh	BM22MTECH11004
Gaurav Kumar Singh	BM22MTECH11002
Sai Prajoth Mallyala	BM20BTECH11008

Acknowledgement

We would like to thank Dr. Nagarajan Ganapathy Sir for his constant support and guidance. We express our gratitude to Saransh Sir from the Biomedical Instrumentation Lab and the Academic Section for procurement of the requisite electronic components and lastly all our friends.

TABLE OF CONTENTS ▪

1. Abstract
2. Introduction
3. Materials
4. Methods
5. Results & Discussion
6. Future Work
7. References

Abstract

According to a report by the World Health Organization, there are currently 284 million people in the world who are visually impaired, and 39 million people are blind and this number is increasing every year. Navigation and walking is the biggest challenge that a blind person faces . As of now, Blind people mainly depend on walking sticks (or Canes) for their navigation. There have been many modifications in the cane stick to make it more efficient but still the cane stick has many limitations .

We see a greater possibility to empower the people with visual disabilities by giving them a much more natural sense of vision through the other modes of senses, mainly touch through haptic technology. Through this project an attempt is made towards this approach to empower people with visual impairments.In this project Vision sensor is used to take the input of the visual field which are then converted to a framework which divided the visual field of space into different segments by using OpenCV which give the command to linear resonant actuator to navigate the user.

We see a high potential of this kind of technology and we aim to use this linear resonant actuator in wearable shoes to guide the user by giving feedback through haptic technology on their feet. further we aim to research the various ways we can stimulate the vision regions of the brain and make a subtle environment exclusive for the visually impaired persons. We also aim to use modern technology like spatial audio to enhance the user experience.

List of Figures -

Fig. no.	Title	Page no.
1	NodeMCU ESP 32 board	8
2	linear resonant actuator	9
3	Framework created by openCV	20
4	Equivalent Circuit Diagram	21
5	Entire prototype	22
6	prototype with the team	22
7a	The divisions in the frame received from the camera	23
7b	The positions of the actuators	23

Introduction:

As of now, Blind people mainly depend on walking sticks (or Canes) for their navigation. Based on the physical properties of materials around them and their interaction with the cane the blind approximate the world around them. There have been a lot of innovations which make the use of the cane more efficient. There have also been construction designs that accommodate features which help the blind and cane model. We see a greater possibility to empower the people with visual disabilities by giving them a much more natural sense of vision through the other modes of senses, mainly touch. With a world of spatial audio and haptic technology that enhance the user experience, it only motivates us to think why these patterns of artificially simulating natural feelings cannot help to provide vision. We aim to use existing technology to process the world using cameras and then provide feedback to the person using the product, a wearable device, which can naturally give an understanding of the physical world around them. We aim to research the various ways we can stimulate the vision regions of the brain and make a subtle environment exclusive for the blind. This way we can overcome the barrier of visual restrictions of people. The more primitive ways of feedback include voice narration, vibrations and beeps to notify the user about the surroundings but these are still very basic. We want to move to a more realistic model and the path to achieving artificial simulations of the natural world makes the project very interesting and novel.

Some of the existing work related to vision assistance to visual impaired persons are:
1. Sixth Sense Technology:
It was developed by Pranav Mistry, a Ph.D. student in the Fluid Interfaces Group at the MIT Media Lab. Sixth Sense is a wearable gesture based device that augments the physical world with digital information and lets people use natural hand gestures to interact with that information. Sixth Sense Technology is a mini-projector coupled with a camera and a cellphone which acts as the computer and connected to the Cloud, all the information stored on the web. Sixth Sense can also obey hand gestures. The camera recognizes objects around a person instantly, with the micro-projector overlaying the information on any surface, including the object itself or hand. Also can access or manipulate the information using fingers. The device has a huge number of applications - (i) make a call by extending your hand in front of the projector and numbers will appear to click. (ii)know the time by drawing a circle on your wrist and a watch will appear. (iii)Take a photo by just making a square with fingers, highlighting what you want to frame, and the system will make the photo which can later be organized with the others using your own hands over the air. (iv)Can be used for

drawing, reading newspapers and conveying visual information of your ticket audibly. It is portable and easy to carry as it can be worn around the neck.

