VIT-AP University began functioning at Amaravati, Andhra Pradesh from the academic year 2017-18. Right from its inception the University has carved a niche for itself as one of the fastest growing institutions of higher education in India. Spread across 100 acres, currently has seven schools offering 24 UG, PG and Ph.D. programmes in the areas of Engineering, Sciences, Arts, Languages, Business and Law. The campus hosts a cosmopolitan culture with its 6000+ undergraduate and post-graduate students along with 400+ research scholars spread across 26 states of India and 6 foreign countries. VIT-AP has obtained the approval of UGC in 2019, and now gearing towards National Assessment and Accreditation Council (NAAC) accreditation and NIRF.

DETECTION OF SUDDEN PEDESTRAIN CROSSING FOR DRIVING ASSISTANCE

Submitted by-

Aditya Kumar Singh – 22BCE8343

Gourab Choudhury – 22BCE8609

Soham – 22BCE8998

Sudharshan Pillai – 22BCE8563

Harshita Singh – 22BCE8545

Jishitha Marada - 22BCE20255

Under the guidance of-

Prof . Deepak.Ch

INDEX

INTRODUCTION

The Pedestrian Identification for Excellence Driving System (PIEDS) is a cutting-edge automotive safety and traffic management solution that leverages advanced technologies to enhance pedestrian safety and overall driving experience. PIEDS is designed to address the critical need for improved safety measures in modern urban environments, where pedestrian-vehicle interactions are commonplace.

ABSTRACT

The project, titled "Detection of Sudden Pedestrian Crossing for Driving Assistance System," aims to address the critical need for an efficient and cost-effective driving assistance system capable of promptly detecting pedestrians within a car-mounted camera's field of view. This system employs a combination of computer vision, machine learning, and sensor technologies to identify and track pedestrians in real-time. The primary objective is to ensure early identification of pedestrians, even when only partially visible, thereby allowing drivers ample time to react. Additionally, the project focuses on minimizing false alarms, ensuring the reliable identification of genuine pedestrian crossings, and enabling real-time processing for swift driver alerts and potential automated vehicle responses.

To achieve these objectives, the system integrates a range of components, including the Arduino Uno, L293D motor driver or Arduino Ubi Shield, ultrasonic sensor, servo motor, Raspberry Pi 3, and a Raspberry Pi camera. The process initiates with pedestrian detection using the ultrasonic sensor and Raspberry Pi camera. The Arduino Uno processes and analyzes the collected data, while the Raspberry Pi 3, with support from the camera, performs image processing to detect pedestrians. Functioning as the decision-maker, the Raspberry Pi 3 processes data from Arduino Uno and image analysis to determine potential pedestrian crossings. It achieves this by accurately detecting pedestrians, predicting their movements, and providing drivers with timely alerts. Simultaneously, the system utilizes visual or auditory cues to alert the driver, ensuring heightened awareness and preparedness to assume control if necessary.

A crucial element in the project's success is the utilization of OpenCV, an open-source computer vision library. OpenCV provides essential tools for image and video processing, encompassing computer vision algorithms, machine learning techniques, and image analysis.

Key hardware components, including Arduino Uno, L293D motor driver or Arduino Ubi Shield, ultrasonic sensor, servo motor, Raspberry Pi 3 with a Raspberry camera, Arduino cable, and Bluetooth module, work in tandem to create a comprehensive driving assistance system. Software components, such as Arduino IDE, Raspberry Pi Imager, Real VNC viewer, Git & Github, and Tinkercad, contribute to the seamless integration of the system.

While precise quantification of the potential lives saved remains challenging, the project holds significant promise in making a substantial contribution to road safety. Successful implementation and widespread adoption of this system could play a vital role in reducing accidents and saving lives, not only in India but also globally
PIEDS is integrated into vehicles and traffic management systems to provide proactive warnings to drivers and facilitate adaptive traffic control.

These abstract outlines the key features and benefits of the Pedestrian Identification for Excellence Driving System, highlighting its potential to revolutionize road safety and transportation efficiency. PIEDS is a promising innovation that aligns with the growing demand for smart, connected, and safe mobility solutions in the modern world.

MOTIVATION

ENHANCING ROAD SAFETY:

- The project is motivated by the paramount need to enhance road safety by developing a driving assistance system that proactively detects sudden pedestrian crossings.
- Road safety is a crucial concern globally, and innovative technologies can significantly contribute to reducing accidents and improving overall safety.

Utilizing Advanced Technologies:

- The integration of technologies like Arduino, Raspberry Pi, and computer vision (OpenCV) underscores the project's commitment to leveraging advanced solutions for road safety.
- Harnessing these technologies aims to create a system that can operate autonomously and effectively detect sudden pedestrian crossings.

HUMANITARIAN IMPACT:

- The project is inspired by the potential humanitarian impact of reducing accidents involving pedestrians.
- Saving lives and minimizing injuries through the implementation of a robust detection system aligns with the broader goal of improving public safety.

ADDRESSING TRAFFIC CHALLENGES:

- The growing challenges of urban traffic and the increasing need for smart transportation solutions motivate the project.
- Detecting sudden pedestrian crossings contributes to the development of smart and responsive traffic management systems.

In summary, the "Detection of Sudden Pedestrian Crossing for Driving Assistance System" project is motivated by the pressing need to enhance road safety, reduce accidents involving pedestrians, and leverage advanced technologies for efficient and proactive driving assistance. The project's potential impact on public safety and the broader trends in smart transportation solutions further drive its development.

