```
import numpy as np
import argparse
import time
import cv2
import os
import speech_recognition as sr
from gtts import gTTS
import serial
from imutils.video import VideoStream



from geopy.geocoders import Nominatim

# initialize Nominatim API
geolocator = Nominatim(user_agent="geoapiExercises")


# Latitude & Longitude input




ser=serial.Serial('com6',9600,timeout=0.1)

ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=False,
 help="path to input image")
ap.add_argument("-y", "--yolo", required=False,
 help="base path to YOLO directory")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
 help="minimum probability to filter weak detections")
ap.add_argument("-t", "--threshold", type=float, default=0.3,
 help="threshold when applyong non-maxima suppression")
args = vars(ap.parse_args())


labelsPath = "coco.names"
LABELS = open(labelsPath).read().strip().split("\n")

np.random.seed(42)
COLORS = np.random.randint(0, 255, size=(len(LABELS), 3),
 dtype="uint8")

weightsPath ='yolov3.weights'
configPath = "yolov3.cfg"


print("[INFO] loading YOLO from disk...")
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
vs=VideoStream(src=0).start()
time.sleep(2)
while(1):
        image = vs.read()
```

```
cv2.imshow('in',image)
key = cv2.waitKey(1)
info=str(ser.readline().decode())
if(len(info)>0):
        inf=info.split(',')
        print(inf[1])
        if(inf[1]=='object'):
                (H, W) = image.shape[:2]

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


                blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416),
                        swapRB=True, crop=False)
                net.setInput(blob)
                start = time.time()
                layerOutputs = net.forward(ln)
                end = time.time()


                print("[INFO] YOLO took {:.6f} seconds".format(end - start))


                boxes = []
                confidences = []
                classIDs = []
                ID = 0


                for output in layerOutputs:

                        for detection in output:

                                scores = detection[5:]
                                classID = np.argmax(scores)
                                confidence = scores[classID]


                                if confidence > args["confidence"]:

                                        box = detection[0:4] * np.array([W, H, W, H])
                                        (centerX, centerY, width, height) = box.astype("int")




                                        x = int(centerX - (width / 2))
                                        y = int(centerY - (height / 2))


                                        boxes.append([x, y, int(width), int(height)])
                                        confidences.append(float(confidence))
                                        classIDs.append(classID)
```

```
        idxs = cv2.dnn.NMSBoxes(boxes, confidences, args["confidence"],
                args["threshold"])


        if len(idxs) > 0:
                list1 = []
                for i in idxs.flatten():

                        (x, y) = (boxes[i][0], boxes[i][1])
                        (w, h) = (boxes[i][2], boxes[i][3])
                        centerx = round((2*x + w)/2)
                        centery = round((2*y + h)/2)
                        if centerX <= W/3:
                                W_pos = "left "
                        elif centerX <= (W/3 * 2):
                                W_pos = "center "
                        else:
                                W_pos = "right "

                        if centerY <= H/3:
                                H_pos = "top "
                        elif centerY <= (H/3 * 2):
                                H_pos = "mid "
                        else:
                                H_pos = "bottom "
                        list1.append(H_pos + W_pos + LABELS[classIDs[i]])

                description = ', '.join(list1)
                print(description)

                myobj = gTTS(text=description, lang="en", slow=False)

                myobj.save("object_detection.mp3")
                os.system("object_detection.mp3")
else:
        Latitude=inf[1]
        Longitude=inf[2]
        print('LT:'+str(Latitude))
        print('LG:'+str(Longitude))

        location = geolocator.reverse(Latitude+","+Longitude)

        address = location.raw['address']

        # traverse the data
        city = address.get('city', '')
        state = address.get('state', '')
        country = address.get('country', '')
        code = address.get('country_code')
        zipcode = address.get('postcode')
        print('City : ', city)
        print('State : ', state)
        print('Country : ', country)
        print('Zip Code : ', zipcode)
```

```
add=city +"  " + state+"    "+country
myobj = gTTS(text=add, lang="en", slow=False)

myobj.save("object_detection.mp3")
os.system("object_detection.mp3")
```