

Platformă inteligentă pentru urmărirea accesului în laboratoarele de cercetare

Candidat: Ioan, Smetanca

Coordonator științific: Conf.dr.ing. Octavian, Cornea

Sesiunea: Iunie 2025

CUPRINS

1	INTRODUCERE	4
1.1	INFORMAȚII GENERALE	5
1.2	SCOPUL LUCRĂRII	5
1.3	MOTIVAȚIA ALEGERII TEMEI	6
2	RECUNOAȘTERE FACIALĂ ÎN TIMP REAL PE RASPBERRY PI CU OPENCV ȘI PYTHON	7
2.1	INSTALAREA RASPBERRY PI	8
2.2	INSTALAREA DEPENDENȚELOR NECESARE PENTRU RECUNOAȘTEREA FACIALĂ	9
2.3	DETECTAREA FEȚEI	9
2.4	RECUNOAȘTEREA FEȚEI	10
2.5	OpenCV	11
2.6	HAAR CASCADE	11
2.7	MEMORIE SWAP	11
2.8	ANTRENAREA MODELULUI PENTRU RECUNOAȘTEREA FACIALĂ: ..	14
2.9	HISTOGRAM OF ORIENTED GRADIENTS	17
2.10	TESTAREA MODELULUI	18
3	EMAIL API	20
3.1	INSTALAREA ȘI NOTIFICAREA PRIN EMAIL, FOLOSIND EMAIL-API ..	20
4	NODE-RED DASHBOARD	25
4.1	Instalare Node-Red pe Raspberry Pi	25
4.2	CONECTARE PRIN SSH LA RPI	26
4.3	FIREWALL	28
4.4	SSH	29
4.5	SERVER NODE-RED	29
4.6	MULTI-UPLOAD	37
4.7	AFIȘARE FOLDER/AFIȘARE FIȘIERE	38
4.8	ȘTERGERE FOLDER	39
4.9	INTRĂRI/IEȘIRI	40
4.10	ÎNREGISTRAREA STUDENȚILOR	42
5	RECUNOAȘTEREA FACIALĂ, LOCAL CU OPENCV, ONLINE CU FLASK ..	44
5.1	COMPARAREA METODELOR DE AFIȘARE A FLUXULUI VIDEO ÎN PYTHON PRIN CV2.IMSHOW() ȘI FLASK, EVIDENȚIIND AVANTAJELE ȘI DEZAVANTAJELE FIECĂREI ABOARDĂRI	44
5.2	AFIȘAREA CU CV2.IMSHOW()	45
5.3	AFIȘAREA CU FLASK(STREAMING VIDEO ÎN BROWSER)	45
6	ACCESAREA ȘI ADMINISTRAREA DE LA DISTANȚĂ A UNUI SERVER NODE-RED DASHBOARD	47
6.1	PORT FORWARDING	47
6.2	NGROK	48
N.	CONCLUZII	49
7	BIBLIOGRAFIE	50

8	ANEXE	52
8.1	Codul sursă pentru recunoașterea facială (var1 fără Flask)	52
8.2	Codul sursă pentru recunoașterea facială (var2 cu Flask).....	56
8.3	Cod sursă pentru colectarea imaginilor.....	62
8.4	Cod sursă pentru antrenarea modelelor	63
8.5	Funcțiile din Node-RED (var2)	65

1 INTRODUCERE

Recunoașterea facială reprezintă o tehnologie avansată ce se bazează pe analiza detaliată a trăsăturilor feței pentru a identifica sau verifica identitatea unei persoane. Această metodă care combină viziunea computerizată cu tehnici sofisticate de învățare automată, transformă o imagine facială într-o serie de indicatori numerici unici, oferind astfel posibilitatea de a face comparații precise între fețe. Această tehnologie funcționează pe mai multe etape, detectarea feței, extragerea caracteristicilor, compararea datelor.

În primul rând pentru detectarea feței, sistemul identifică prezența unei fețe în cadrul unei imagini sau al unui flux video, folosind algoritmi specializați care pot recunoaște rapid contururile și caracteristicile principale.

