Rapport de Projet :

Systeme d’ Access Facial pour un Module IOT

Realisé Par : Soufiane Kremcht

Introduction :

L'IoT, ou Internet des objets, désigne le processus de connexion d'objets physiques à Internet, des objets du quotidien tels que les dispositifs médicaux, appareils portables, appareils intelligents ou encore feux de circulation routière dans les villes intelligentes.

Comment fonctionne l'IoT ?

L'Internet des objets fait référence aux appareils physiques qui reçoivent et transfèrent des données sur des réseaux sans fil, avec une intervention humaine limitée. Cette technologie repose sur l'intégration d'un système informatique à toutes sortes d'objets.

Par exemple, un thermostat connecté (l'adjectif « connecté » renvoie souvent à l'Internet des objets) reçoit des données de géolocalisation issues de votre voiture connectée lorsque vous rentrez du travail et règle la température intérieure de votre logement avant votre arrivée. Aucune intervention de votre part n'est nécessaire, et le résultat s'avère meilleur que si vous aviez réglé manuellement le thermostat.

Reprenons l'exemple de la maison connectée. Pour prévoir le moment optimal de réglage du thermostat avant votre retour, votre système IoT peut se connecter à l'API Google Maps afin d'obtenir une modélisation du trafic en temps réel dans votre zone. Il peut également utiliser des données relatives à vos trajets habituels que votre voiture a recueillies sur une longue période. De plus, les données IoT recueillies par le thermostat de chaque client peuvent être analysées par des fournisseurs d'énergie dans le but d'optimiser leurs services à grande échelle.

_____

Parfois, ces systemes la besoin d'etre securisé contre les gens qui'sont pas l'authorisation de manipuler ces systemes.

Dans ce Projet, un systeme de access facials était developeé comme un prototype . le rapport suivants detaillé les conception, la developement avec les idées pour améliorer la development de ce concept.

On va realisé une systeme d' access Facial pour IOT, ce systeme va detecter le visage de l'utilisateur et le authoriser a utiliser le Composant IOT si il existe de la base des donneés .

Description generale de systeme :

Le systeme proposé est consiste à une microprocessur RPI4, un camera (IP ou USB), un module bluetooth pour la communication sans fils et un

Ardiuno our ESP32 microcontrolleur .le protocolue de communication WIFI dans le cas d'un camera IP.

Description materielle de systeme:

1.1- RPI4

Le systeme est basée sur le Raspberry PI4 qui s'agit d' un carte de processeur ARM un poil plus grande qu'une carte de credit.

Le Raspberry Pi possède un processeur ARM11 à 700 MHz. Il inclut 1, 2 4 ports USB, un port RJ45 et 256 Mo de mémoire vive pour le modèle d'origine jusqu'à 8 Go sur les dernières versions).

Le carte supporte aussi un module de bluetooth et WIFI pour la communication sans fil.ece card etait largement utilisé pour les taches de traitement des images.

