**Arduino code –**

// IR sensor sends a boot up signal and a light switch flip, computer detects face for security, and unlocks gestures for the rest

const int trigPin1 = 2; // the number of the trigger output pin ( sensor 1 )

const int echoPin1 = 3; // the number of the echo input pin ( sensor 1 )

const int trigPin2 = 4; // the number of the trigger output pin ( sensor 2 )

const int echoPin2 = 5; // the number of the echo input pin ( sensor 2 )

const int lightPin1 = 9; // the number of the LED input pin

const int lightPin2 = 10; // the number of the LED input pin

const int lightPin3 = 11; // the number of the LED input pin

const int motorPin1 = 9; // the number of the first motor input pin

const int motorPin2 = 10; // the number of the second motor input pin

const int motorPin3 = 6; // the number of the third motor input pin

const int IRPin1 = A1; // the number of the IR sensor input pin

int l1 = 0; // state variable for lights

// variables used for distance calculation

long duration;

int distance1, distance2;

float r;

unsigned long temp=0;

int temp1=0;

int l=0;

////////////////////////////////

void find\_distance (void);

// this function returns the value in cm.

void find\_distance (void)

{

digitalWrite(trigPin1, LOW);

delayMicroseconds(2);

digitalWrite(trigPin1, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin1, LOW);

duration = pulseIn(echoPin1, HIGH, 5000);// here this pulse in function wont wait more then 5000us for the ultrasonic sound to came back. (due to this it wont measure more than 60cm)

// it helps this project to use the gesture control in the defined space.

// so that, it will return zero if distance greater then 60m. ( it helps usually if we remove our hands infront of the sensors ).

r = 3.4 \* duration / 2; // calculation to get the measurement in cm using the time returned by the pulsein function.

distance1 = r / 100.00;

/////////////////////////////////////////upper part for left sensor and lower part for right sensor

digitalWrite(trigPin2, LOW);

delayMicroseconds(2);

digitalWrite(trigPin2, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin2, LOW);

duration = pulseIn(echoPin2, HIGH, 5000);

r = 3.4 \* duration / 2;

distance2 = r / 100.00;

delay(100);

}

void bootup(void)

{

a :

if(digitalRead(IRPin1) == LOW)

{

//digitalWrite(motorPin1, HIGH); // engages motor to trigger switch

//digitalWrite(motorPin2, LOW);

delay(100);

//digitalWrite(motorPin1, LOW);

//digitalWrite(motorPin2, HIGH);

Serial.println("security");

}

else goto a ;

}

void toggleLights()

{

if (Serial.available() > 0)

{

Serial.println("serial available");

l1 = Serial.parseInt();

if (l1 == 1)

{

digitalWrite(lightPin1, HIGH);

digitalWrite(lightPin2, HIGH);

digitalWrite(lightPin3, HIGH);

digitalWrite(13, HIGH);

}

else if (l1 == 0)

{

digitalWrite(lightPin1, LOW);

digitalWrite(lightPin2, LOW);

digitalWrite(lightPin3, LOW);

digitalWrite(13, LOW);

}

}

}

void setup()

{

// put your setup code here, to run once

Serial.begin(9600);

pinMode(trigPin1, OUTPUT); // initialize the trigger and echo pins of both the sensor as input and output:

pinMode(echoPin1, INPUT);

pinMode(trigPin2, OUTPUT);

pinMode(echoPin2, INPUT);

pinMode(lightPin1, OUTPUT);

pinMode(lightPin2, OUTPUT);

pinMode(lightPin3, OUTPUT);

pinMode(motorPin1, OUTPUT);

pinMode(motorPin2, OUTPUT);

pinMode(motorPin3, OUTPUT);

pinMode(13, OUTPUT);

pinMode(IRPin1, INPUT);

analogWrite(motorPin3,255);

bootup();

}

void loop()

{

// put your main code here, to run repeatedly:

toggleLights();

find\_distance(); // this function will stores the current distance measured by the ultrasonic sensor in the global variable "distance1 and distance2"

// no matter what, the program has to call this "find\_distance" function continuously to get the distance value at all time.

if(distance2<=35 && distance2>=15) // once if we placed our hands in front of the right sensor in the range between 15 to 35cm this condition becomes true.

{

temp=millis(); // store the current time in the variable temp. (" millis " Returns the number of milliseconds since the Arduino board began running the current program )

while(millis()<=(temp+300)) // this loop measures the distance for another 300 milliseconds. ( it helps to find the difference between the swipe and stay of our hand in front of the right sensor )

find\_distance();

if(distance2<=35 && distance2>=15) // this condition will true if we place our hand in front of the right sensor for more then 300 milli seconds.

{

temp=distance2; // store the current position of our hand in the variable temp.

while(distance2<=50 || distance2==0) // this loop will run untill we removes our hand in front of the right sensor.

{

find\_distance(); // call this function continuously to get the live data.

if((temp+6)<distance2) // this condition becomes true if we moves our hand away from the right sensor (\*\*but in front of it ). here " temp+6 " is for calibration.

{

// Serial.println("down"); // send "down" serially. Commented as function not required atm

}

else if((temp-6)>distance2) // this condition becomes true if we moves our hand closer to the right sensor.

{

// Serial.println("up"); // send "up" serially. Commented as function not required atm

}

}

}

else // this condition becomes true, if we only swipe in front of the right sensor .

{

Serial.println("next"); // send "next" serially.

}

}

else if(distance1<=35 && distance1>=15) // once if we placed our hands in front of the left sensor in the range between 15 to 35cm this condition becomes true.