Dacă în acel flux video sau în imaginea respectivă a fost detectată o față, fața este procesată pentru a extrage un set de caracteristici esențiale, de la distanța dintre ochi și forma nasului până la texturile fine ale pielii. Cele mai noi modele de rețele neuronale (FaceNet sau ArcFace), convertesc informațiile într-un vector numeric de dimensiune fixă. [1],[2]

Odată convertite informațiile în vectori, acești vectori sunt apoi comparați cu cei dintr-o bază de date existentă, folosind metode matematice precum distanța Euclidiană sau similitudinea cosinus. Dacă diferențele sunt sub un anumit prag, se poate concluziona că fețele aparțin aceleiași persoane. [1],[2]

În ziua de azi deblocarea telefoanelor inteligente se poate face folosind o astfel de tehnologie (Face Id) până la controlul accesului în clădiri și zone restricționate, această tehnologie sporește nivelul de siguranță.

Această tehnologie poate fi folosită și în supravegherea publică (aeroporturi, gări și spații publice) pentru a identifica persoane suspecte sau infractori, contribuind la prevenirea activităților ilegale.

Sistemele de plată și bănci, aici recunoașterea facială poate ajuta la verificarea identității în tranzacțiile financiare, reducând riscul de fraudă.

Aceste sisteme de recunoaștere facială vin și cu anumite provocări cât și probleme tehnice și de acuratețe. Aceste modele pot prezenta performanțe inegale pentru anumite persoane, grupuri etnice, vârstă sau gen, în funcție de diversitatea și calitatea seturilor de date utilizate pentru antrenare. În funcție de robustețea și condiții variabile, schimbările de iluminare, expresiile faciale, unghiurile de captare pot afecta acuratețea acestor sisteme, necesitând tehnici de preprocesare și normalizare riguroase.

Stocare și gestionarea datelor faciale implică riscuri majore de confidențialitate, deoarece aceste informații sunt sensibile și odată compromise pot fi dificil de recuperat. Sistemele trebuie să fie capabile să distingă între o față reală și diverse tehnici de fraudă (fotografii videoclipuri), implementând așa ziși algoritmi anti-spoofing.

Utilizarea recunoașterii faciale în spații publice ridică multe semne de întrebare legate de consimțământul individual și de monitorizare în masă. În contextul global, diferite țări abordează reglementarea acestei tehnologii în moduri variate, iar evoluția normelor juridice este esențială pentru o utilizare responsabilă. [3],[4]

1.1 INFORMAȚII GENERALE

Această lucrare prezintă o soluție practică pentru identificarea facială utilizând un Raspberry Pi, un microcomputer accesibil și eficient din punct de vedere energetic. Procesul de recunoaștere facială este realizat printr-un script Python care analizează imaginile preluate de o cameră și compară fețele detectate cu o bază de date predefinită.

Pentru o gestionare mai intuitivă a sistemului, Node-RED Dashboard este utilizat pentru a crea o interfață grafică unde utilizatorul poate porni sau opri analiza facială și vizualiza rezultatele în timp real. Execuția scriptului Python este declanșată direct din Node-RED prin intermediul modulului node-red-node-pythonshell, eliminând necesitatea altor protocoale de comunicație. Funcționarea acestui proces este următorul:

- Capturarea imaginii se face datorită camerei care este conectată la Raspberry Pi care surprinde cadre video.

- Analiza și identificarea feței se face cu ajutorul scriptului scris în Python care prelucrează imaginile și le verifică dacă fețele detectate există în baza de date (în folderele respective).

- Interacțiunea cu Node-RED este foarte simplă astfel încât utilizatorul poate porni și opri recunoașterea facială printr-un buton din interfață Dashboard, care activează scriptul python prin pythonshell.

- Afișarea rezultatelor este la fel de simplă deoarece dacă o persoană este identificată, informațiile sunt trimise către Node-RED și afișate pe interfață web astfel încât oricine care are acces pe interfața respectivă poate vedea acest lucru în timp real.

Prin această metodă, se obține un sistem flexibil și ușor de utilizat, aplicabil în domenii precum securitatea accesului, educație și prezență, medicină și sănătate.

1.2 SCOPUL LUCRĂRII

Scopul acestei lucrări este de a dezvolta un sistem automatizat de verificarea prezenței a studenților, utilizând recunoașterea facială implementată pe Raspberry Pi. Prin acest sistem, identificarea studenților se realizează rapid și fără intervenție manuală, eliminând astfel metodele tradiționale de înregistrare a prezenței.

Pentru procesarea imaginilor și identificarea studenților se folosește un script Python care analizează cadrele video preluate de la o cameră conectată la Raspberry Pi. Integrarea cu Node-RED Dashboard permite afișarea rezultatelor într-o interfață grafică intuitivă, unde tutorii pot verifica în timp real prezența studenților. Scriptul Python este executat direct din Node-Red folosind modulul node-red-node-pythonshell, asigurând astfel o comunicare directă între procesarea datelor și interfața web.