2. Project Ray

Project Ray aims to deliver all the benefits of the smartphone revolution to the visually impaired community. Project ray has come up with the Ray Vision app, coupled with patented Ray Click technology, it is a specially designed user experience that provides visually impaired individuals with full accessibility, instantly for any touch device.

3. YOLO

It is an algorithm that uses a neural network to provide real-time object detection. Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images.

4. OrCam MyEye Pro

It is a device that is attached to a glass and can be activated by voice to and convey visual information audibly to read to you text from a book, smartphone screen or any other surface, recognize faces. This technology is related to our work as a vision sensor is also used in our project.

5. Stereolab ZED camera

The ZED camera is a binocular vision system that can be used to provide a 3D perception of the world. It can be applied in autonomous robot navigation, virtual reality, tracking, motion analysis. This vision sensor can be used in development of devices to aid vision impaired persons.

Materials:

1. NodeMCU ESP 32
   The NodeMCU ESP32 board (in some cases also known as ESP32-DevkitC) is fully supported by ESPHome.

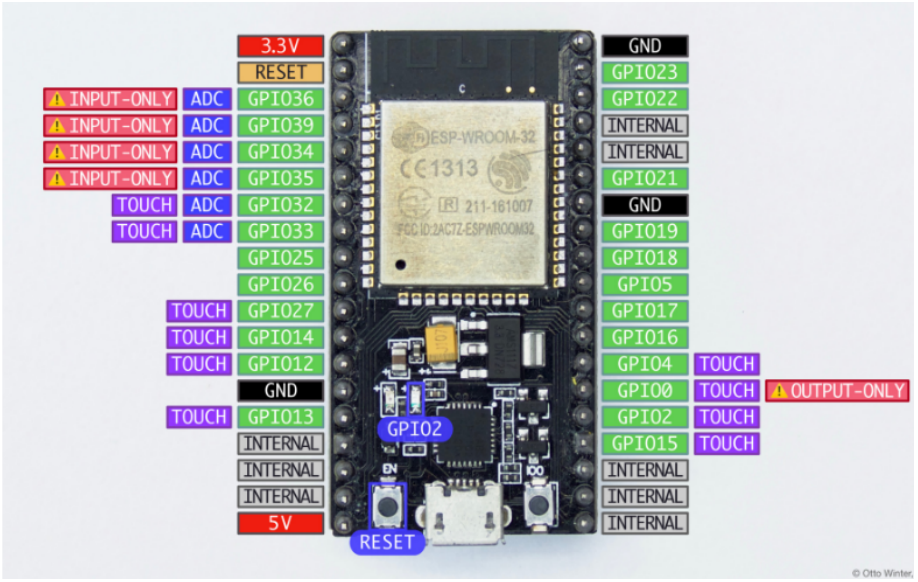

Fig 1- NodeMCU ESP32 board

2. linear resonant actuator
   A linear resonant actuator is a precision vibration motor that produces an oscillating force across a single axis. Unlike a DC eccentric rotating mass (ERM) motor, a linear vibration motor relies on an AC voltage to drive a voice coil pressed against a moving mass that is connected to a spring.

Fig2 - linear resonant actuator

3. LED
   A light-emitting diode (LED) is a semiconductor light source that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons.

4. Breadboard
   A breadboard, solderless breadboard, protoboard, or terminal array board is a construction base used to build semi-permanent prototypes of electronic circuits.

5. Jumper wires

6. OpenCV modules
   OpenCV is a Python library that allows you to perform image processing and computer vision tasks. It provides a wide range of features, including object detection, face recognition, and tracking.

Method:

1. **Processing the video (Open CV):**

Import and export to google drive:

```
from google.colab import drive
drive.mount('/content/drive')
```

Importing Libraries

```
import os
import cv2
import numpy as np
import tensorflow as tf
import sys
import time
!git clone https://github.com/tensorflow/models.git
!apt-get -qq install libprotobuf-java protobuf-compiler
!protoc
./models/research/object_detection/protos/string_int_label_m
ap.proto --python_out=.
!cp -R models/research/object_detection/ object_detection/
!rm -rf models
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
```

```
import tensorflow as tf
import zipfile
from collections import defaultdict
from io import StringIO
import matplotlib.pyplot as plt
from PIL import Image
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as
vis_util
```

Uploading the input file in video format

```
from google.colab import files
def upload_files():
  from google.colab import files
  uploaded = files.upload()
  for k, v in uploaded.items():
    open(k, 'wb').write(v)
  return list(uploaded.keys())
upload_files()
```

