**AUTOMATED ACCESS CONTROL USING FACIAL DETECTION, BLUETOOTH AND GESTURE**

**A**

**PROJECT BY**

**KAUSTAV BORA**

**2ND YEAR**

COMPUTER SCIENCE ENGINEERING

SPECIALIZATION IN BIG DATA ANALYTICS

**ABOUT THE PROJECT:**

This project is “Automated access control using Facial Recognition, Bluetooth and Gesture “. This project is primely developed to make the whole locking/unlocking system autonomous and less cumbersome.

**MATERIALS USED:**

* Arduino Uno
* Bluetooth Module HC-05
* MPU 6050
* OpenCV
* Python
* Windows Platform
* LEDs, Resistors
* Camera
* Android device

**HOW THE PROJECT WORKS?**

Firstly, we start the project by connecting the Arduino Board to the power supply and the external power supply to the devices. Then we upload the Arduino code to the northern we run the python script for the facial recognition part. But before running the script we must generate facial datasets for the algorithm to recognize. We create the data set by running the createdata.py script. But before running the algorithm we must create a folder inside the dataset folder. After running the Createdata.py script the facial data will automatically get stored. Once the data has been stored the face\_recognize.py script can recognize the face. After running the script, we can access the lock by using gestures, this is done by using the MPU6050 with sends gestures data to the Arduino board. We can also lock unlock by using an android app that I created. We need to send the password through the app by the Bluetooth module to the Arduino board.

**HOW TO RUN THE PROJECT:**

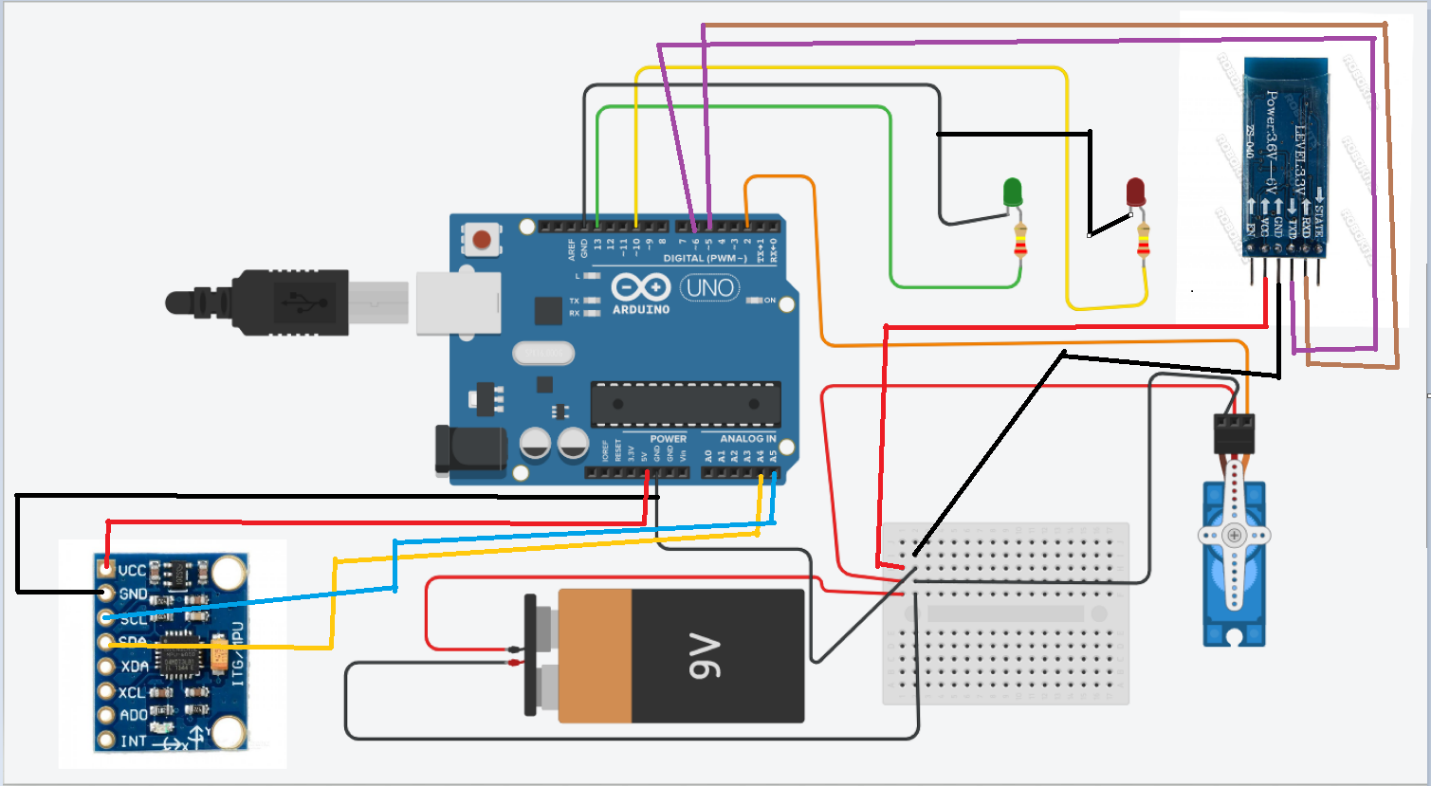
* First make up a directory by any name and make a folder dataset. Then inside the dataset folder create another folder with the folder name of the person whose dataset you want to create.
* Then run the create\_data.py file by modifying the code containing the folder name of the person you want to store the data.
* Run the python script.
* After running the script, the algorithm will recognize human face with the webcam feed and capture the facial data.
* Then copy the Arduino code and upload in the Arduino Uno board.
* Make the connections as per the circuit diagram given below.
* Then provide the power supply as given in the circuit diagram
* Then run the python script “face\_recognize.py”
* The project will start working
* For the Bluetooth part install the apk as provided by me in the GitHub.