Implementarea acestui sistem contribuie la eficientizarea procesului de verificare a prezenței cât și reduce riscul de fraudă (scrierea unui student pe foaie care nu este prezent în laborator).

1.3 MOTIVAȚIA ALEGERII TEMEI

Această temă a fost aleasă din dorința de a îmbunătăți modul în care se realizează verificarea prezenței studenților la cursuri. Metodele tradiționale (studenții să se scrie pe o foaie de hârtie) sunt consumatoare de timp și pot fi ușor fraudate. În plus, în cazul grupurilor mari de studenți, gestionarea manuală a prezenței devine dificilă și inefficientă.

Prin implementarea unui sistem automatizat bazat pe recunoașterea facială, prezența poate fi verificată rapid, fără intervenție din partea profesorului sau a studenților. Alegerea Raspberry Pi pentru dezvoltare a fost motivată de accesibilitatea și flexibilitatea acestuia, permițând crearea unui sistem compact și economic.

Integrarea cu Node-RED Dashboard oferă un mod simplu și eficient de afișare a datelor, astfel încât profesorii să poată monitoriza în timp real prezența studenților. Această soluție modernă contribuie și la prevenirea fraudelor (scrierea altui student care nu este prezent în laborator/sala de curs) făcând sistemul educațional mai transparent și mai bine organizat.

2 RECUNOAȘTERE FACIALĂ ÎN TIMP REAL PE RASPBERRY PI CU OPENCV ȘI PYTHON

Raspberry Pi este un SBC-low cost(Single Board Computer, care se referă la un computer complet funcțional integrat pe o singură placă de circuit fiind disponibile la un preț accesibil) a fost dezvoltat de fundația Raspberry Pi o organizație caritabilă din Marea Britanie.De la prima lansare în 2012 au fost lansate mai multe generații de computere Raspberry Pi, care pot fi clasificate în 3 modele distincte:Raspberry Pi A , B și 0.Fiecare dintre acestea având câte un sistem pe cip care constă dintr-un CPU integrat(unitate centrală de procesare) și un GPU(unitate de procesare grafică), memorie integrată și o intrare de alimentare de 5V DC.Toate modelele au de asemenea un port pentru a conecta o cameră dedicată,precum și o serie de pini de intrare/ieșire de uz general (GPIO) care pot fi utilizați pentru a comunica cu o gamă largă de electronice, de la led-uri și butoane, la servo-uri și motoare,reele de mare putere și o gamă largă de senzori .Plăcile de expansiune speciale care se conectează la pinii GPIO numite hardware atașat în partea de sus(HAT), pot oferi funcționalități suplimentare,variind de la gestionarea energiei, identificarea prin radiofrecvență (RFID) controlere de motor și înregistrare audio de înaltă calitate.Majoritatea modelelor dispun,de asemenea de o conexiune Ethernet și conectivitate Wireless(Wi-Fi și Bluetooth), care în combinație cu porturile GPIO ,conferă lui Raspberry Pi o versatilitate uriașă .Raspberry Pi are toate funcționalitățile unui computer standard .Ca atare putem conecta un mouse, o tastatură și un ecran fără nici o configurație și putem avea control asupra unui mediu desktop Linux ușor de utilizat sau a altor sisteme de operare populare,inclusive Windows 10 IoT și Android. Raspberry Pi poate fi folosit și ca unitate fără cap (fără tastatură, mouse și ecran atașat) care poate fi controlată de la distanță și programată pentru a rula scripturi în mod autonom folosind o gamă largă de limbaje de computer. Raspberry Pi este diferit de un microcontroler, cum ar fi Arduino sau recent lansat Raspberry Pi Pico, care poate fi programat să execute un singur program scris de utilizator și să comunice cu senzori și alte componente electronice.[5]

