**SECURE HOME AUTOMATION USING RASPBERRY PI BY TELEGRAM APP**

CODE IMPLEMENTATION:

import sys

import urllib.request

import cv2

import numpy as np

import os

from datetime import datetime

import time

import RPi.GPIO as GPIO

from gpiozero import Servo

servo = Servo(2)

time.sleep(0.2)

servo.value=1

rled=26

gled=19

buz=3

GPIO.setup(rled,GPIO.OUT)

GPIO.setup(gled,GPIO.OUT)

GPIO.setup(buz,GPIO.OUT)

GPIO.output(rled,0)

GPIO.output(gled,0)

GPIO.output(buz,0)

def distance(v1, v2):

# Eucledian

return np.sqrt(((v1 - v2) \*\* 2).sum())

def knn(train, test, k=5):

dist = []

for i in range(train.shape[0]):

# Get the vector and label

ix = train[i, :-1]

iy = train[i, -1]

# Compute the distance from test point

d = distance(test, ix)

dist.append([d, iy])

#print(dist)

# Sort based on distance and get top k

dk = sorted(dist, key=lambda x: x[0])[:k]

# Retrieve only the labels

labels = np.array(dk)[:, -1]

# Get frequencies of each label

output = np.unique(labels, return\_counts=True)

# Find max frequency and corresponding label

index = np.argmax(output[1])

return output[0][index]

cap = cv2.VideoCapture(0)

face\_cascade = cv2.CascadeClassifier("haarcascade\_frontalface\_alt.xml")

skip = 0

dataset\_path = r"/home/pi/Desktop/code/data/"

face\_data = []

number = []

labels = []

class\_id = 0 # Labels for the given file

names = {} # Mapping btw id - name

# Data Preparation

for fx in os.listdir(dataset\_path):

if fx.endswith('.npy'):

# Create a mapping btw class\_id and name

names[class\_id] = fx[:-4]

#print("Loaded " + fx)

data\_item = np.load(dataset\_path + fx)

face\_data.append(data\_item)

# Create Labels for the class

target = class\_id \* np.ones((data\_item.shape[0],))

class\_id += 1

labels.append(target)

face\_dataset = np.concatenate(face\_data, axis=0)

face\_labels = np.concatenate(labels, axis=0).reshape((-1, 1))

print(face\_dataset.shape)

print(face\_labels.shape)

trainset = np.concatenate((face\_dataset, face\_labels), axis=1)

print(trainset.shape)

# Testing

attn = []

appn = []

time.sleep(1)

ij=0

prv=0

def cntrl():

global prv

r\_link='https://api.thingspeak.com/channels/661489/fields/1/last?api\_key=GKA3RET23Y4YFG5D'

f=urllib.request.urlopen(r\_link)

rcv = (f.readline()).decode()

if(prv != rcv):

prv=rcv

if(rcv[0]=='1'):

servo.value=-1

if(rcv[0]=='2'):

servo.value=1

while(1):

prvname=""

fd=0

sc=0

while fd==0:

cntrl()

ret, frame = cap.read()

if ret == False:

continue

faces = face\_cascade.detectMultiScale(frame, 1.3, 5)

if (len(faces) > 0):

for face in faces:

x, y, w, h = face

# Get the face ROI

offset = 10

face\_section = frame[y - offset:y + h + offset, x - offset:x + w + offset]

face\_section = cv2.resize(face\_section, (100, 100))

# Predicted Label (out)

out = knn(trainset, face\_section.flatten())

# Display on the screen the name and rectangle around it

pred\_name = names[int(out)]

if(prvname==pred\_name):

sc=sc+1

else:

sc=0

prvname=pred\_name

cv2.putText(frame, pred\_name, (x, y - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 0, 0), 2, cv2.LINE\_AA)

cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 255), 2)

#if pred\_name not in attn:

# attn.append(pred\_name)

#else:

# continue

if(sc>1):

sc=0

fd=1

cv2.imshow("Faces", frame)

key = cv2.waitKey(1) & 0xFF

if key == ord('q'):

break

cv2.imshow("Faces", frame)

key = cv2.waitKey(1) & 0xFF

if key == ord('q'):

break

print ("NAME:" + pred\_name)

if(pred\_name=='satya'):

GPIO.output(rled,0)

GPIO.output(gled,1)

print("Authorized..")

servo.value=-1

time.sleep(5)

servo.value=1

time.sleep(2)

GPIO.output(rled,0)

GPIO.output(gled,0)

else:

GPIO.output(buz,1)

print('UnAuthorized')

GPIO.output(buz,0)