```
    classes = [line.strip() for line in f.readlines()]

layer_names = net.getLayerNames()
output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]

# Initialize USB camera
cap = cv2.VideoCapture(0)

# Adjust the index if your camera is not the default
def get_distance():
    # Trigger the ultrasonic sensor
    GPIO.output(TRIG_PIN, GPIO.HIGH)
    time.sleep(0.00001)
    GPIO.output(TRIG_PIN, GPIO.LOW)

 # Measure the time for the ECHO pin to go high
    while GPIO.input(ECHO_PIN) == 0:
        pulse_start = time.time()

    while GPIO.input(ECHO_PIN) == 1:
        pulse_end = time.time()

  # Calculate distance from the time difference
    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17150

  # Speed of sound is 343 meters per second at sea level
    distance = round(distance, 2)

 # Round to two decimal places
    return distance
while True:
```

```
# Capture frame-by-frame
ret, frame = cap.read()
if not ret:
    break
height, width, channels = frame.shape

# Detecting objects
blob = cv2.dnn.blobFromImage(frame, 0.00392, (220, 220), (0, 0, 0), True, crop=False)
net.setInput(blob)
outs = net.forward(output_layers)

# Showing information on the screen
for out in outs:
    for detection in out:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.5:

            # Object detected
            center_x = int(detection[0] * width)
            center_y = int(detection[1] * height)
            w = int(detection[2] * width)
            h = int(detection[3] * height)

            # Rectangle coordinates
            x = int(center_x - w / 2)
            y = int(center_y - h / 2)
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            cv2.putText(frame, classes[class_id], (x, y -
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
            print(classes[class_id])
            lcd.clear() # we need to clear so it sets cursor back
            lcd.lcd_string(classes[class_id],lcd.LCD_LINE_1)
```

```
                m="espeak the_object_is"+str(classes[class_id])
                import os
                os.system(m)

    # Display the resulting frame
    cv2.imshow('Object Detection', frame)
    distance = get_distance()
    print(f"Distance: {distance} cm")
    if distance < 10:
        print("hi")
        p.ChangeDutyCycle(10)
        time.sleep(0.5)
        p.ChangeDutyCycle(2.5)
        time.sleep(0.5)

    # Press 'q' to quit the application
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the capture
cap.release()
cv2.destroyAllWindows()
```

7. CODING

```
import cv2
import numpy as np
import RPi.GPIO as GPIO
import time
from LCDI2C_backpack import LCDI2C_backpack
import time
import RPi.GPIO as GPIO
import time

TRIG_PIN = 20
ECHO_PIN = 21
GPIO.setmode(GPIO.BCM)
GPIO.setup(TRIG_PIN, GPIO.OUT)
GPIO.setup(ECHO_PIN, GPIO.IN)

# Define GPIO pins
servoPIN = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(servoPIN, GPIO.OUT)
lcd = LCDI2C_backpack(0x27)
p = GPIO.PWM(servoPIN, 50)
p.start(2.5)
lcd.lcd_string("Blind Walking",lcd.LCD_LINE_1)
lcd.lcd_string("stick",lcd.LCD_LINE_2)
time.sleep(2)
lcd.clear()

# Load YOLO
net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
classes = []
with open("yolov3.txt", "r") as f:
```