Atât configurarea cât și instalarea sistemului de operare a Raspberry Pi este foarte simplă.Folosind un alt computer,pur și simplu descărcăm Raspberry Pi Imager de pe site-ul oficial,conectăm cardul microsd și selectăm sistemul de operare dorit.Se recomandă cea mai recentă versiune a sistemului de operare Raspberry Pi(denumită anterior Raspbian) cu desktop.După ce cardul SD este scris conectăm Raspberry Pi și îl pornim prin conectarea la alimentare.O alternativă este să cumpărăm un card SD preincarcat cu o versiune instalabila a sistemului de operare.Pentru a ne putea conecta o să avem nevoie de un ecran o tastatură și un mouse.Cu toate acestea,cu câțiva pași simpli este de asemenea posibil să rulăm imediat cardul SD proaspăt instalat fără aceste periferice.La prima pornire mediul desktop Raspberry Pi OS,o să fim ghidați printr o serie scurtă de pași pentru a ne ajuta la configurarea sistemului și câțiva pași suplimentari de configurare și măsuri de Securitate.Trebuie să ne asigurăm că oprim întotdeauna Raspberry Pi înainte de a deconecta alimentarea pentru a evita pierderea datelor și posibil coruperea sistemului.[5]

Specificații Raspberry Pi 3 Model B+:

Raspberry Pi 3 Model B+

- 64-bit quad-core ARMv8 CPU
- 2.4&5 GHz 802.11b/g/n/ac Wireless LAN
- Bluetooth 4.2/BLE

2.1 INSTALAREA RASPBERRY PI

Primul pas pentru instalarea RASPBERRY Pi OS trebuie să descarcăm Raspberry Pi Imager[6].



Fig1.Descarcare Raspberry Pi Imager

După ce instalăm Raspberry Pi OS, o să fim nevoiți să alegem sistemul de operare și pe ce device să instalăm Raspberry Pi(usb sau micro sd).Noi am ales pe micro sd!



Fig 2.Alegerea sistemului de operare si a dispozitivului pentru scrierea Pi Imager
La “CHOOSE DEVICE” alegem Raspberry Pi 3, sistemul de operare (“CHOOSE OS”) alegem Raspberry Pi OS(64-bit) cel recomandat. La “CHOOSE STORAGE” putem instala sistemul de operare atât pe USB cât și pe microsd(dacă instalăm pe microsd o să avem nevoie de un adaptor pentru a face acest lucru).După care apăsăm next și la General ne facem setările dorite(Set Hostname,Configure Wireless LAN,Set Locale,settings),salvăm și așteptăm să înceapă procesul de instalare a sistemului de operare pe STICK/micro sd,în cazul nostru pe micro sd.

Dacă toți pașii au fost respectați ar trebui să apară o imagine în felul următor, urmând ca mai departe să apară configurațiile basic ale OS(Set Country,Create User,Select WiFi Network) și în cele mai multe cazuri un “update software”.

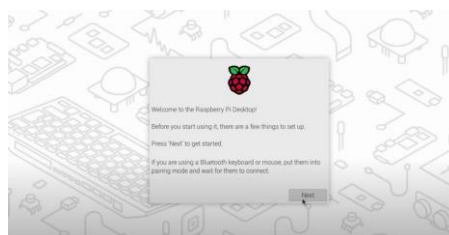


Fig 3. Instalarea sistemului de operare a avut succes

2.2 INSTALAREA DEPENDENȚELOR NECESARE PENTRU RECUNOAȘTEREA FACIALĂ

În acest pas o să instalăm OpenCV, face-recognition, imutils și o să modificăm în swapfile pentru a crește spațiul de schimb. Pentru utilizarea zilnică este suficientă o memorie swap de 100 Mbyte. Dar pentru construirea OpenCV o să avem nevoie de 5.8 GB de memorie.[7]

Vederea computerizată este una dintre cele mai fascinante și provocatoare sarcini din domeniul inteligenței artificiale. Vederea computerizată servește ca o punte de legătură între software-ul computerului și imaginile pe care le vedem în jurul nostru. Aceasta permite software-ului să înțeleagă și să învețe despre imaginile din mediul său.

De exemplu: un fruct este determinat pe baza culorii, formei și dimensiunii sale. Această sarcină poate părea simplă pentru creierul uman, însă, în cadrul procesului de vedere computerizată, mai întâi colectăm date, apoi efectuăm operațiuni de procesare a datelor și, ulterior, antrenăm și educăm modelul pentru a învăța să facă diferența între fructe pe baza dimensiunii, formei și culorii (în cazul nostru pentru fețe).[8],[9]

Scopul principal este de a identifica și înțelege imaginile și de a genera noi imagini care să fie mai utile pentru noi în diverse domenii ale vieții.