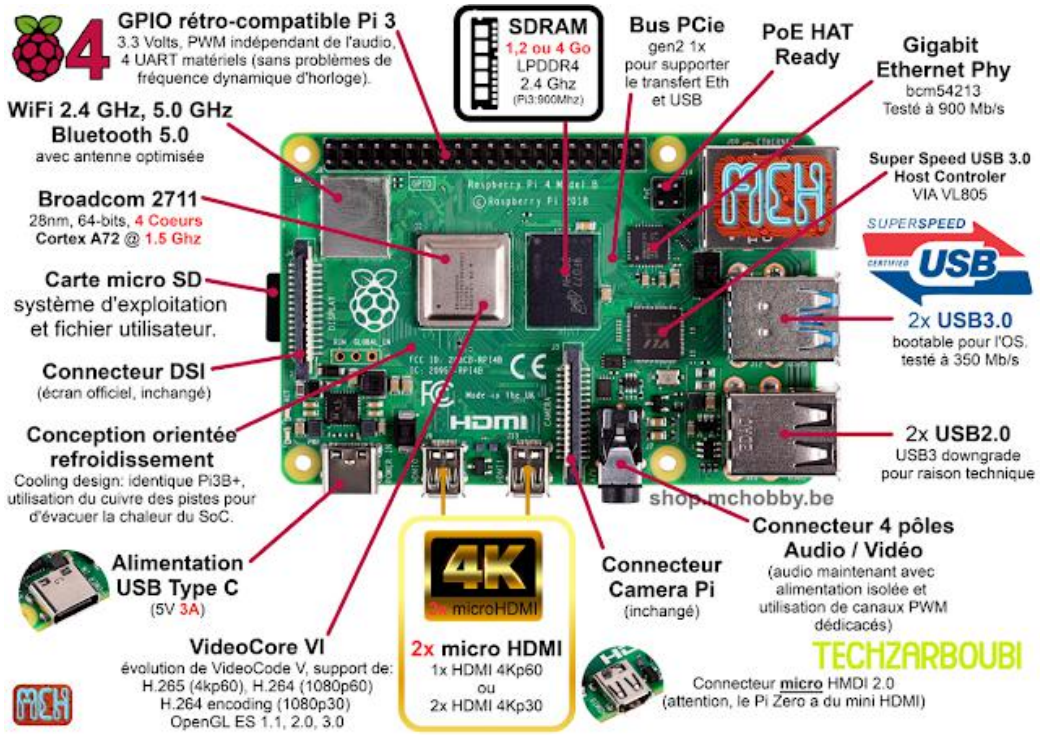


Figure 1 : les composants de Rasbperry Pi 4

1.2- Camera IP:

La camera IP s'agit d' un fusiion entre les capacités d'une caméra et les fonctionnalités d'un petit ordinateur connecté a l'internet.

Dans caméra IP, « IP » signifie « Internet Protocol ». La caméra IP est donc une caméra de surveillance connectée à Internet.

Une caméra IP ou caméra réseau est une caméra de surveillance utilisant le Protocole Internet pour transmettre des images et des signaux de commande via une liaison Fast Ethernet. Certaines caméras IP sont reliées à un enregistreur vidéo numérique (DVR) ou un enregistreur vidéo en réseau (NVR) pour former un système de surveillance vidéo.

L'avantage des caméras IP est qu'elles permettent aux propriétaires et aux entreprises de consulter leurs caméras depuis n'importe quelle connexion internet via un ordinateur portable ou un téléphone 3G.

Figure 2 : exemple d' un camera IP

1.3- Arduino avec une module bluetooth :

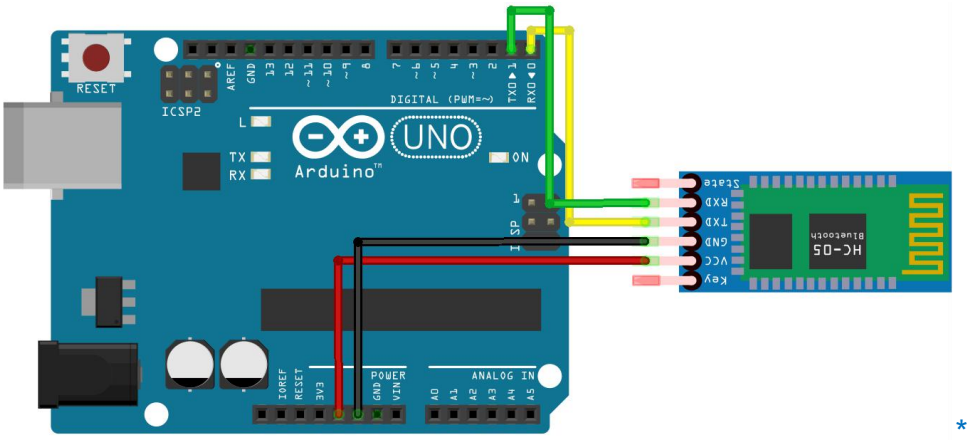


Figure 3: Arduino avec un module Bluetooth

1.4- ESP32 :

La carte ESP32 est un Systeme sur puce devéloppé par la société ESpressif dédié à l'internet des objets (IOT). et plus particulierement les communication sans fil.

Il integre d'un module Bluetooth 4.2 et Wi-F.un microcontolleur 32bit et il se caractérisé par un memoire de 520 KIO SRAM.