Detection Model :

```
MODEL_NAME = 'faster_rcnn_inception_v2_coco_2018_01_28'
#MODEL_NAME = 'ssd_inception_v2_coco_2017_11_17'
# high accuracy but very slow
# MODEL_NAME = 'faster_rcnn_resnet101_coco_2017_11_08'
MODEL_FILE = MODEL_NAME + '.tar.gz'
DOWNLOAD_BASE =
'http://download.tensorflow.org/models/object_detection/'
# Path to frozen detection graph. This is the actual model
that is used for the object detection.
PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'
# List of the strings that is used to add correct label for
each box.
PATH_TO_LABELS = os.path.join('object_detection/data',
'mscoco_label_map.pbtxt')
NUM_CLASSES = 10
opener = urllib.request.URLopener()
opener.retrieve(DOWNLOAD_BASE + MODEL_FILE, MODEL_FILE)
tar_file = tarfile.open(MODEL_FILE)
for file in tar_file.getmembers():
  file_name = os.path.basename(file.name)
  if 'frozen_inference_graph.pb' in file_name:
```

```
        tar_file.extract(file, os.getcwd())

#created the detection graph for the location -
detection_graph = tf.Graph()
with detection_graph.as_default():
  od_graph_def =tf.compat.v1.GraphDef()
  with tf.io.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
    serialized_graph = fid.read()
    od_graph_def.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph_def, name='')

label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories =
label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)
category_index =
label_map_util.create_category_index(categories)
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.compat.v1.GraphDef()
    with tf.io.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')
sess = tf.compat.v1.Session(graph=detection_graph)
# Define input and output tensors (i.e. data) for the object
detection classifier
# Input tensor is the image
image_tensor =
detection_graph.get_tensor_by_name('image_tensor:0')
# Output tensors are the detection boxes, scores, and
classes
# Each box represents a part of the image where a particular
object was detected
detection_boxes =
detection_graph.get_tensor_by_name('detection_boxes:0')
# Each score represents level of confidence for each of the
objects.
# The score is shown on the result image, together with the
class label.
```

```
detection_scores =
detection_graph.get_tensor_by_name('detection_scores:0')
detection_classes =
detection_graph.get_tensor_by_name('detection_classes:0')
# Number of objects detected
num_detections =
detection_graph.get_tensor_by_name('num_detections:0')

def region_of_interest(img, vertices):
    mask = np.zeros_like(img)
    if len(img.shape) > 2:
        channel_count = img.shape[2]
        ignore_mask_color = (255,) * channel_count
    else:
        ignore_mask_color = 255
    cv2.fillPoly(mask, vertices, ignore_mask_color)
    masked_image = cv2.bitwise_and(img, mask)
    return masked_image
    cv2.imshow("Image",masked_image)
```