Termenul „OpenCV” este o abreviere pentru „open source computer vision” (vedere computerizată cu sursă deschisă). Arhitectura sa este alcătuită din software, baze de date și pluginuri preprogramate, care oferă suport pentru integrarea aplicațiilor de vedere computerizată[10]

Este unul dintre cele mai utilizate kituri de instrumente, având o comunitate mare de dezvoltatori. OpenCV este bine cunoscut pentru amploarea utilizării sale în scenarii reale din industrie.

OpenCV suportă limbajele de programare C/C++, Python și Java și poate fi utilizat pentru dezvoltarea de software de vedere computerizată pe platforme desktop și mobile, cum ar fi Windows, Linux, macOS, Android și iOS.

2.3 DETECTAREA FEȚEI

Detectarea feței este o formă de vedere computerizată care ajută la identificarea și vizualizarea trăsăturilor faciale în imagini capturate sau în videoclipuri în timp real. Această tehnică de detectare a obiectelor identifică instanțe ale unor artefacte semantice dintr-o anumită clasă (cum ar fi persoane, mașini sau case) în imagini și videoclipuri digitale.[11]

Recunoașterea facială a devenit din ce în ce mai importantă odată cu avansul tehnologiei, în special în domenii precum fotografia, apărarea și marketingul.

Detectarea feței a primit o atenție considerabilă în ultimii ani datorită aplicațiilor sale în interacțiunea dintre computere și oameni. Detectarea feței este un subset al procesării imaginii. Procesarea imaginii este, în principal, o tehnică utilizată pentru comprimarea, îmbunătățirea sau extragerea de informații valoroase din imagini.

Tehnologia de recunoaștere facială poate identifica una sau mai multe fețe într-o imagine. Un algoritm de identificare a feței trebuie, în esență, să clasifice imaginile în două categorii, pe baza existenței sau inexistenței unei fețe. Scopul algoritmului de detectare a feței este de a analiza în detaliu imaginea, de a identifica prezența fețelor și de a elimina fundalul imaginii.[11]

Erorile în detectarea feței sunt clasificate în două tipuri: fals negativ și fals pozitiv. Un fals pozitiv apare atunci când algoritmul identifică o față într-o imagine care nu conține nicio față. Un fals negativ apare atunci când algoritmul nu reușește să detecteze o față prezentă în imagine.[11]

Rata de detecție reprezintă raportul dintre numărul de fețe identificate de oameni și numărul de fețe detectate corect de sistem. Algoritmul de detectare a feței trebuie să aibă o rată de detecție cât mai mare posibil.

2.4 RECUNOAȘTEREA FEȚEI

Tehnologia de recunoaștere facială a devenit o caracteristică biometrică comună în laptopuri, telefoane mobile, sisteme de urmărire a prezenței și alte dispozitive în ultimii ani. Multe organizații folosesc sisteme de recunoaștere facială pentru supraveghere, automatizare pentru securitatea casei, securitate [12].

Accesul și procedurile de identificare penală din cauza progreselor tehnologice recente. În esență, software-ul de recunoaștere a feței compară trăsăturile geometrice ale unui individ cu trăsăturile feței. Abordarea convențională stochează imaginile utilizatorului într-o bază de date, care este apoi folosită pentru a compara fotografiile colectate cu imaginile utilizatorului. O varietate de trăsături au fost preluate și aplicate ca trăsături faciale, inclusiv colțuri, margini și descriptori de textură

Recunoașterea facială este cea mai avansată și rapidă tehnologie biometrică din lume. Aceasta utilizează într-un mod non-intruziv cea mai vizibilă componentă a corpului uman și anume fața. Potrivit datelor la nivel global, majoritatea persoanelor nu sunt conștiente de procesul de recunoaștere facială care se desfășoară asupra lor, ceea ce o face una dintre cele mai puțin invazive metode, cu un timp de procesare minim.[11]

Algoritmul de recunoaștere facială analizează diferitele trăsături ale unei fețe din imaginea de intrare. Această tehnologie biometrică a fost larg apreciată și, poate, exagerat promovată ca o metodă eficientă de identificare a posibilelor amenințări, cum ar fi teroriștii, escrocii și alții. Cu toate acestea, încă nu a fost acceptată pe scară largă pentru utilizare la nivel înalt.

Se preconizează că tehnologia biometrică de recunoaștere facială va depăși amprente digitale și va deveni cea mai comună metodă de identificare și autentificare a utilizatorilor în viitorul apropiat.