Figure 4: le module ESP32

1.5- Diagramme de bloc materielle:

Un Diagrame des blocs hardware dans la figure suivant, il detaille l'utilité de chaque composant et leur interaction.

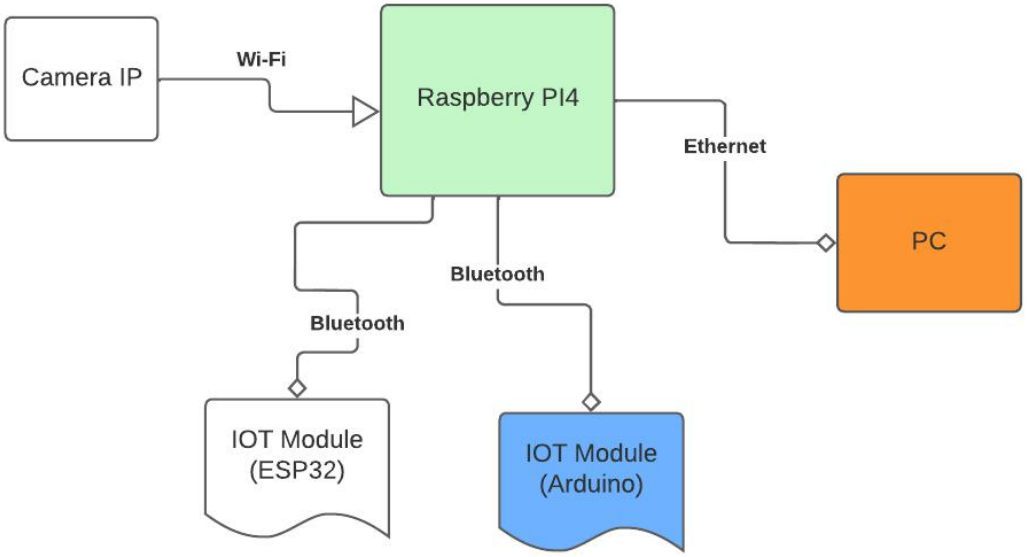


Figure 5: Schema fonctionnel du materiel.

Description logiciel de systeme:

Le logiciel etait réalise d'etre flexible et facile a comprendre. en basant sur la language de programmation Python .

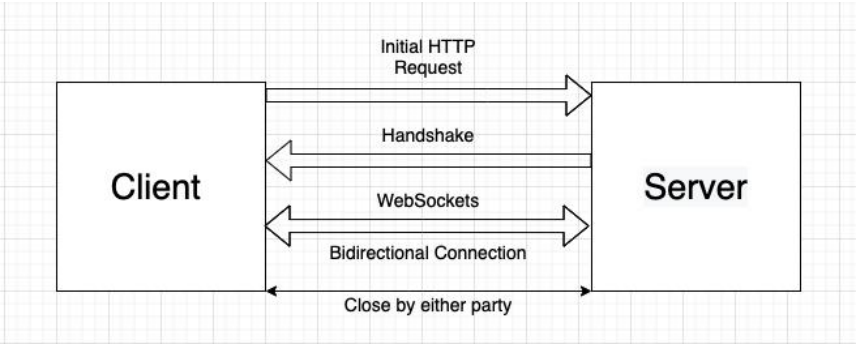
Les outils :

- Python : un langage de programmation libere interprété multiplatformes. It était le plus employé par les infromaticiens.

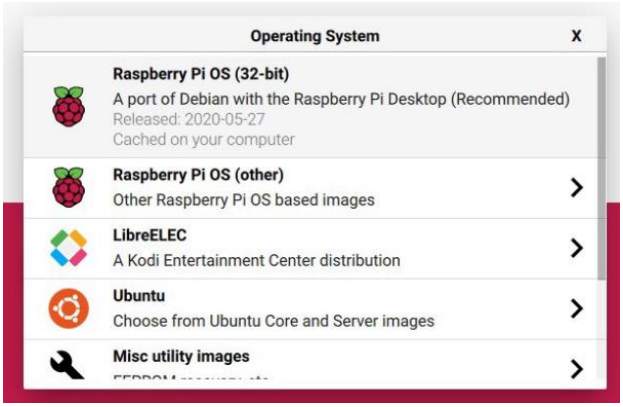
- OpenCV : une bibliothèque libre developper par intel. Spécialisée dans le traitement d'images en temps réels.