Saving the output file in video format and text format:

```
video = cv2.VideoCapture("test1.mp4")
fourcc = cv2.VideoWriter_fourcc(*'XVID')
frame_width = int(video.get(3))
frame_height = int(video.get(4))
# out = cv2.VideoWriter('output.avi',fourcc, 20.0,
(640,480))
# out =
cv2.VideoWriter('outpy.mp4',cv2.VideoWriter_fourcc('M','J','
P','G'), 20, (frame_width,frame_height))
#out =
cv2.VideoWriter('output_video.avi',cv2.VideoWriter_fourcc('M
','J','P','G'), 20 , (frame_width,frame_height))
out =
cv2.VideoWriter('/content/drive/test1_out.mp4',cv2.VideoWrit
er_fourcc('M','J','P','G'), 90 , (frame_width,frame_height))
try:

  while(video.isOpened()):
      #defining the bouandary of the boxes -
      ret, frame = video.read()
      stime = time.time()
```

```
        objects = []
        class_str = ""
        frame_width = frame.shape[0]
        frame_height = frame.shape[1]
        rows, cols = frame.shape[:2]
        left_boundary = [int(cols*0.40), int(rows*0.95)]
        left_boundary_top = [int(cols*0.40), int(rows*0.20)]
        right_boundary = [int(cols*0.60), int(rows*0.95)]
        right_boundary_top = [int(cols*0.60), int(rows*0.20)]
        bottom_left  = [int(cols*0.20), int(rows*0.95)]
        top_left     = [int(cols*0.20), int(rows*0.20)]
        bottom_right = [int(cols*0.80), int(rows*0.95)]
        top_right    = [int(cols*0.80), int(rows*0.20)]
        vertices = np.array([[bottom_left, top_left,
top_right, bottom_right]], dtype=np.int32)
        cv2.line(frame,tuple(bottom_left),tuple(bottom_right),
(255, 0, 0), 5)
        cv2.line(frame,tuple(bottom_right),tuple(top_right),
(255, 0, 0), 5)
        cv2.line(frame,tuple(top_left),tuple(bottom_left),
(255, 0, 0), 5)
        cv2.line(frame,tuple(top_left),tuple(top_right), (255,
0, 0), 5)
        copied = np.copy(frame)
        interested=region_of_interest(copied,vertices)
        frame_expanded = np.expand_dims(interested, axis=0)

        (boxes, scores, classes, num) = sess.run(
            [detection_boxes, detection_scores,
detection_classes, num_detections],
            feed_dict={image_tensor: frame_expanded})
        vis_util.visualize_boxes_and_labels_on_image_array(
            frame,
            np.squeeze(boxes),
            np.squeeze(classes).astype(np.int32),
            np.squeeze(scores),
            category_index,
            use_normalized_coordinates=True,
            line_thickness=8,
            min_score_thresh=0.78)
        print(frame_width,frame_height)
```

```
        ymin = int((boxes[0][0][0]*frame_width))
        xmin = int((boxes[0][0][1]*frame_height))
        ymax = int((boxes[0][0][2]*frame_width))
        xmax = int((boxes[0][0][3]*frame_height))
        Result = np.array(frame[ymin:ymax,xmin:xmax])

        ymin_str='y min  = %.2f '%(ymin)
        ymax_str='y max  = %.2f '%(ymax)
        xmin_str='x min  = %.2f '%(xmin)
        xmax_str='x max  = %.2f '%(xmax)

        cv2.putText(frame,ymin_str, (50,
50),cv2.FONT_HERSHEY_SIMPLEX,0.6,(255,0,0),2)
        cv2.putText(frame,ymax_str, (50,
70),cv2.FONT_HERSHEY_SIMPLEX,0.6,(255,0,0),2)
        cv2.putText(frame,xmin_str, (50,
90),cv2.FONT_HERSHEY_SIMPLEX,0.6,(255,0,0),2)
        cv2.putText(frame,xmax_str, (50,
110),cv2.FONT_HERSHEY_SIMPLEX,0.6,(255,0,0),2)
        print(scores.max())

        #print("left_boundary[0],right_boundary[0] :",
left_boundary[0], right_boundary[0])
        #print("left_boundary[1],right_boundary[1] :",
left_boundary[1], right_boundary[1])
        #print("xmin, xmax :", xmin, xmax)
        #print("ymin, ymax :", ymin, ymax)
        if scores.max() > 0.78:
           print("inif")
        if(xmin >= left_boundary[0]):
            print("Move_Left")
            cv2.putText(frame,"Move_Left", (300,
100),cv2.FONT_HERSHEY_SIMPLEX,1.5,(0,255,0),2)
        elif(xmax <= right_boundary[0]):
            print("Move Right")
            cv2.putText(frame,"Move Right", (300,
100),cv2.FONT_HERSHEY_SIMPLEX,1.5,(0,255,0),2)
        elif(xmin <= left_boundary[0] and xmax >=
right_boundary[0]):
            print("STOP")
```

```
                cv2.putText(frame,"STOP", (300,
100),cv2.FONT_HERSHEY_SIMPLEX,1.5,(0,255,0),2)

cv2.line(frame,tuple(left_boundary),tuple(left_boundary_top)
, (255, 0, 0), 5)

cv2.line(frame,tuple(right_boundary),tuple(right_boundary_to
p), (255, 0, 0), 5)
        out.write(frame)
except:
  pass
```