(GitHub link is below)

* Then for the access through the Bluetooth app use the code “admin” as the passkey
* That all enjoy the project demonstration.

(For project demo the video link is provided below and its also in my GitHub repository)

**CIRCUIT DIAGRAM:**

****

**SOURCE CODE: (ARDUINO)**

#include <Wire.h> // importing the required libraries

#include<SoftwareSerial.h>

#include <MPU6050.h> // library for the gyroscope and accelerometer

#include<String.h> // for the password from bluetooth

#include <Servo.h> // servo lib for rotating the servo

char final[5]; // declaring for the password

char correct[5] = {'a','d','m','i','n'}; //the correct sequence of password from the bluetooth device

Servo sg90;

int pass\_true; // creating a servo object

char state[10]; // declaring state for matching the password

int servo\_pin = 2;

MPU6050 sensor ;

int16\_t ax, ay, az ; // declaring variableds for storing accelometer values

int16\_t gx, gy, gz ; // declaring variableds for storing gyroscope values

SoftwareSerial ab(5,6); // declaring Serial communication pins for the bluetooth device

void setup ( )

{

sg90.attach ( servo\_pin );

Wire.begin ( );

ab.begin(9600); // begining bluetooth module communication baudrate

Serial.begin (19200); // begining python script communication baudrate for face recognise

Serial.println ( "Initializing the sensor" );

sensor.initialize ( );

Serial.println (sensor.testConnection ( ) ? "Successfully Connected" : "Connection failed");

delay (1000);

Serial.println ( "Taking Values from the sensor" );

delay (1000);

pinMode(13,OUTPUT); // output for the green LED

pinMode(10,OUTPUT); // output for the red LED

}

void loop ( )

{

sensor.getMotion6 (&ax, &ay, &az, &gx, &gy, &gz); // getting the sensor data

ax = map (ax, -17000, 17000, 0, 180) ; // mapping the sensor axis

if(Serial.read() == 'A') //loop for checking serial communication from the python script

{

sg90.write (90);

digitalWrite(10,LOW);

digitalWrite(13,HIGH);

delay (10000);

}

{

while(Serial.available()){ // loop for checking the data received from bluetooth module

for(int i=0; i<5; i++){

final[i] = Serial.read();

}

for(int i=0; i<5; i++){

if(final[i]==correct[i]){

pass\_true = 1;

}

else{

pass\_true = 0;

break;

}

}

}

if(pass\_true==1){

sg90.write (90);

digitalWrite(10,LOW);

digitalWrite(13,HIGH);

delay (10000);

pass\_true = 0;

}}

{

while(ab.available()){ // same loop if we dont give this loop the python script data generates some garbage values and continiously moves the servo

for(int i=0; i<5; i++){

final[i] = ab.read();

}

for(int i=0; i<5; i++){

if(final[i]==correct[i]){

pass\_true = 1;

}

else{

pass\_true = 0;

break;

}

}

}

if(pass\_true==1){

sg90.write (90);

digitalWrite(10,LOW);

digitalWrite(13,HIGH);

delay (10000);

pass\_true = 0;

}}

if(ax>140){ // for checking the feedback from the accelerometer

sg90.write (90);

delay (200);

digitalWrite(10,LOW);

digitalWrite(13,HIGH);

}

else if(ax<140){ // else part if all the other conditions fails the lock wont open

sg90.write(0);

delay (200);

digitalWrite(13,LOW);

digitalWrite(10,HIGH);