- Face Recognition Library: une bibliothéque python basé sur l' apprentissage automatique supervisée et la technologie deep learning pour reconnaissance de visage.

- Web socket : une technologie qui permet d'ouvrir de facon permanente une canal de communication bidirectionnel entre un client et un serveur.sun une seule socket TCP.



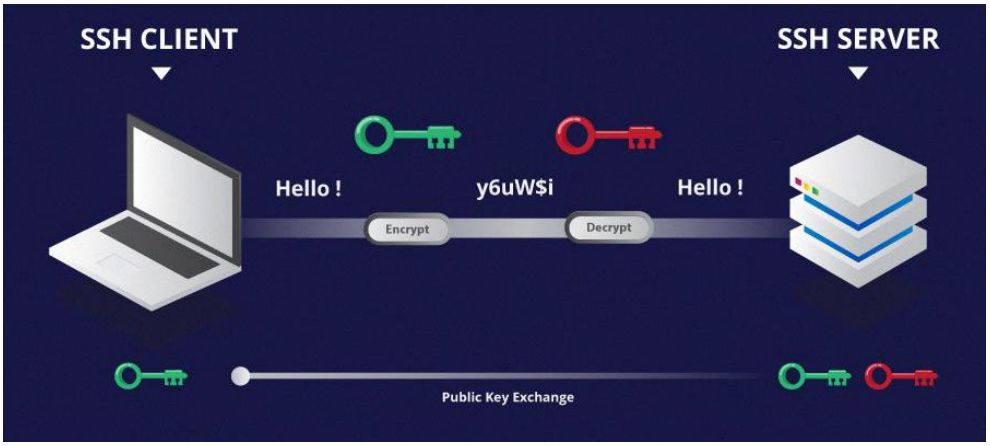
- Raspbian OS lite : un distribution linux devolopéé pour linux , contient seulemnt le terminal ... pour minimiser l'espace et l'energie.
st un système d'exploitation libre et gratuit basé sur Debian optimisé pour fonctionner sur les différents Raspberry Pi.



- Putty pour SSH : un logiciel pour la communication ssh avec des systeme d'exploitations Linux ..

SSH (ou Secure SHell) est un protocole qui facilite les connexions sécurisées entre deux systèmes à l'aide d'une architecture client/serveur et permet aux utilisateurs de se connecter à distance à des systèmes hôte de serveurs.

Les protocoles SSH définissent des normes pour l'exploitation sécurisée des services réseau d'hôtes non approuvés à travers des réseaux non sécurisés. Les communications entre un client et un serveur utilisant SSH sont cryptées et sont donc idéales pour une utilisation sur les réseaux non sécurisés.