{

temp=millis();

while(millis()<=(temp+300))

{

find\_distance();

if(distance2<=35 && distance2>=15) // if our hand detects in the right sensor before 300 milli seconds this condition becomes true. ( usually it happens if we swipe our hand from left to right sensor )

{

//Serial.println("change"); // send "change" serially. Commented as function not required atm

l=1; // store 1 in variable l. ( it avoids the program to enter into the upcoming if condition )

break; // break the loop.

}

}

if(l==0) // this condition will become true, only if we swipe our hand in front of left sensor.

{

Serial.println("select"); // send "select" serially.

while(distance1<=35 && distance1>=15) // this loop will rotate untill we removes our hand infront of the left sensor. this will avoid not to enter this if condition again.

find\_distance();

}

l=0; // make l=0 for the next round.

}

}

**Python script (dataset creation) –**

from imutils import paths

import face\_recognition

import pickle

import time

import cv2

import os

print("Creating data set ...")

imagePaths = list(paths.list\_images('Sriram')) # Replace 'Face' with your name if you want, but it should match the name of the folder containing your images

knownEncodings = []

knownNames = []

for (i, imagePath) in enumerate(imagePaths):

name = imagePath.split(os.path.sep)[-2]

image = cv2.imread(imagePath)

rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

boxes = face\_recognition.face\_locations(rgb,model='hog')

encodings = face\_recognition.face\_encodings(rgb, boxes)

for encoding in encodings:

knownEncodings.append(encoding)

knownNames.append(name)

data = {"encodings": knownEncodings, "names": knownNames}

f = open("face\_enc", "wb")

f.write(pickle.dumps(data))

f.close()

print()

print("Data set created")

time.sleep(3)

**Python script (main) –**

# Project python file

import pygame

from pygame import mixer

import face\_recognition

import imutils

import pickle

import time

import cv2

import serial

import pyautogui

import random

import os

import sys

z = 0

l1 = 0

s1 = 0

Arduino\_Serial = serial.Serial('COM9',9600) # Initialize serial and Create Serial port object called Arduino\_Serial

def write\_read(x) :

x=str(x)

Arduino\_Serial.write(bytes(x, 'utf-8'))

time.sleep(0.05)

data = str(Arduino\_Serial.readline())

print(data)

def verifyFace() : # face recognition function

global z

z = 0

cascPathface = os.path.dirname(

cv2.\_\_file\_\_) + "/data/haarcascade\_frontalface\_alt2.xml"

faceCascade = cv2.CascadeClassifier(cascPathface)

data = pickle.loads(open('face\_enc', "rb").read())

print("Streaming started")

video\_capture = cv2.VideoCapture(0)

while True:

ret, frame = video\_capture.read()

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

faces = faceCascade.detectMultiScale(gray,scaleFactor=1.1,minNeighbors=5,minSize=(60, 60),flags=cv2.CASCADE\_SCALE\_IMAGE)

rgb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

encodings = face\_recognition.face\_encodings(rgb)

names = []

for encoding in encodings:

matches = face\_recognition.compare\_faces(data["encodings"], encoding)

name = "Unknown"

if True in matches:

matchedIdxs = [i for (i, b) in enumerate(matches) if b]

counts = {}

for i in matchedIdxs:

name = data["names"][i]

counts[name] = counts.get(name, 0) + 1

name = max(counts, key=counts.get)

z = 1

names.append(name)

for ((x, y, w, h), name) in zip(faces, names):

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

cv2.putText(frame, name+' .Press q', (x, y), cv2.FONT\_HERSHEY\_SIMPLEX, 0.75, (0, 255, 0), 2)

cv2.imshow("Frame", frame)

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

break

time.sleep(5)

video\_capture.release()

cv2.destroyAllWindows()

def verifySecurity() :

verifyFace()

if z == 1 :

pygame.init()

y = mixer.Sound("identity\_verified.mp3") # Paste The audio file location

y.play()

elif z == 0 :

pygame.init()

y = mixer.Sound("identity\_unverified.mp3")

y.play()

sys.exit()

def playMusic() :

pygame.init()

ys = mixer.Sound("spectre.mp3")

ys.play()

def toggleLights() : # toggles secondary lights

global l1

l1 = not l1

if l1 == 0 :

write\_read(0)

elif l1 == 1 :

write\_read(1)

def choiceSelect(i) :

if i == 1 : # check face for security

verifySecurity()

elif i == 2 : # secondary lights

toggleLights()

elif i == 3 : # plays a random track from folder

playMusic()

a = 1

while 1:

incoming\_data = str (Arduino\_Serial.readline()) # read the serial data and print it as line

print(incoming\_data) # print the incoming Serial data

if 'security' in incoming\_data:

verifySecurity()

if 'next' in incoming\_data: # if incoming data is 'next'

if a == 1 :

# play voice (create new .py file to initialize audio)

pygame.init()

y = mixer.Sound("verify\_security.mp3")

y.play()

z = 0

a += 1

elif a == 2 :

# play voice (create new .py file to initialize audio)

pygame.init()

y = mixer.Sound("lights.mp3")

y.play()

a += 1

elif a == 3 :

# play voice (create new .py file to initialize audio)

pygame.init()

y = mixer.Sound("music\_q.mp3")

y.play()

a += 1

elif a > 3 :

a = 1

if 'select' in incoming\_data: # if incoming data is 'select'

choiceSelect(a-1)

incoming\_data = ""; # clears the data