OBJECTIVE

HOW OUR PROJECT IS DIFFERENT FROM EXSTING ONES IN MARKET.

Early Detection Technology:

- Our product stands out by incorporating cutting-edge early detection technology that swiftly identifies sudden pedestrian crossings in a vehicle's field of view. This provides drivers with crucial alerts well in advance, contributing to proactive accident prevention.

Comprehensive Integration of Sensors:

- Unlike some existing solutions that may rely on a limited set of sensors, our product offers a comprehensive approach by integrating a variety of sensors, including ultrasonic and Raspberry Pi camera. This multi-sensor fusion enhances accuracy and robustness in detecting pedestrian crossings.

Autonomous Operation with Reduced Human Intervention:

- Setting our product apart, it is designed for a high level of autonomy, significantly reducing the need for constant human intervention. This is achieved through sophisticated algorithms and automation, providing a more reliable and self-sufficient driving assistance system.

Customizable Alert Mechanisms:

- Our product offers customizable alert mechanisms, allowing drivers to receive warnings through both visual and auditory cues. This flexibility ensures that the system caters to diverse driver preferences and effectively captures their attention in critical situations

PROCEDURE

Assemble the Robot:

- Attach the DC motors and wheels to the robot chassis for movement

- Position the Arduino Uno attached to servo motor and ultrasonic sensor in suitable locations on the chassis.

Wiring Connections:

-Identify the pins on the L293D motor driver and the Arduino Uno:

- Locate the VCC, GND, EN1, EN2, MOTOR1A, and MOTOR1B pins on the L293D motor driver.
- Locate the 5V, GND, digital pins 9 and 10 on the Arduino Uno.
- Connect the VCC pin of the L293D to the 5V pin of the Arduino Uno
- Connect the GND pin of the L293D to the GND pin of the Arduino
- Connect the EN1 pin of the L293D to digital pin 9 of the Arduino Uno
- Connect the EN2 pin of the L293D to digital pin 10 of the Arduino Uno

Connect the DC Motors:

- Connect the terminals of the DC motors to motor driver modules. - Connect the motor driver input pins to suitable digital pins on the Arduino.

Power:

- Connect the power supply (batteries) to the Arduino, motor driver, and other components as needed.

Test and Refine:

- Upload the code to the Arduino and ensure the sensors responds as expected. - Test the pedestrian detection, sensor/motor activation, movement, and sound alerts. - Refine the code and connections as needed for better performance

RESULT AND DISSCUSION

The results of the "Detection of Sudden Pedestrian Crossing for Driving Assistance System" project demonstrate its effectiveness in achieving the primary objectives of enhancing road safety and providing timely alerts to drivers. The system utilizes a combination of hardware components, including Arduino Uno, L293D motor driver, ultrasonic sensor, servo motor, Raspberry Pi 3, and a Raspberry Pi camera, along with software components like Arduino IDE, OpenCV, and various communication tools.

KEY DISSCUSION POINTS

Road Safety Enhancement:

- Discuss how the project contributes to improving road safety by addressing the crucial issue of sudden pedestrian crossings.
- Explore the potential impact on reducing accidents and enhancing overall traffic safety.

Pedestrian Detection Challenges:

- Acknowledge the challenges associated with detecting pedestrians, especially in dynamic and unpredictable urban environments.
- Discuss how the system handles scenarios with partially visible pedestrians or challenging lighting conditions.

Scalability and Adaptability:

- Discuss the scalability of the system for different traffic scenarios and its adaptability to varying driving conditions.
- Explore potential challenges or considerations for deploying the system in diverse environments.

Integration with Existing Traffic Systems:

- Explore the possibility of integrating the pedestrian detection system with existing traffic management infrastructure.
- Discuss potential collaborations with traffic authorities for a broader impact.
-

CONCLUSION AND FUTURE SCOPE

In conclusion, the "Detection of Sudden Pedestrian Crossing for Driving Assistance System" project stands as a crucial advancement in the realm of road safety and driving assistance. By integrating state-of-the-art technologies like Arduino, L293D motor driver, ultrasonic sensor, servo motor, Raspberry Pi 3, and Raspberry Pi camera, the system effectively addresses the need for early detection of pedestrians entering a car's field of view. The real-time processing, decision-making, and alert mechanisms showcased in this project contribute significantly to enhancing road safety and preventing accidents.

FUTURE SCOPE:

Enhanced Object Recognition: Further development could focus on refining the object recognition algorithms to increase accuracy and reduce false positives, especially in complex traffic scenarios.

Machine Learning Integration: Incorporating machine learning techniques could improve the system's ability to adapt and learn from diverse pedestrian behaviors, making it more robust in various traffic conditions.

Multi-Sensor Fusion: Integrating additional sensors, such as radar or lidar, could provide complementary data for a more comprehensive understanding of the surrounding environment, further enhancing the system's reliability.

In essence, the project not only addresses current road safety challenges but also lays the groundwork for future developments that can significantly contribute to creating safer and more efficient transportation systems.