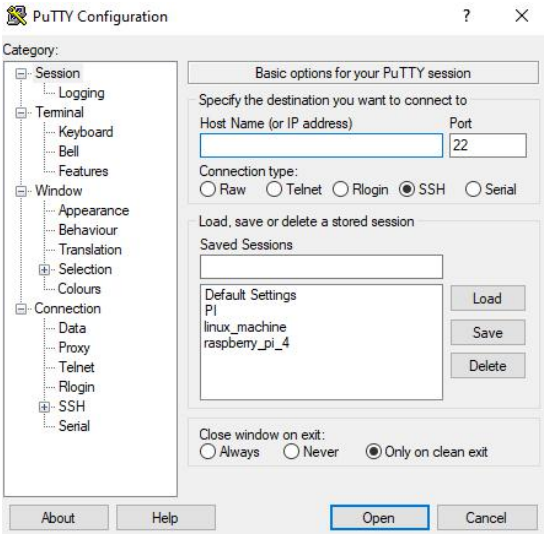
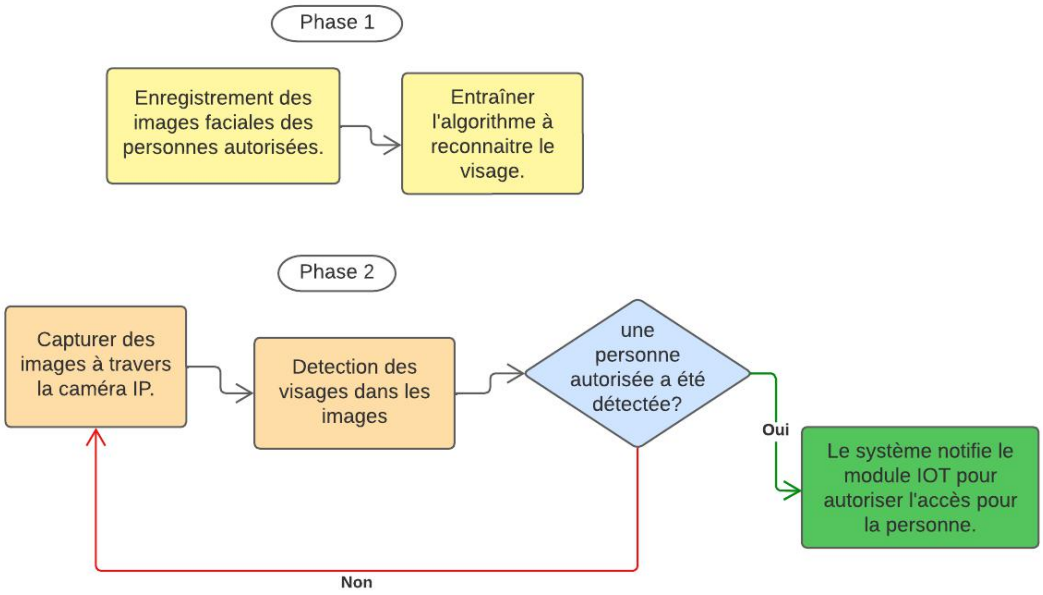


Schéma fonctionnele de logiciel:



Les modules de logiciel realisé:

```
pi@raspberrypi:~/Desktop $ cd facial_access_control_iot/
pi@raspberrypi:~/Desktop/facial_access_control_iot $ ls
assignment    before.save.1      facial_access_control_main.py   __pycache__
before.save   encodings.pickle   iot_sys_notification.py         train_face
pi@raspberrypi:~/Desktop/facial_access_control_iot $
```

- Reconnaissance de visage (headshots.py / train_model.py)

```python
# IP Camera URL
camera_url = "http://souf:123@192.168.0.100:5000/video"

#full_path = os.path.join()
cam = cv2.VideoCapture(camera_url)

cv2.namedWindow("press space to take a photo", cv2.WINDOW_NORMAL)
cv2.resizeWindow("press space to take a photo", 500, 300)

img_counter = 0

while True:
    ret, frame = cam.read()
    if not ret:
        print("failed to grab frame")
        break
    cv2.imshow("press space to take a photo", frame)

    k = cv2.waitKey(1)
    if k%256 == 27:
        # ESC pressed
        print("Escape hit, closing...")
        break
    elif k%256 == 32:
        # SPACE pressed
        img_name = "image_{}.jpg".format(img_counter)
        cv2.imwrite(img_name, frame)
        if not cv2.imwrite(img_name, frame):
            raise Exception("Could not write image")
        print("{} written!".format(img_name))
        img_counter += 1

cam.release()

cv2.destroyAllWindows()
```

Dans ce module ....,

```python
imagePaths = list(paths.list_images("dataset"))

# initialize the list of known encodings and known names
knownEncodings = []
knownNames = []

# loop over the image paths
for (i, imagePath) in enumerate(imagePaths):
    # extract the person name from the image path
    print("[INFO] processing image {}/{}".format(i + 1,
        len(imagePaths)))
    name = imagePath.split(os.path.sep)[-2]

    # load the input image and convert it from RGB (OpenCV ordering)
    # to dlib ordering (RGB)
    image = cv2.imread(imagePath)
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # detect the (x, y)-coordinates of the bounding boxes
    # corresponding to each face in the input image
    boxes = face_recognition.face_locations(rgb,
        model="hog")

    # compute the facial embedding for the face
    encodings = face_recognition.face_encodings(rgb, boxes)

    # loop over the encodings
    for encoding in encodings:
        # add each encoding + name to our set of known names and
        # encodings
        knownEncodings.append(encoding)
        knownNames.append(name)
# dump the facial encodings + names to disk
print("[INFO] serializing encodings...")
data = {"encodings": knownEncodings, "names": knownNames}
f = open("encodings.pickle", "wb")
f.write(pickle.dumps(data))
f.close()
```