}

}

**SOURCE CODE: (Python face data creation part):**

# Creating database

# It captures images and stores them in datasets

# folder under the folder name of sub\_data

import cv2, sys, numpy, os

haar\_file = 'haarcascade\_frontalface\_default.xml'

# All the faces data will be

# present this folder

datasets = 'datasets'

# These are sub data sets of folder,

# for my faces I've used my name you can

# change the label here

sub\_data = 'Kaustav'

path = os.path.join(datasets, sub\_data)

if not os.path.isdir(path):

os.mkdir(path)

# defining the size of images

(width, height) = (130, 100)

#'0' is used for my webcam,

# if you've any other camera

# attached use '1' like this

face\_cascade = cv2.CascadeClassifier(haar\_file)

webcam = cv2.VideoCapture(0)

# The program loops until it has 30 images of the face.

count = 1

while count < 30:

(\_, im) = webcam.read()

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

faces = face\_cascade.detectMultiScale(gray, 1.3, 4)

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

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

face = gray[y:y + h, x:x + w]

face\_resize = cv2.resize(face, (width, height))

cv2.imwrite('% s/% s.png' % (path, count), face\_resize)

count += 1

cv2.imshow('OpenCV', im)

key = cv2.waitKey(10)

if key == 27:

break

**SOURCE CODE: (Python face recognition part):**

# -\*- coding: utf-8 -\*-

"""

Created on Wed Mar 27 03:28:28 2019

@author: kaust

"""

# It helps in identifying the faces

import cv2,numpy, os , openpyxl,serial,struct,time

size = 4

haar\_file = 'haarcascade\_frontalface\_default.xml'

datasets = 'datasets'

students=['']

Myserial = serial.Serial('COM3',baudrate=19200)

time.sleep(2)

a=0

b=0

x=0

y=0

# Part 1: Create fisherRecognizer

print('Recognizing Face Please Be in sufficient Lights...')

# Create a list of images and a list of corresponding names

(images, lables, names, id) = ([], [], {}, 0)

for (subdirs, dirs, files) in os.walk(datasets):

for subdir in dirs:

names[id] = subdir

subjectpath = os.path.join(datasets, subdir)

for filename in os.listdir(subjectpath):

path = subjectpath + '/' + filename

lable = id

images.append(cv2.imread(path, 0))

lables.append(int(lable))

id += 1

(width, height) = (130, 100)

# Create a Numpy array from the two lists above

(images, lables) = [numpy.array(lis) for lis in [images, lables]]

# OpenCV trains a model from the images

model = cv2.face.LBPHFaceRecognizer\_create()

model.train(images, lables)

# Part 2: Use fisherRecognizer on camera stream

face\_cascade = cv2.CascadeClassifier(haar\_file)

webcam = cv2.VideoCapture(0)

while True:

(\_, im) = webcam.read()

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

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

'''Myserial.write(str.encode('B'))'''

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

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

face = gray[y:y + h, x:x + w]

face\_resize = cv2.resize(face, (width, height))

a=int((2\*x+w)/2)

b=int((2\*y+h)/2)

p=int(a/3.66)

q=int(b/2.55)

'''Myserial.write(struct.pack('>BB', p,q))'''

# Try to recognize the face

prediction = model.predict(face\_resize)

cv2.rectangle(im, (x, y), (x + w, y + h), (0, 255, 0), 3)

if prediction[1]<100:

cv2.putText(im, '% s - %.0f' %

(names[prediction[0]], prediction[1]), (x-10, y-10),

cv2.FONT\_HERSHEY\_PLAIN, 1, (0, 255, 0))

Myserial.write(str.encode('A'))

else:

Myserial.write(str.encode('B'))

cv2.putText(im, 'not recognized',

(x-10, y-10), cv2.FONT\_HERSHEY\_PLAIN, 1, (0, 255, 0))

cv2.imshow('OpenCV', im)

key = cv2.waitKey(10)

if key == 27:

Myserial.write(str.encode('B'))

cv2.destroyAllWindows()

webcam.release()

**RESULT:**

Once all the program gets run, we can lock unlock the device if we provide the proper inputs to the algorithm.

**ERROR HANDLING / DEBUGGING:**

* Open cv module not found:

Just install the open cv module by typing in command prompt: pip install opencv-python

* Serial data communication related errors:

Check if the pyserial lib is installed in python or not

If not install install by “pip install pyserial”

* Check for the Arduino COM port and baud rated in the python scripts and also in the Arduino code

**Any more errors kindly contact me email-** [**kaustavbora11@gmail.com**](mailto:kaustavbora11@gmail.com)

**Project links:**

**GitHub link:** <https://github.com/Kaustav872/Automated-access-control-using-Facial-Recognition-Bluetooth-and-Gestures>

**Youtube project video link:** <https://www.youtube.com/watch?v=s-qWrtp8xl8&t=4s>