2.5 OpenCV

Este o bibliotecă masivă de procesare a imaginilor open-source, învățare automată și viziune pe computer. OpenCV este compatibil cu o gamă largă de limbaje de programare, inclusiv Python, C++ și Java. Va analiza fotografii și videoclipuri pentru a recunoaște artefacte, chipuri și chiar scrisul uman. Atunci când sunt asociate cu multe alte biblioteci, cum ar fi Numpy, o bibliotecă de înaltă performanță pentru mașini de strunjit, obțineți o performanță bună; adică toate serviciile care pot fi efectuate în canalul Numpy, de asemenea, să fie integrate cu OpenCV. Este scris pe baza C++ și are o interfață C++ ca interfață principală, dar are și o formare de limbă mai veche mai puțin robustă, dar încă detaliată. Atât cele mai recente tehnologii, cât și algoritmi sunt vizibili în C++ GUI. Legăturile Python, Java și MATLAB/OCTAVE sunt disponibile [13]. Au fost create wrapper-uri într-o varietate de limbaje de programare pentru a promova o acceptare mai largă. Pluginurile JavaScript pentru o variantă a funcțiilor OpenCV sunt publicate ca OpenCV.js în versiunea 3.4, care poate fi folosită pe platformele web. Proiectul OpenCV, care a fost dezvăluit oficial în 1999, a fost inițial programul de cercetare al Intel pentru a susține aplicații cu consum mare de CPU [13]. OpenCV este o platformă populară pentru implementarea algoritmilor de detectare și recunoaștere a feței. Sunt foarte multi algoritmi pentru detectare și recunoaștere a feței (LBP (Local Binary Pattern), Haar Cascade, EigenFaces, FisherFaces, LBPH (Local Binary Pattern Histogram), YOLO, Faster R-CNN, Single Shot Detectors (SSDs)) [14], [15]. Noi am folosit HAAR CASCADE.

IMUTILS este un pachet Python care oferă funcții simple pentru a facilita operațiunile grafice de bază ale OpenCV.

2.6 HAAR CASCADE

Cascada Haar este o metodă eficientă pentru detectarea obiectelor. Este o metodă bazată pe învățarea automată în care o cascadă de acțiuni este învățată dintr-un număr mare de imagini pozitive și negative. Se obișnuiește să vadă lucrurile în cadre diferite [15]

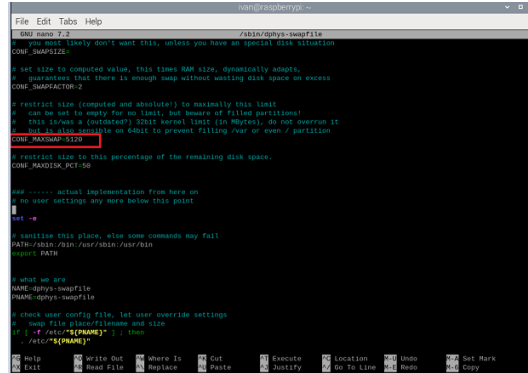
2.7 MEMORIE SWAP

Spațiul de schimb este utilizat atunci când cantitatea de memorie fizică (RAM) este plină. Dacă sistemul are nevoie de mai multe resurse de memorie și memoria RAM este plină, paginile inactive din memorie sunt mutate în spațiul de swap. În timp ce spațiul de schimb poate ajuta mașinile cu o cantitate mică de RAM, nu ar trebui să fie considerat un înlocuitor pentru RAM. [16]

Spațiul de schimb este situat pe hard disk-uri, care au un timp de acces mai lent decât memoria fizică. Spațiul de swap poate fi o partiție de swap dedicată (recomandat), un fișier de swap sau o combinație de partiții de swap și fișiere de swap. [16]

În anii trecuți, cantitatea recomandată de spațiu de schimb a crescut liniar cu cantitatea de RAM din sistem. Cu toate acestea, sistemele moderne includ adesea sute de gigaocteți de memorie RAM. Intrăm în terminal pentru a face aceste setări pentru Raspberry Pi