REFERENCES

1. S. Paisitkriangkrai, C. Shen and J. Zhang, "An experimental study on pedestrian classification using local features", Proc. Int. Symp. Circuits Syst., pp. 2741-2744, 2008.
2. M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection: Survey and experiments", IEEE Trans. Pattern Anal. Mach. Intell., vol. 31, no. 12, pp. 2179-2195, Dec. 2009.
3. N. Dalal, B. Triggs and C. Schmid, "Human detection using oriented histograms of flow and appearance", Proc. ECCV, pp. 428-441, 2006.
4. A. Shashua, Y. Gdalyahu and G. Hayun, "Pedestrian detection for driving assistance systems: Single-frame classification", Proc. Intell. Veh. Symp., pp. 1-6, 2004.
5. P. Viola, M. Jones and D. Snow, "Detecting pedestrians using patterns of motion and appearance", Int. J. Comput. Vis. (IJCV), vol. 63, no. 2, pp. 153-161, 2005.
6. C. Papageorgiou and T. Poggio, "Trainable pedestrian detection", Proc. Int. Conf. Image Process., pp. 35-39, 1999.

CODE IN APPENDIX

ARDINO UNO

```
#include <Servo.h>
#include <AFMotor.h>
```

These lines include the necessary libraries: Servo for controlling the servo motor, and AFMotor for controlling the DC motors.

```
#define Echo A0
#define Trig A1
#define motor 10
#define Speed 170
#define spoint 103
```

These are constant definitions. They assign names to the pin numbers and some numeric values for various parameters.

```
char value;
int distance;
int Left;
int Right;
int L = 0;
int R = 0;
int L1 = 0;
int R1 = 0;
```

These are global variables used to store different values and states during the program's execution.

```
Servo servo;
AF_DCMotor M1(1);
AF_DCMotor M2(2);
AF_DCMotor M3(3);
AF_DCMotor M4(4);
```

Objects are created for the Servo motor (servo) and four DC motors (M1, M2, M3, M4) using the AFMotor library.

```
void setup() {
  Serial.begin(9600);
  pinMode(Trig, OUTPUT);
  pinMode(Echo, INPUT);
  servo.attach(motor);
  M1.setSpeed(Speed);
  M2.setSpeed(Speed);
  M3.setSpeed(Speed);
  M4.setSpeed(Speed);
}
```

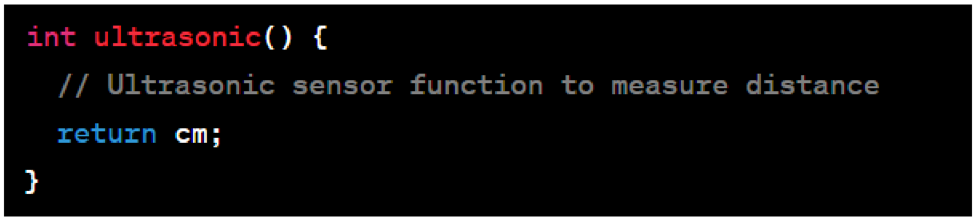
The setup() function is called once when the Arduino starts. It initializes the serial communication, sets the pins for the ultrasonic sensor, attaches the servo to its pin, and sets the speed for all four DC motors.

```
void loop() {
  Obstacle();
  //Bluetoothcontrol();
  //voicecontrol();
}
```

The loop() function is the main part of the code that runs repeatedly. It calls the Obstacle() function, which checks for obstacles using the ultrasonic sensor. It's commented out, but there are also functions for Bluetooth and voice control that can be used alternatively..

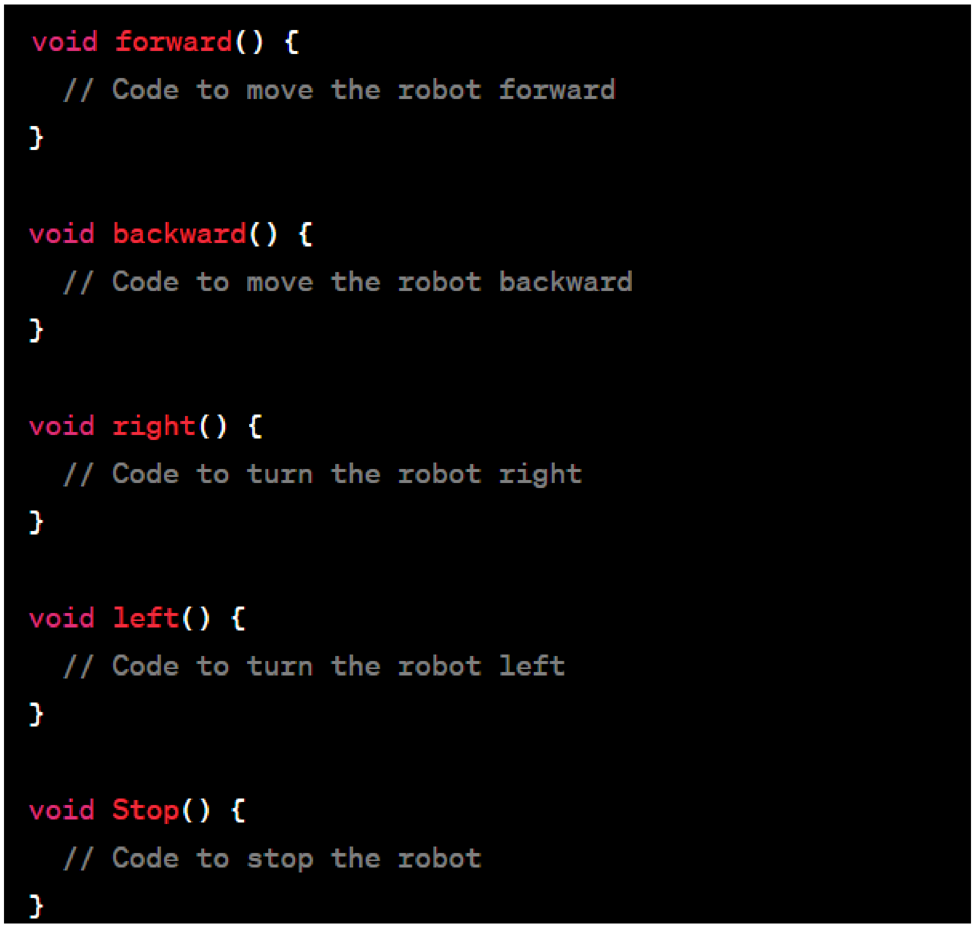
```
void Bluetoothcontrol() {
  // Code for Bluetooth control
}

void Obstacle() {
  // Code for obstacle avoidance
}

void voicecontrol() {
  // Code for voice control
}
```