2. **Creating websocket server (Python):**

```
import asyncio
import websockets
import json
import time
HOST = "IP address"  # Empty denotes a localhost.
PORT = 7891
CONNECTIONS = set()


async def handler(websocket):
  CONNECTIONS.add(websocket)

  count = 0
  while True:
    try:
      message = await websocket.recv()
      count+=1
      print(message)
      print(count)
      # Send a response to all connected client except the server
      for conn in CONNECTIONS:
        if conn != websocket:

          await conn.send(message)
    except websockets.exceptions.ConnectionClosedError as error1:
```

```
            print(f'Server Error: {error1}')
            CONNECTIONS.remove(websocket)

async def main():
   async with websockets.serve(handler, HOST,
PORT,ping_interval=None, ping_timeout=None):
      await asyncio.Future()
      # run forever



if __name__ == "__main__":
  asyncio.run(main())
```

3. Websocket client interface (Python):

```
import websocket
import time

SOCKET = "ws://192.168.51.29:7891/"

def on_open(ws):
   with open('test2.txt','r') as f: #test2.txt is the output of the openCV
code
      text = f.readlines()
      i = 1
      for line in text:
         if i%3 == 0:
            ws.send(line)
            print(line)
            time.sleep(0.01)
         i+=1

ws = websocket.WebSocketApp(SOCKET, on_open = on_open)
ws.run_forever()
```

4. ESP32 Client interface (Arduino IDLE):

```
#include <Arduino.h>
#include <WiFi.h>
```

```
#include <WebSocketsClient.h>
#include <ArduinoJson.h>
WebSocketsClient webSocket;

int left_actuator = 2;
int right_actuator = 4;
int center_actuator = 18;

const char* ssid = "ssid";
const char* password = "password";

unsigned long messageInterval = 1;
bool connected = false;
#define DEBUG_SERIAL Serial

void webSocketEvent(WStype_t type, uint8_t * payload, size_t length)
{

  switch(type) {
      case WStype_DISCONNECTED:
         DEBUG_SERIAL.printf("[WSc] Disconnected!\n");
         connected = false;
         break;
      case WStype_CONNECTED: {
         //DEBUG_SERIAL.printf("[WSc] Connected to url: %s\n",
payload);
         connected = true;

         // send message to server when Connected
         DEBUG_SERIAL.println("[WSc] SENT: Connected");
      }
         break;
      case WStype_TEXT:
         DEBUG_SERIAL.printf("[WSc] RESPONSE: %s\n", payload);
         String command = (char *) payload;
         if (strcmp(command,""){
           digitalWrite(right_actuator, HIGH);
           delay(50);
           digitalWrite(right_actuator, LOW);
         }
```

```
        else if (strcmp(command,""){
          digitalWrite(left_actuator, HIGH);
          delay(50);
          digitalWrite(left_actuator, LOW);
        }
        else if (strcmp(command,""){
          digitalWrite(center_actuator, HIGH);
          delay(50);
          digitalWrite(center_actuator, LOW);
        }
        break;
    }
}


void setup() {
    DEBUG_SERIAL.begin(9600);

    pinMode(left_actuator, OUTPUT);
    pinMode(right_actuator, OUTPUT);
    pinMode(center_actuator, OUTPUT);

    DEBUG_SERIAL.println();
    DEBUG_SERIAL.println();
    DEBUG_SERIAL.println();
    for(uint8_t t = 4; t > 0; t--) {
        DEBUG_SERIAL.printf("[SETUP] BOOT WAIT %d...\n", t);
        DEBUG_SERIAL.flush();
        delay(1000);
    }
    WiFi.begin(ssid, password);

    while ( WiFi.status() != WL_CONNECTED ) {
      delay ( 500 );
      DEBUG_SERIAL.print ( "." );
    }
    DEBUG_SERIAL.print("Local IP: ");
DEBUG_SERIAL.println(WiFi.localIP());
    // server address, port and URL
    webSocket.begin("IP address", PORT);
```

```
  // event handler
  webSocket.onEvent(webSocketEvent);
}

unsigned long lastUpdate = millis();


void loop() {
  webSocket.loop();
  if (connected && lastUpdate+messageInterval<millis()){
    lastUpdate = millis();
  }
```

Results & Discussion :

Obstacles were created and simulated in the space using a combination of chairs then an input video was taken and processed using open cv module.open cv created a framework which divided the visual field of space into three parts . Now comparing the middle segment with left and right a output of MOVE LEFT ! or MOVE RIGHT! is actuated such that the middle segment does not have any obstacle. In case all three segments encounter the obstacle it gives STOPP! signal.
Now this output is transmitted to server and from there it will be fetched in esp wifi module .The esp wifi module is programmed in such a manner that it activates the corresponding linear resonant actuator as per the input signal