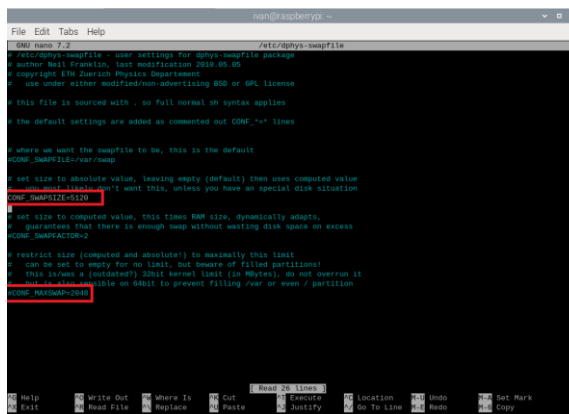
Prima comandă: `sudo nano /sbin/dphys-swapfile`
-se folosește pentru a mări limita(`CONF_MAXSWAP`)
-la ieșire folosim următoarea combinație de taste (`Ctrl+X“y”`)



```
File Edit Tabs Help
/sbin/dphys-swapfile
You most likely don't want this, unless you have a special disk situation
CONF_SWAPSIZE=
# set size to computed value, this times RAM size, dynamically adapts.
# guarantees that there is enough swap without wasting disk space on excess
CONF_SWAPFACTOR=2
# restrict size (computed and absolute) to maximally this limit
# can be set to empty for no limit, but beware of filled partitions!
# this is/was a (deprecated) 32bit kernel limit (in Mbytes), do not overrun it
# can be 4096 to prevent filling /var or even / partition
CONF_MAXSWAP=5120
# restrict size to this percentage of the remaining disk space.
CONF_MAXDISK_PCT=50
see ----- actual implementation from here on
# no user settings any more below this point
!
set -e
# enable this place, also some commands may fail
PATH=/sbin:/bin:/usr/sbin:/usr/bin
export PATH
# what we are
NAME=dphys-swapfile
PRNAME=dphys-swapfile
# check user config file, let user override settings
# check file permissions and size
if [ -f /etc/${PRNAME} ] ; then
    . /etc/${PRNAME}
fi
```

Fig4.Limita maximă pentru memoria swap care poate fi utilizată sau alocată

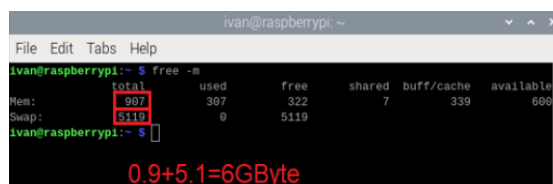
A doua comandă:
-`sudo nano /etc/dphys-swapfile`
-dimensiunea de memorie necesară (`CONF_SWAPSIZE`) și o să modificăm
“`CONF_SWAPSIZE`” cu 5120(dacă avem 1GB RAM) și comentăm “`CONF_MAXSWAP`”.
-la ieșire folosim următoarea combinație de taste (`Ctrl+X“y”`)



```
File Edit Tabs Help
/etc/dphys-swapfile
# dphys-swapfile - user settings for dphys-swapfile package
# author Neil Franklin, last modification 2018.05.05
# copyright the German Physics Department
# use under either modified/non-advertising BSD or GPL license
# this file is sourced with . so full normal sh syntax applies
# the default settings are added as commented out CONF="+" lines
# where we want the swapfile to be, this is the default
CONF_SWAPFILE=/var/swap
# set size to absolute value, leaving empty (default) then uses computed value
# can be set to empty for no limit, but beware of filled partitions!
CONF_SWAPSIZE=5120
# set size to computed value, this times RAM size, dynamically adapts.
# guarantees that there is enough swap without wasting disk space on excess
CONF_SWAPFACTOR=2
# restrict size (computed and absolute) to maximally this limit
# can be set to empty for no limit, but beware of filled partitions!
# this is/was a (deprecated) 32bit kernel limit (in Mbytes), do not overrun it
# can be 4096 to prevent filling /var or even / partition
CONF_MAXSWAP=2048
```

Fig5. Configurare swap manual: 5GB alocată

A treia comanda :
`free -m`



```
ivan@raspberrypi: ~
File Edit Tabs Help
ivan@raspberrypi:~$ free -m
              total        used        free      shared  buff/cache   available
Mem:           907           307           322            7           339           600
Swap:          5119             0          5119
ivan@raspberrypi:~$
```

0.9+5.1=6GByte

Fig6.Memorie de 1GB RAM si 5GB memorie swap

Dacă avem mai multă memorie ram, putem aloca mai puțină memorie swap.

