**Code**

import RPi.GPIO as GPIO

import time

import sys

GPIO.setmode(GPIO.BOARD)

TRIG = 16

ECHO = 18

LED = 15 #sanitizer

leddoor = 22 # Door

leddoor2 = 21

GPIO.setup(TRIG,GPIO.OUT)

GPIO.setup(ECHO,GPIO.IN)

GPIO.setup(LED,GPIO.OUT)

GPIO.setup(leddoor, GPIO.OUT)

GPIO.setup(leddoor2, GPIO.OUT)

GPIO.output(leddoor,True)

GPIO.output(leddoor2,False)

GPIO.output(LED,True)

GPIO.output(TRIG, False)

print("Calibrating.....")

time.sleep(2)

def ultra():

i=0

GPIO.output(TRIG, True)

time.sleep(0.00001)

GPIO.output(TRIG, False)

while GPIO.input(ECHO)==0:

pulse\_start = time.time()

while GPIO.input(ECHO)==1:

pulse\_end = time.time()

pulse\_duration = pulse\_end - pulse\_start

distance = pulse\_duration \* 17150

distance = round(distance+1.15, 2)

if distance<=15 and distance>=5:

print ("distance:",distance,"cm")

print ("sanitization ON ")

GPIO.output(LED,False)

time.sleep(0.5)

GPIO.output(LED,True)

i=1

if distance>20 and i==1:

print ("place the object....")

GPIO.output(LED,True)

i=0

def doorled(arg1):

if arg1==1:

GPIO.output(leddoor,True)

GPIO.output(leddoor2,False)

print("DOOR open")

else:

GPIO.output(leddoor,False)

GPIO.output(leddoor2,True)

print("DOOR close")

#pi camera changes

from picamera.array import PiRGBArray

from picamera import PiCamera

# Import OpenCV2 for image processing

import cv2

# initialize the camera and grab a reference to the raw camera capture

camera = PiCamera()

camera.resolution = (640, 480)

camera.framerate = 32

rawCapture = PiRGBArray(camera, size=(640, 480))

# allow the camera to warmup

time.sleep(0.1)

# Import numpy for matrices calculations

import numpy as np

# Create Local Binary Patterns Histograms for face recognization

recognizer = cv2.face.LBPHFaceRecognizer\_create()

# Load the trained mode

recognizer.read('trainer/trainer.yml')

# Load prebuilt model for Frontal Face

cascadePath = "haarcascade\_frontalface\_default.xml"

# Create classifier from prebuilt model

faceCascade = cv2.CascadeClassifier(cascadePath);

# Set the font style

font = cv2.FONT\_HERSHEY\_SIMPLEX

# Initialize and start the video frame capture

#cam = cv2.VideoCapture(0)

# Loop

try:

while True:

# Read the video frame

#ret, im =cam.read()

for frame in camera.capture\_continuous(rawCapture, format="bgr", use\_video\_port=True):

doorled(0)

ultra()

# grab the raw NumPy array representing the image, then initialize the timestamp

# and occupied/unoccupied text

im = frame.array

# Convert the captured frame into grayscale

gray = cv2.cvtColor(im,cv2.COLOR\_BGR2GRAY)

# Get all face from the video frame

faces = faceCascade.detectMultiScale(gray, 1.2,5)

# For each face in faces

for(x,y,w,h) in faces:

# Create rectangle around the face

cv2.rectangle(im, (x-20,y-20), (x+w+20,y+h+20), (0,255,0), 4)

# Recognize the face belongs to which ID

Id,conf = recognizer.predict(gray[y:y+h,x:x+w])

# Check the ID if exist

if(Id == 2 and conf <100):

Id = "masked"

doorled(1)

#If not exist, then it is Unknown

else:

Id = "Unmasked"

doorled(0)

print(Id,",",conf)

# Put text describe who is in the picture

cv2.rectangle(im, (x-22,y-90), (x+w+22, y-22), (0,255,0), -1)

cv2.putText(im, str(Id), (x,y-40), font, 2, (255,255,255), 3)

ultra()

# Display the video frame with the bounded rectangle

cv2.imshow('im',im)

rawCapture.truncate(0)

# If 'q' is pressed, close program

if cv2.waitKey(1) & 0xFF == ord('q'):

break

except KeyboardInterrupt:

print("interrupted by user!")

# Stop the camera

#cam.release()

# Close all windows

cv2.destroyAllWindows()

GPIO.output(leddoor,False)

GPIO.output(LED,True)

GPIO.output(leddoor2,False)

GPIO.cleanup()