- Capture les images et Detection de visage. (facial_access_control_Rpi4.py)

Cette partie ..

```python
from imutils.video import VideoStream
from imutils.video import FPS
import face_recognition
import imutils
import pickle
import time
import cv2

import socket

from threading import Thread
```

```python
class FaceRecognitionWrapper :
    def __init__(self, frame):
        self.frame = frame
        self.faceAllowed = False
    def start(self):
        Thread(target=self.update, arg=()).start()

    def update(self):···

    def isFaceAllowed(self):
        return self.faceAllowed
```

```python
if __name__ == "__main__":

    #Initialize 'currentname' to trigger only when a new person is identified.
    currentname = "unknown"
    #Determine faces from encodings.pickle file model created from train_model.py
    encodingsP = "encodings.pickle"

    # load the known faces and embeddings along with OpenCV's Haar
    # cascade for face detection
    print("[INFO] loading encodings + face detector...")
    data = pickle.loads(open(encodingsP, "rb").read())
    print("[INFO] loading file is successful")

    # Set your Host_IP & Port
    HOST_IP = '192.168.0.108'
    PORT = 8000
    ## Start a WebSocket
    client_socket = socket.socket()
    client_socket.connect((HOST_IP, PORT))  # ADD IP HERE
    # Make a file-like object out of the connection
    connection = client_socket.makefile('wb')

    camera_url = "http://souf:123@192.168.0.104:5000/video"
```

```python
    try:
        # initialize the video stream and allow the camera sensor to warm up
        vs = VideoStream(src=camera_url).start()
        start = time.time()
        while True:
            # loop over frames from the video file stream
            # grab the frame from the threaded video stream and resize it to 500px (to speedup processing)
            frame = vs.read()
            frame = imutils.resize(frame, width=500)
            face_recognize = FaceRecognitionWrapper(frame)
            updated_frame = face_recognize.read()

            ### Send data over the network using socket
            if face_recgnize.isFaceAllowed():
                #  Notify The IOT Systeme By Bluetooth
                IOTSysteme.notify()
            # Convert Frame to JPG & Save it as stream IO

            buffer = cv2.imencode('.jpeg', updated_frame)[1]
            #img_bytes = buffer.tobytes()

            io_buf = io.BytesIO(buffer)
            byte_im = io_buf.getvalue()
            io_buf.seek(0,2)

            print("Buffer data size\n " + str(len(io_buf.getvalue())))
            #connection.write(struct.pack('<L', len(io_buf.getvalue())))
            connection.write(struct.pack('<L', io_buf.tell()))
            connection.flush()

            # Rewind the stream and send the image data over the wire
            io_buf.seek(0)
            connection.write(io_buf.read())
        # Write a length of zero to the stream to signal we're done
        connection.write(struct.pack('<L', 0))
        vs.stop()
```

- Notification pour le system IOT (iot_sys_notification.py)

Lorsque l' algroithm trouve que le visage detecté etait dans le base de donnés, il va l'authorisé d' acces par envoyer un notification vers le module IOT.(pour notre Cas ESP 32..)

```python
#!/usr/bin/env python3
import serial
import time

## Notify  & Allow the User to Access the System by Sending a specific Code.

def notify_iot_arduino():
    ser = serial.Serial('/dev/ttyACM0', 9600, timeout=1)
    ser.reset_input_buffer()

    while True:
        ser.write(b"Hello from RPi4 !\n")
        line = ser.readline().decode('utf-8').rstrip()
        print(line)
        time.sleep(1)
```

- Affichage de image dans le PC (server_PC_streaming.py)

