import cv2

import pyttsx3

#Libraries

import RPi.GPIO as GPIO

import time

from time import sleep

#GPIO Mode (BOARD / BCM)

GPIO.setmode(GPIO.BCM)

#set GPIO Pins

GPIO\_TRIGGER = 18

GPIO\_ECHO = 24

buzzer=23

GPIO.setup(buzzer,GPIO.OUT)

#set GPIO direction (IN / OUT)

GPIO.setup(GPIO\_TRIGGER, GPIO.OUT)

GPIO.setup(GPIO\_ECHO, GPIO.IN)

def distance():

# set Trigger to HIGH

GPIO.output(GPIO\_TRIGGER, True)

# set Trigger after 0.01ms to LOW

time.sleep(0.00001)

GPIO.output(GPIO\_TRIGGER, False)

StartTime = time.time()

StopTime = time.time()

# save StartTime

while GPIO.input(GPIO\_ECHO) == 0:

StartTime = time.time()

# save time of arrival

while GPIO.input(GPIO\_ECHO) == 1:

StopTime = time.time()

# time difference between start and arrival

TimeElapsed = StopTime - StartTime

# multiply with the sonic speed (34300 cm/s)

# and divide by 2, because there and back

distance = (TimeElapsed \* 34300) / 2

return distance

#thres = 0.45 # Threshold to detect object

# init function to get an engine instance for the speech synthesis

engine = pyttsx3.init()

classNames = []

classFile = "/home/pi/Desktop/Object\_Detection\_Files/coco.names"

with open(classFile,"rt") as f:

classNames = f.read().rstrip("\n").split("\n")

configPath = "/home/pi/Desktop/Object\_Detection\_Files/ssd\_mobilenet\_v3\_large\_coco\_2020\_01\_14.pbtxt"

weightsPath = "/home/pi/Desktop/Object\_Detection\_Files/frozen\_inference\_graph.pb"

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)

def getObjects(img, thres, nms, draw=True, objects=[]):

classIds, confs, bbox = net.detect(img,confThreshold=thres,nmsThreshold=nms)

#print(classIds,bbox)

if len(objects) == 0: objects = classNames

objectInfo =[]

if len(classIds) != 0:

for classId, confidence,box in zip(classIds.flatten(),confs.flatten(),bbox):

className = classNames[classId - 1]

print(className)

engine.say(className)

engine.runAndWait()

dist=int(distance())

print(dist)

if(dist<90):

GPIO.output(buzzer,GPIO.HIGH)

print ("Beep")

sleep(1)

engine.say(className +"is in"+str(dist)+"inches")

engine.runAndWait()

else:

GPIO.output(buzzer,GPIO.LOW)

print ("No Beep")

sleep(0.1)

if className in objects:

objectInfo.append([box,className])

if (draw):

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,box[1]+30),

cv2.FONT\_HERSHEY\_COMPLEX,1,(0,255,0),2)

return img,objectInfo

if \_\_name\_\_ == "\_\_main\_\_":

cap = cv2.VideoCapture(0)

cap.set(3,640)

cap.set(4,480)

#cap.set(10,70)

while True:

success, img = cap.read()

result, objectInfo = getObjects(img,0.45,0.2)

#print(objectInfo)

cv2.imshow("Output",img)

cv2.waitKey(1)