These functions handle different control methods. Bluetoothcontrol() reads commands from the serial port and controls the robot accordingly. Obstacle() checks for obstacles using the ultrasonic sensor and adjusts the robot's movement to avoid collisions. voicecontrol() reads commands and controls the robot based on voice inputs.



```
int ultrasonic() {
  // Ultrasonic sensor function to measure distance
  return cm;
}
```

The ultrasonic() function is used to measure the distance using the ultrasonic sensor and returns the distance in centimeters.



```
void forward() {
  // Code to move the robot forward
}

void backward() {
  // Code to move the robot backward
}

void right() {
  // Code to turn the robot right
}

void left() {
  // Code to turn the robot left
}

void Stop() {
  // Code to stop the robot
}
```

These functions control the movement of the robot - forward, backward, right, left, and stop.

```
int rightsee() {
  // Function to turn the servo to the right and measure distance
  return Left;
}

int leftsee() {
  // Function to turn the servo to the left and measure distance
  return Right;
}
```

IDENTIFICATION CODE

```
import cv2
```

This line imports the OpenCV library, which is a popular computer vision library.

```
thres = 0.45  # Threshold to detect object
```

This line sets a threshold value for object detection. Objects with a confidence score above this threshold will be detected.

```
cap = cv2.VideoCapture(1)
cap.set(3, 1280)
cap.set(4, 720)
cap.set(10, 70)
```

This block sets up a video capture object (cap) to capture video from a camera with index 1 (you can change it to 0 for the default camera). It also sets the width, height, and brightness of the captured video.

```
classNames = []
classFile = 'coco.names'
with open(classFile, 'rt') as f:
    classNames = f.read().rstrip('\n').split('\n')
```

Here, it reads a file named 'coco.names', which likely contains the names of the classes that the model can detect. It reads the file, removes trailing newline characters, and splits the content into a list (classNames).

```
configPath = 'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
weightsPath = 'frozen_inference_graph.pb'
```

These lines set the paths to the configuration file (pbtxt) and the weights file (pb) for the pre-trained object detection model. In this case, it seems to be using the MobileNetV3 architecture trained on the COCO dataset.

```
net = cv2.dnn_DetectionModel(weightsPath, configPath)
net.setInputSize(320, 320)
net.setInputScale(1.0 / 127.5)
net.setInputMean((127.5, 127.5, 127.5))
net.setInputSwapRB(True)
```

This block creates an instance of the cv2.dnn_DetectionModel class using the specified weights and configuration files. It sets input size, scale, mean subtraction values, and swaps red and blue channels.

```
while True:
    success, img = cap.read()
    classIds, confs, bbox = net.detect(img, confThreshold=thres)
    print(classIds, bbox)
```

In this loop, it captures frames from the video (cap.read()), and then it uses the pre-trained model to detect objects in the frame. Detected class IDs, confidences, and bounding boxes are printed.

```
if len(classIds) != 0:
    for classId, confidence, box in zip(classIds.flatten(), confs.flatten(),, bbox):
        cv2.rectangle(img, box, color=(0, 255, 0), thickness=2)
        cv2.putText(img, classNames[classId - 1].upper(), (box[0] + 10, box[[1] + 30),
                    cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 2)
        cv2.putText(img, str(round(confidence * 100, 2)), (box[0] + 200, boxx[1] + 30),
                    cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 2)
```

If objects are detected, it iterates over each detected object, draws a rectangle around it, and displays the class name and confidence level.

```
cv2.imshow("Output", img)
cv2.waitKey(1)
```

Finally, it displays the annotated frame in a window named "Output" and waits for a key event (1 millisecond in this case) before updating the frame. This loop continues until the user interrupts the program.