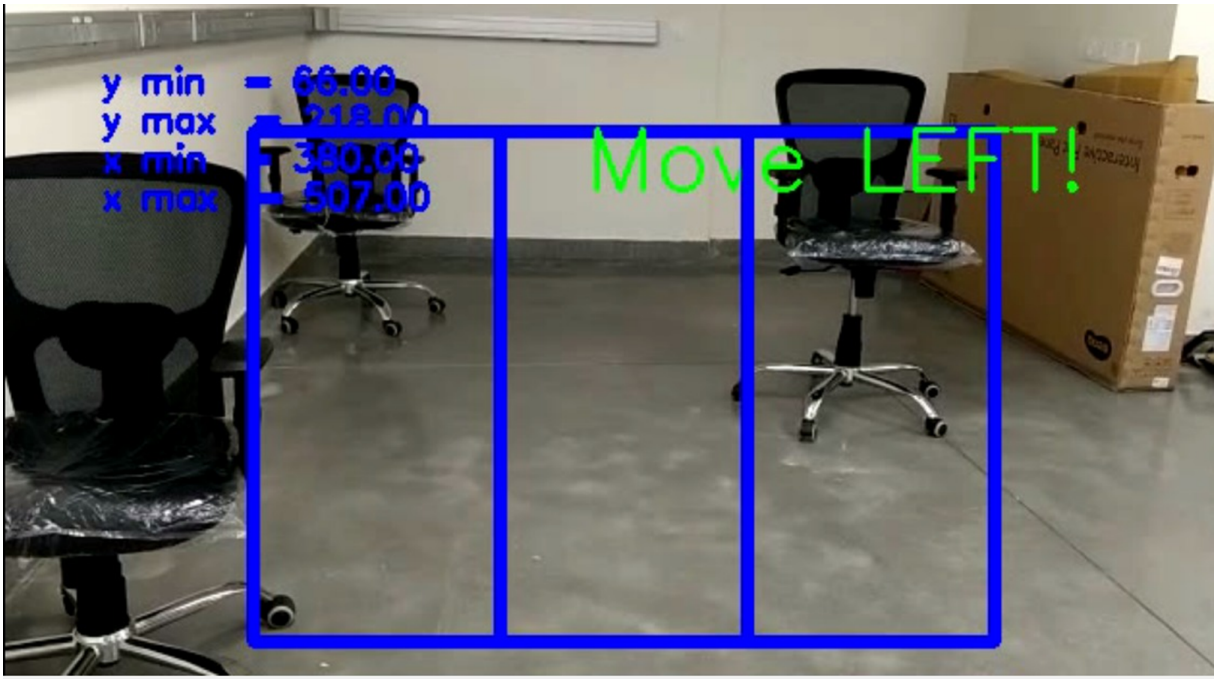

Fig 3 - Framework created by openCV

Equivalent Circuit Diagram :

