

# NAVIGATOR FOR VISUALLY IMPAIRED PERSON

## **Guide:**

Prof. S.S. Patil

## **Students:**

Nikhil Kanitkar (23)

Dewoo Kudtarkar (27)

Mandar Naik (40)

Pranit Patil (48)

# TABLE OF CONTENT

- 1 INTRODUCTION
- 2 LITERATURE SURVEY
- 3 BLOCK DIAGRAM
- 4 METHODOLOGY
- 5 PLANNING
- 6 PROCESS FLOW
- 7 CODE FOR OBJECT DETECTION AND IDENTIFICATION
- 8 OUTPUT
- 9 PARTIAL EXPLANATION
- 10 COSTING

# INTRODUCTION

Globally, At least **2.2 billion people** have a near or distance vision impairment.

In at least 1 billion – or nearly half – of these cases, vision impairment could have been prevented or has yet to be addressed.

In another way, Creating a fusion of sensing technology and voice-based guidance system, products can be developed which could give better results than individual technology.



Figure: Blind Person

# LITERATURE SURVEY

IEEE ID	Name	Proposed Work	Drawbacks
ISBN:978-1-5386-2456-2	Smart Cap Wearable Visual Guidance System For Blind.	Not able to identify objects near to that person.	Capture image of that object with a specific distance.
ISBN:978-1-7281-1322-7	Smart Assistive Navigation Devices for Visually Impaired People.	This device is based on internet connectivity hence it is not reliable.	Need to make device offline so that it is suitable for everyone.
ISBN:978-1-5386-9471-8	Smart Eye for Visually Impaired-An aid to help the blind people.	Difficult to identify objects at ground level.	Need to pair one more device for ground level object detection.

Table: Literature Survey

# BLOCK DIAGRAM

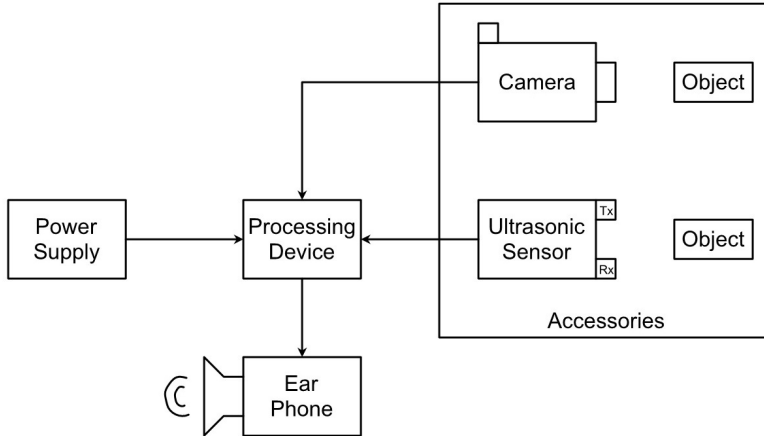


Figure: Block Diagram

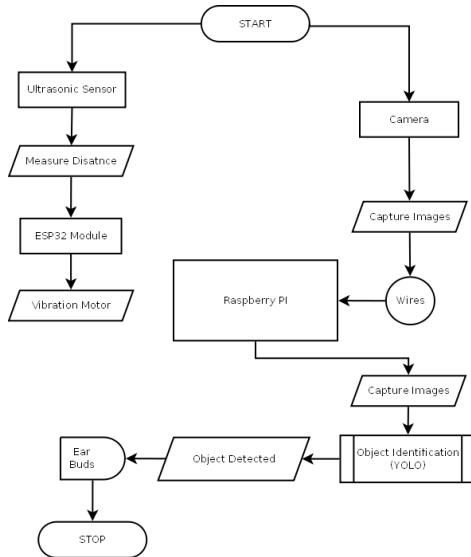
# METHODOLOGY

- The block diagram consists of camera unit, sensor unit, processing unit, power and output unit.
- The camera unit is responsible for capturing objects while the sensor unit provides the distance of object from unit.
- The processing unit plays an important role in detecting and identifying objects (image processing) ,it also receives data from ultrasonic sensor then instruct the user about object identified and distance it is located at (So the user can navigate accordingly).
- The output is provided to user in terms of audio signal using ear phones.

# PLANNING

- The aim is to create a Good user interface to a blind person.
- First, we identify actual problems they faced in their daily life.
- Study and Research on Multiple papers and projects.
- Solve that problem by using two accessories.
  - Smart Glasses - for capturing and identifying images in front of them and output goes to earphones.
  - Smart Shoes - For improvement action of Walk and its output goes to Vibrating Sensor.
- We collect multiple resources to make it affordable and sustainable.
- Now, Working on its Simulation part.
- Once Done, We will move to make its actual Model.