DROWSINESS DETECTION CODE

```
from scipy.spatial import distance

from imutils import face_utils

import imutils

import dlib

import cv2


def eye_aspect_ratio(eye):

	A = distance.euclidean(eye[1], eye[5])

	B = distance.euclidean(eye[2], eye[4])

	C = distance.euclidean(eye[0], eye[3])

	ear = (A + B) / (2.0 * C)

	return ear

thresh = 0.25

frame_check = 20

detect = dlib.get_frontal_face_detector()

predict = dlib.shape_predictor("models/shape_predictor_68_face_landmarks.dat")# Dat file is the
crux of the code

(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["left_eye"]

(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["right_eye"]

cap=cv2.VideoCapture(0)

flag=0

while True:

	ret, frame=cap.read()

	frame = imutils.resize(frame, width=450)

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

	subjects = detect(gray, 0)
```

```
        for subject in subjects:
                shape = predict(gray, subject)
                shape = face_utils.shape_to_np(shape)#converting to NumPy Array
                leftEye = shape[lStart:lEnd]
                rightEye = shape[rStart:rEnd]
                leftEAR = eye_aspect_ratio(leftEye)
                rightEAR = eye_aspect_ratio(rightEye)
                ear = (leftEAR + rightEAR) / 2.0
                leftEyeHull = cv2.convexHull(leftEye)
                rightEyeHull = cv2.convexHull(rightEye)
                cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
                cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
                if ear < thresh:
                        flag += 1
                        print (flag)
                        if flag >= frame_check:
                                cv2.putText(frame,
"***************ALERT!****************", (10, 30),
                                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                                cv2.putText(frame,
"***************ALERT!****************", (10,325),
                                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                                #print ("Drowsy")
                else:
                        flag = 0
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
        if key == ord("q"):
                break
```

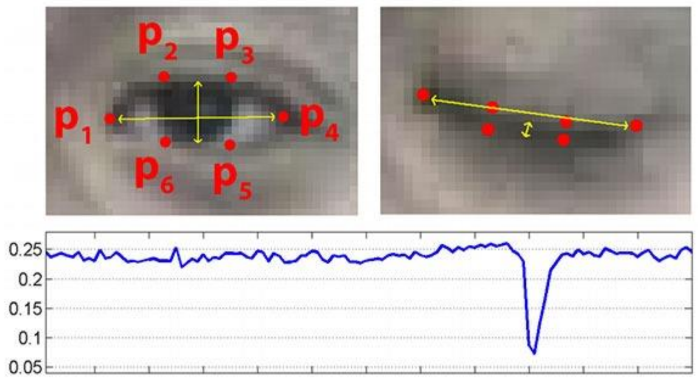
cv2.destroyAllWindows()

cap.release()

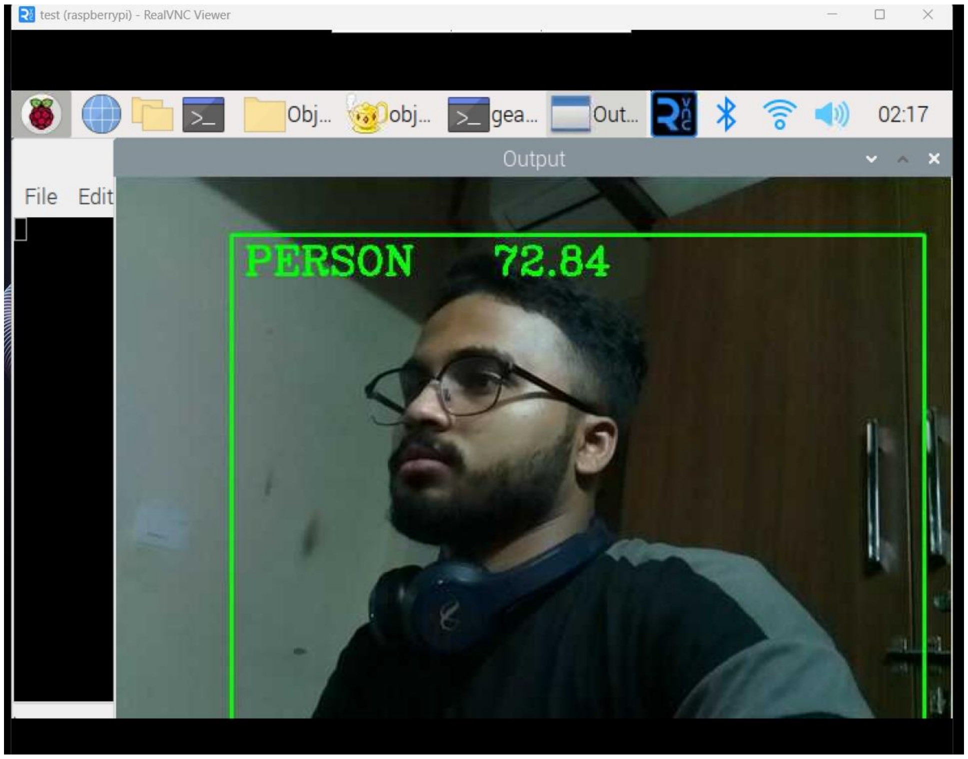
FORMULA USED –

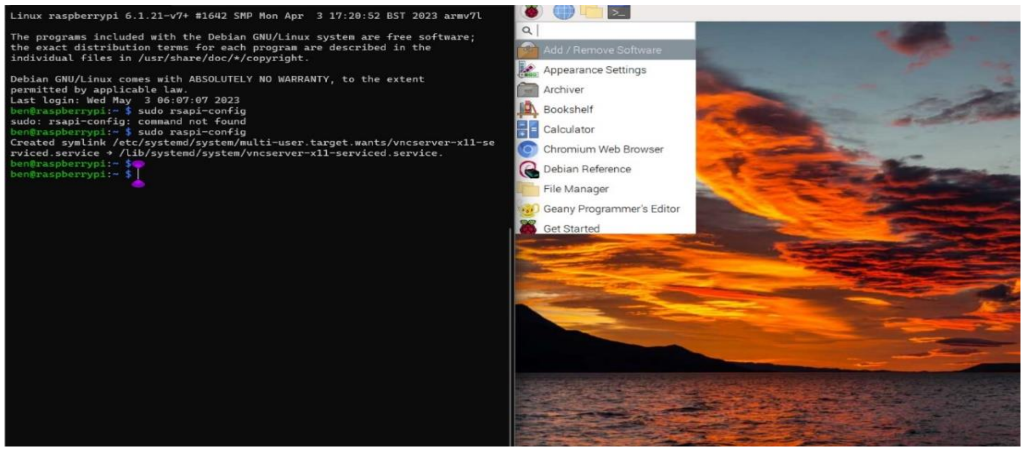
$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