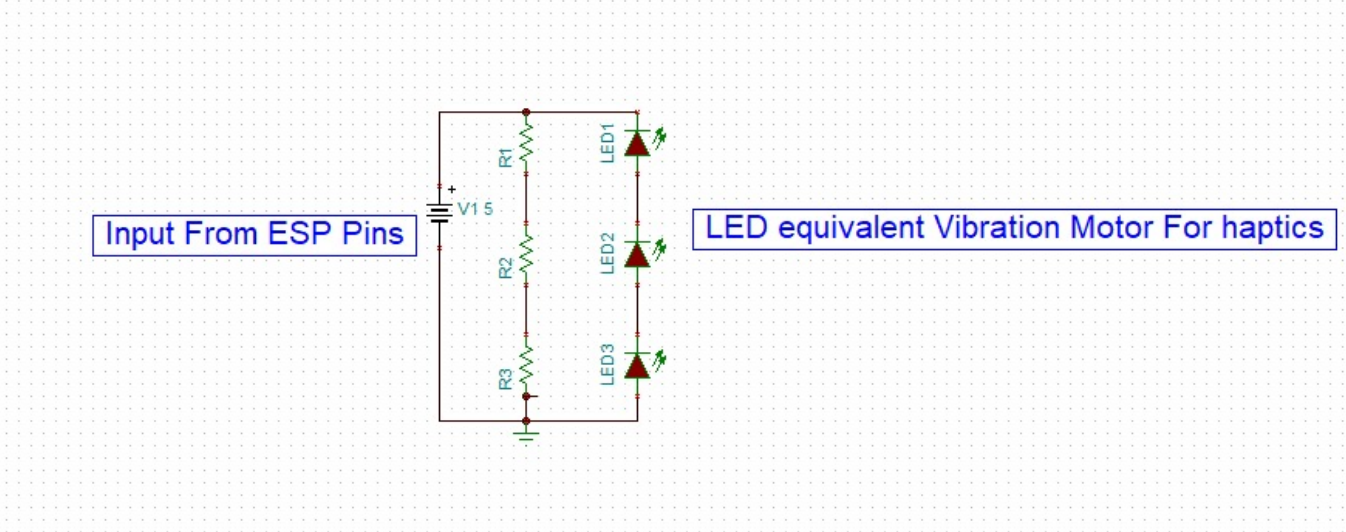

Fig4 -Equivalent Circuit Diagram

A picture of the entire prototype –

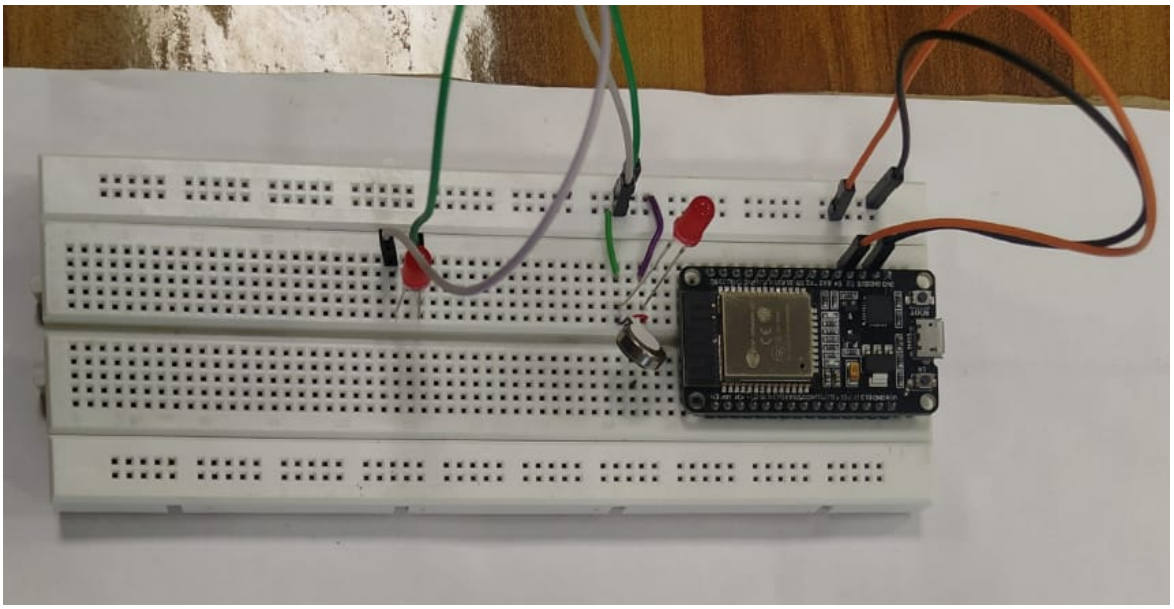

Fig 5 - Entire prototype

A picture of the prototype with the team

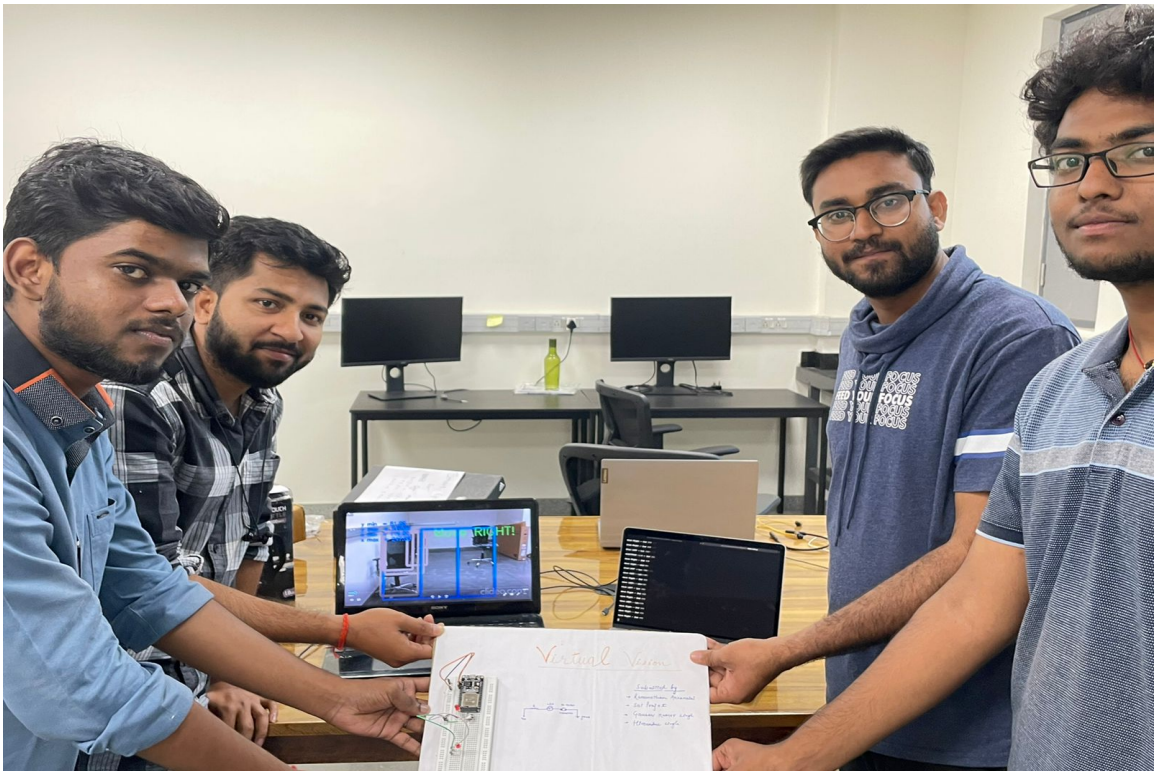

Fig 6-prototype with the team

Future Works:

•The current project is merely a proof of concept and can be extended to a product by the following:

1. We plan to use one of the existing image processing algorithms (Open CV) which give us details on object recognition and depth analysis.
2.
3. • We plan to use Machine Learning models and algorithms to facilitate faster computation of the input and generation of output .
4.
5. •we plan for the division of the field of vision and assigning each actuator to a particular part to make the working of the model more efficient
6. •actuators will be positioned in both the feet of the user as shown in the figure 7.

Fig7. The positions of the actuators

7. •Working of each actuator:
    a. The actuators labeled 1 correspond to the obstructions present in the straight path of the user and will vibrate to indicate the obstruction.
    b. The actuators labeled 2 correspond to the obstructions present in the right and left side of the user will vibrate to indicate the obstruction.
    c. The actuators 3 correspond to the depressions and elevations in the terrain. (This includes ramps, steps and other platforms that can be climbed up or down). The actuators can be tuned to different vibrations to indicate different terrain complications.

•Further aim is to research the various ways we can stimulate the vision regions of the brain and make a subtle environment exclusive for the visually impaired persons.

• Improving the actuator condition in terms of use case, position and output and designing an efficient prototype.

•Using modern technology like spatial audio to enhance the user experience.

•Adding smart mobility features as GPS tracking to the device inorder to make it more user-friendly.

References -

1. Yip W. Determining success of the Orcam MyEye/MyReader in patients with visual impairment. Investigative Ophthalmology & Visual Science. 2017 Jun 23;58(8):3271-.
2. Ortiz LE, Cabrera EV, Gonçalves LM. Depth data error modeling of the ZED 3D vision sensor from stereolabs. ELCVIA: electronic letters on computer vision and image analysis. 2018;17(1):0001-15.
3. Rao SS. Sixth sense technology. In2010 International Conference on Communication and Computational Intelligence (INCOCCI) 2010 Dec 27 (pp. 336-339). IEEE.
4. M, Jovović I, Forenbacher I. The development of information and communication services and devices for the visually impaired. InProceedings in Research Conference In Technical Disciplines RCITD 2013.
5. Stilwell JM, Cermak SA. Perceptual functions of the hand. Hand function in the child. 1995:55-92.