# PROCESS FLOW





# CODE FOR OBJECT DETECTION AND IDENTIFICATION

```
import cv2
import numpy as np
import argparse
from pathlib import Path

# parse arguments
parser = argparse.ArgumentParser()
parser.add_argument("-i", "--img", required=True, help="Path to input image")
parser.add_argument("-c", "--config", required=True, help="Path to config file")
parser.add_argument("-w", "--weights", required=True, help="Path to weights file")
parser.add_argument("-cl", "--classes", required=True, help="Path to text file containing classes")
args = parser.parse_args()

# load model
model = torch.load(args.weights)
model.eval()

# load classes
classes = load_classes(args.classes)

# load image
img = cv2.imread(args.img)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = img.astype(np.float32)
img = img / 255.0

# preprocess
img = preprocess(img, model)

# predict
out = model(img)

# post-process
out = post_process(out, model, classes)

# draw
cv2.imshow("Image", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Figure: Code Part 1

```
def load_classes(classes_path):
    with open(classes_path, 'r') as f:
        lines = f.readlines()
    classes = [line.strip() for line in lines]
    num_classes = len(classes)
    return classes

def load_weights(weights_path):
    with open(weights_path, 'rb') as f:
        header = f.read(4)
        if header != '\x00\x00\x00\x00':
            raise ValueError("Invalid header for weights file")
        f.seek(4)
        weights = f.read()
    return weights

def preprocess(img, model):
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = img.astype(np.float32)
    img = img / 255.0
    img = img.transpose((2, 1, 0))
    img = img[None, :, :, :]
    return img
```

Figure: Code Part 2

```
def post_process(out, model, classes):
    out = out.cpu().numpy()
    out = out.transpose(0, 2, 3, 1)
    out = out * 255.0
    out = out.astype(np.uint8)
    out = out[0]
    out = cv2.cvtColor(out, cv2.COLOR_RGB2BGR)
    out = out.astype(np.uint8)
    return out

def draw(img, out, classes):
    img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
    img = img.astype(np.uint8)
    img = img[0]
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = img.astype(np.float32)
    img = img / 255.0
    img = img.transpose((2, 1, 0))
    img = img[None, :, :, :]
    return img
```

Figure: Code Part 3

```
def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("-i", "--img", required=True, help="Path to input image")
    parser.add_argument("-c", "--config", required=True, help="Path to config file")
    parser.add_argument("-w", "--weights", required=True, help="Path to weights file")
    parser.add_argument("-cl", "--classes", required=True, help="Path to text file containing classes")
    args = parser.parse_args()

    # load model
    model = torch.load(args.weights)
    model.eval()

    # load classes
    classes = load_classes(args.classes)

    # load image
    img = cv2.imread(args.img)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = img.astype(np.float32)
    img = img / 255.0

    # preprocess
    img = preprocess(img, model)

    # predict
    out = model(img)

    # post-process
    out = post_process(out, model, classes)

    # draw
    cv2.imshow("Image", img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Figure: Code Part 4

# OUTPUT



Figure: Before



Figure: After

## PARTIAL EXPLANATION

- The glasses will detect the object which is in front of the person.
- With the help of a camera model, we capture the image and process it in raspberry Pi with the help of the yolo model.
- The shoes will detect the object that is in front of the foot.
- With the help of ultrasonic sensors, we detect the object and process the input in ESP -32 and send the output to the vibrating motor.

# COSTING



Figure: Raspberry PI Zero



Figure: Raspberry PI CAM



Figure: Ultrasonic Sensor

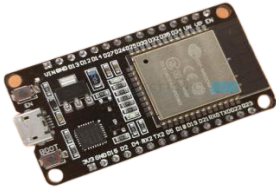


Figure: ESP32 Module



Figure: Vibrating Motor



Figure: Wires

# COSTING



Figure: Battery



Figure: Ear Buds

# COSTING

SR No.	Component	Nos	Cost
1	Raspberry PI Zero	1	5000
2	Raspberry PI CAM	1	400
3	Ultrasonic Sensor	2	60
4	ESP32 Module	2	540
5	Vibration Motor	2	30
6	Power Bank	1	500
7	Ear Buds	1	800
8	Wires	-	50
	Approx.		8000

Table: Costing

THANK YOU