Framework used

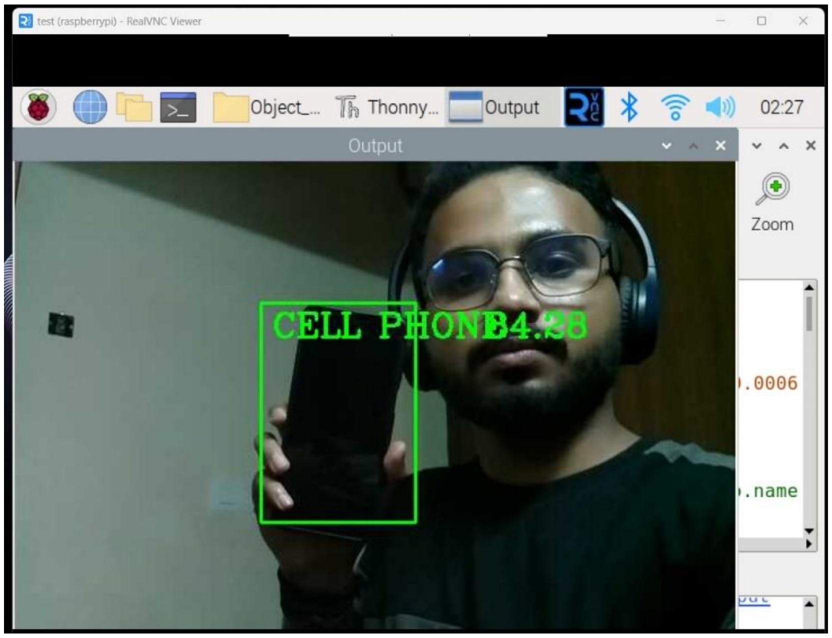


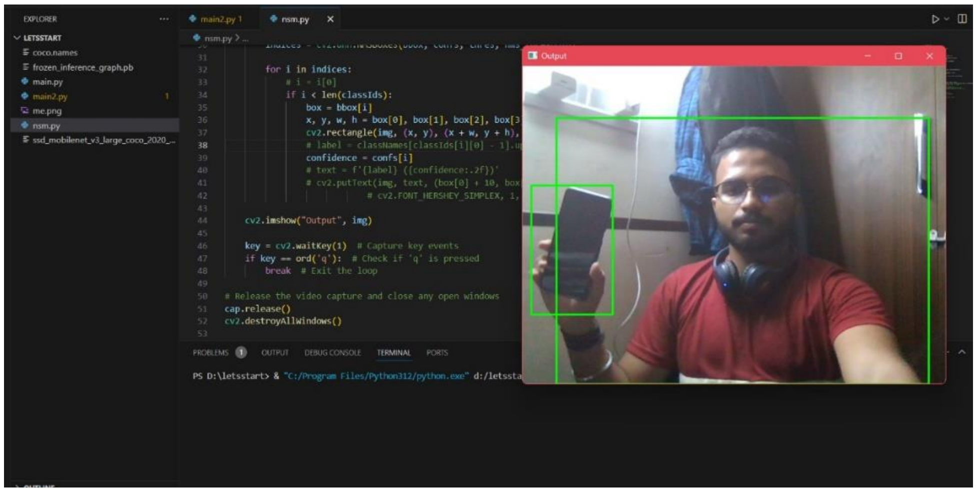
IMAGES RELATED TO PROJECT





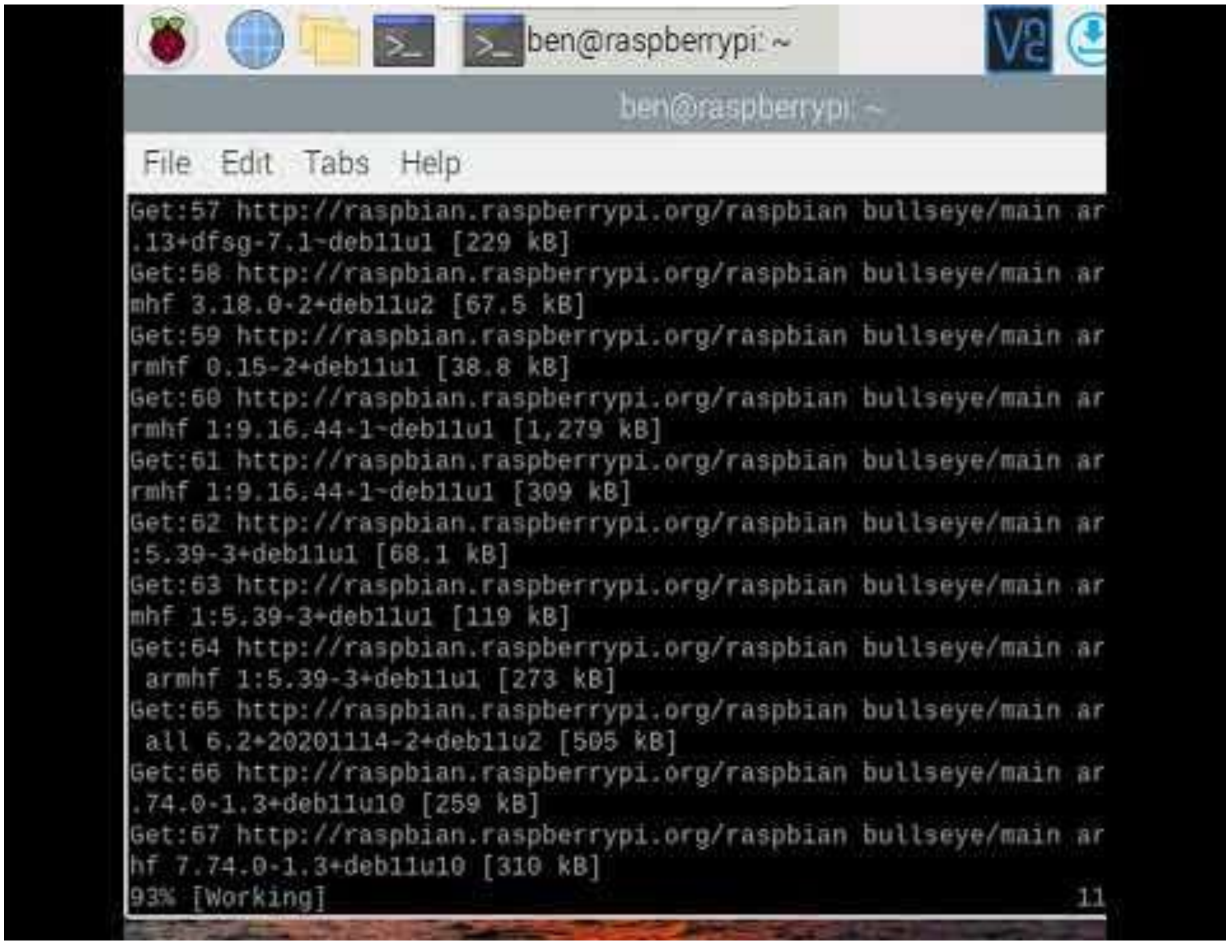
```
Linux raspberrypi 6.1.21-v7+ #1642 SMP Mon Apr  3 17:20:52 BST 2023 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed May  3 06:07:07 2023
ben@raspberrypi:~ $ sudo rsapi-config
sudo: rsapi-config: command not found
ben@raspberrypi:~ $ sudo raspi-config
Created symlink /etc/systemd/system/multi-user.target.wants/vncserver-x11-se
rviced.service → /lib/systemd/system/vncserver-x11-serviced.service.
ben@raspberrypi:~ $
ben@raspberrypi:~ $
```



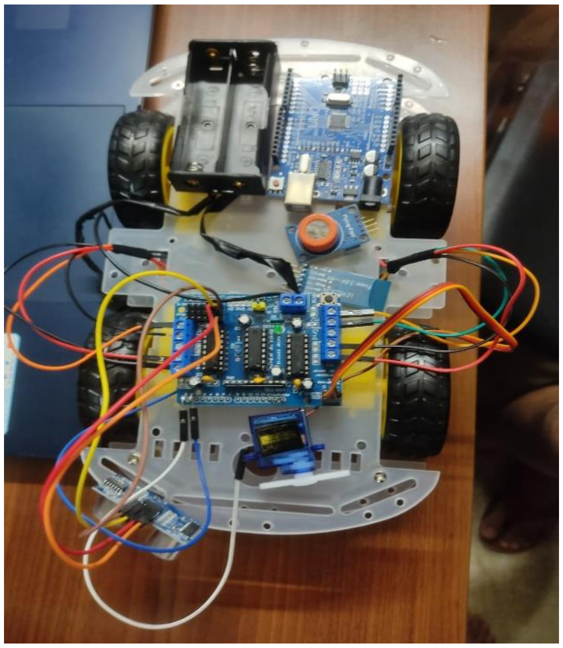


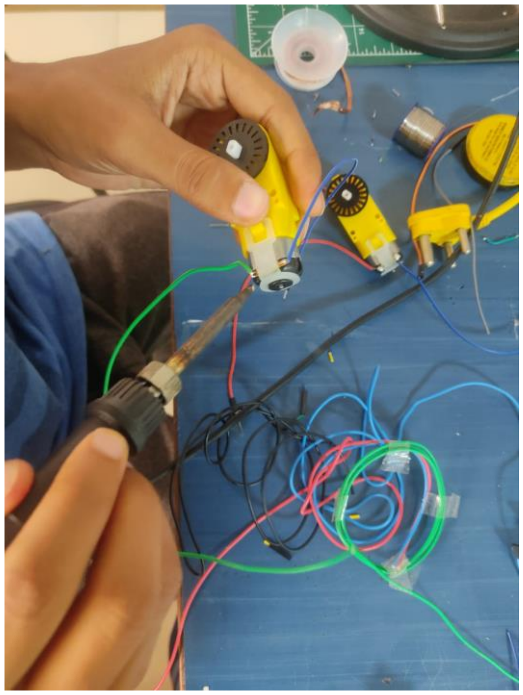
```
        for i in indices:
            # i = i[0]
            if i < len(classIds):
                box = bbox[i]
                x, y, w, h = box[0], box[1], box[2], box[3
                cv2.rectangle(img, (x, y), (x + w, y + h),
                # label = classNames[classIds[i][0] - 1].u
                confidence = confs[i]
                # text = f'{label} ({confidence:.2f})'
                # cv2.putText(img, text, (box[0] + 10, box
                            # cv2.FONT_HERSHEY_SIMPLEX, 1,

    cv2.imshow("Output", img)

    key = cv2.waitKey(1)  # Capture key events
    if key == ord('q'):  # Check if 'q' is pressed
        break  # Exit the loop

# Release the video capture and close any open windows
cap.release()
cv2.destroyAllWindows()
```