Pour afficher les images capter par le system avec un pC , on a va basé sur la technologie Websocket , on va ouvrir un canal de communication TCP pour envoyer les trames des images. Et le PC va capturer et lui afficher ..

```python
import io
import socket
import struct
import cv2
import numpy as np

HOST_IP = '192.168.0.105'
server_socket = socket.socket()
server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server_socket.bind((HOST_IP, 7210))
server_socket.listen(3)
# Accept a single connection and make a file-like object out of it
connection = server_socket.accept()[0].makefile('rb')
try:
    print("Recieving Image Data from")
    img = None
    while True:
        # Read the length of the image as a 32-bit unsigned int. If the
        # length is zero, quit the loop
        image_len = struct.unpack('<L', connection.read(struct.calcsize('<L')))[0]
        print("Image len " + str(image_len))
        if not image_len:
            print("Image len is NULL")
            break
        # Construct a stream to hold the image data and read the image data from the connection
        image_stream = io.BytesIO()
        image_stream.write(connection.read(image_len)
        # Rewind the stream, open it as an image with PIL and do some processing on it
        image_stream.seek(0)
        ##### Displaying Images

        frame = cv2.imdecode(np.frombuffer(image_stream.read(), np.uint8), 1)
        cv2.imshow('frame',frame)

        if cv2.waitKey(1) == ord('q'):
            break
```

Les Etapes de verification :

- Prendre les images de visage a reconnaitre par l'algorithme .

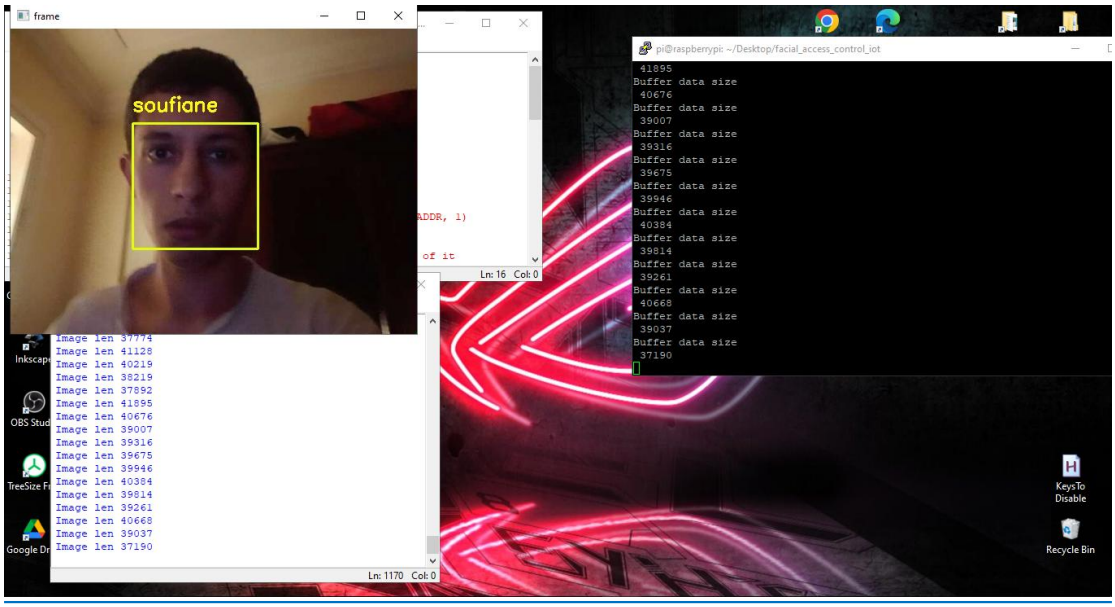
- Entrainer le modéle de la bibliothique de reconnaisance avec les données des visage.
- Capturer les images en temps réels avec un camére IP et faire la detection des visage.

- L'algorithm va essaye de reconnaitre de visage dans l'image et verifier si le person était dans la base des données.

- Le systeme va notifier la composant IOT.
- Affichage des images dans le PC avec la communication WebSocket .

- Communication de Camera IP avec RPI4 (Linux)
- Affichages de flux video dans le PC avec la communication websocket
- Analyser le visage detecter et Notifier le systeme IOT si le visage etait authoriser d' acceder le composant IOT.

Les Resultats :



Conclusion

Ce projet etait réalisé a l'objective de donner un prototype d' une methode de securiser les modules IOT utiliser dans les domaines domestiques, et dans l'entreprises.