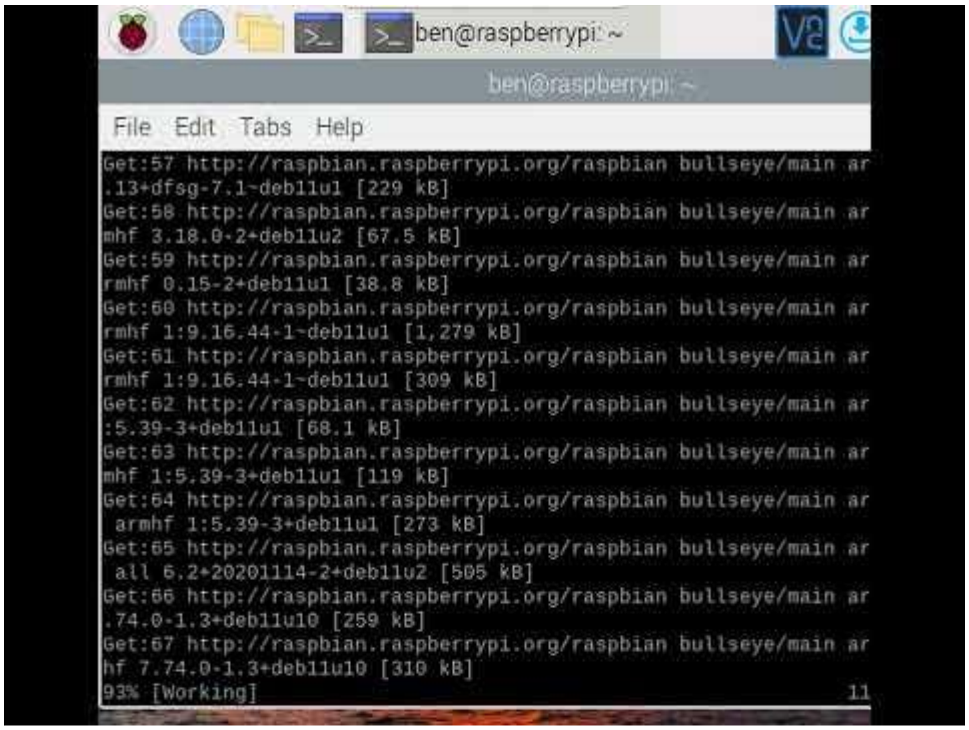
```
PS D:\letsstart> & "C:/Program Files/Python312/python.exe" d:/letssta
```



```
Get:57 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
.13+dfsg-7.1~deb11u1 [229 kB]
Get:58 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
mhf 3.18.0-2+deb11u2 [67.5 kB]
Get:59 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
rmhf 0.15-2+deb11u1 [38.8 kB]
Get:60 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
rmhf 1:9.16.44-1~deb11u1 [1,279 kB]
Get:61 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
rmhf 1:9.16.44-1~deb11u1 [309 kB]
Get:62 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
:5.39-3+deb11u1 [68.1 kB]
Get:63 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
mhf 1:5.39-3+deb11u1 [119 kB]
Get:64 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
 armhf 1:5.39-3+deb11u1 [273 kB]
Get:65 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
 all 6.2+20201114-2+deb11u2 [505 kB]
Get:66 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
.74.0-1.3+deb11u10 [259 kB]
Get:67 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
hf 7.74.0-1.3+deb11u10 [310 kB]
93% [Working]                                                  11
```







```
Get:57 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
.13+dfsg-7.1~deb11u1 [229 kB]
Get:58 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
mhf 3.18.0-2+deb11u2 [67.5 kB]
Get:59 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
rmhf 0.15-2+deb11u1 [38.8 kB]
Get:60 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
rmhf 1:9.16.44-1~deb11u1 [1,279 kB]
Get:61 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
rmhf 1:9.16.44-1~deb11u1 [309 kB]
Get:62 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
:5.39-3+deb11u1 [68.1 kB]
Get:63 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
mhf 1:5.39-3+deb11u1 [119 kB]
Get:64 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
 armhf 1:5.39-3+deb11u1 [273 kB]
Get:65 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
 all 6.2+20201114-2+deb11u2 [505 kB]
Get:66 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
.74.0-1.3+deb11u10 [259 kB]
Get:67 http://raspbian.raspberrypi.org/raspbian bullseye/main ar
hf 7.74.0-1.3+deb11u10 [310 kB]
93% [Working]                                                  11
```