În final o să restartăm sistemul cu următoarea comandă:

„sudo reboot”

O să rulăm următoarele comenzi din tabel:

1	sudo apt install cmake build-essential pkg-config git
2	sudo apt install libjpeg-dev libtiff-dev libjasper-dev libpng-dev libwebp-dev libopenexr-dev
3	sudo apt install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev libxvidcore-dev libx264-dev libdc1394-22-dev libgstreamer-plugins-base1.0-dev libgstreamer1.0-dev
4	sudo apt install libgtk-3-dev libqtgui4 libqtwebkit4 libqt4-test python3-pyqt5
5	sudo apt install libatlas-base-dev liblapacke-dev gfortran
6	sudo apt install libhdf5-dev libhdf5-103
7	sudo apt install python3-dev python3-pip python3-numpy
8	git clone https://github.com/opencv/opencv.git
9	git clone https://github.com/opencv/opencv_contrib.git
10	mkdir ~/opencv/build
11	cd ~/opencv/build
12	cmake -D CMAKE_BUILD_TYPE=RELEASE \
13	-D CMAKE_INSTALL_PREFIX=/usr/local \
14	-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules \
15	-D ENABLE_NEON=ON \
16	-D ENABLE_VFPV3=ON \
17	-D BUILD_TESTS=OFF \
18	-D INSTALL_PYTHON_EXAMPLES=OFF \
19	-D INSTALL_PYTHON_EXAMPLES=OFF \
20	-D CMAKE_SHARED_LINKER_FLAGS=-latomic \
21	-D BUILD_EXAMPLES=OFF ..
22	make -j\$(nproc)
23	sudo make install
24	sudo ldconfig

Acum putem instala si face recognition si imutils pentru recunoastere faciala:

Install face recognition:

-pip install face-recognition

Install imutils:

-pip install imutils

2.8 ANTRENAREA MODELULUI PENTRU RECUNOAȘTEREA FACIALĂ:

Executam această comandă pentru a obține toate fișierele/folderurile necesare:
„**git clone https://github.com/Ivan13s/facial_recognition**”

Click dreapta in folderul dataset pentru a crea un nou folder

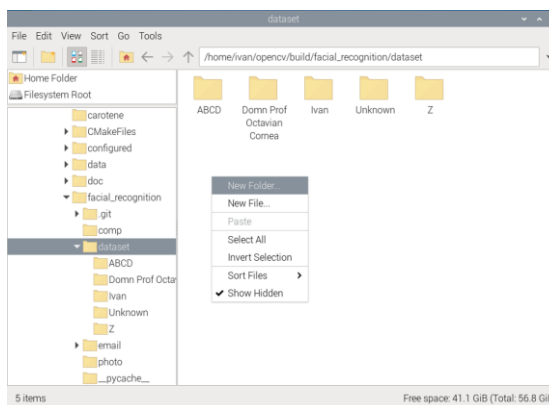


Fig7.Creare folder in dataset

Introduceți prenumele(dvs sau al altei persoane) pentru numele folderului nou creat.

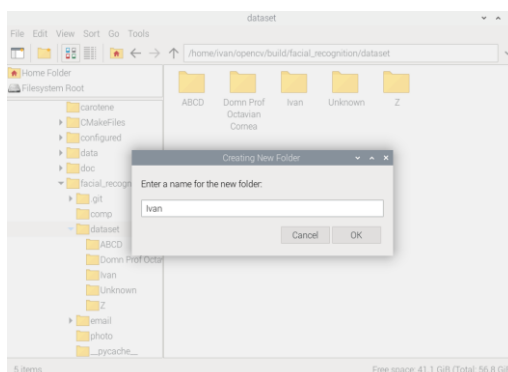
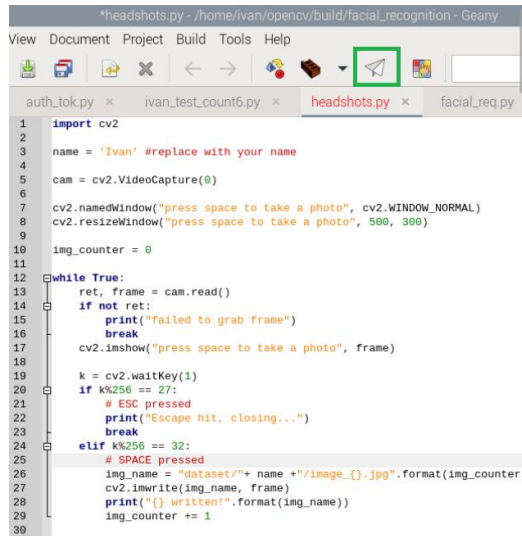


Fig8.Denumirea folderului cu numele sau prenumele nostru

Următorul pas este să ne întoarcem în facial recognition și să deschidem fișierul headshots.py (click dreapta pe acest fișier și îl deschidem cu Geany)



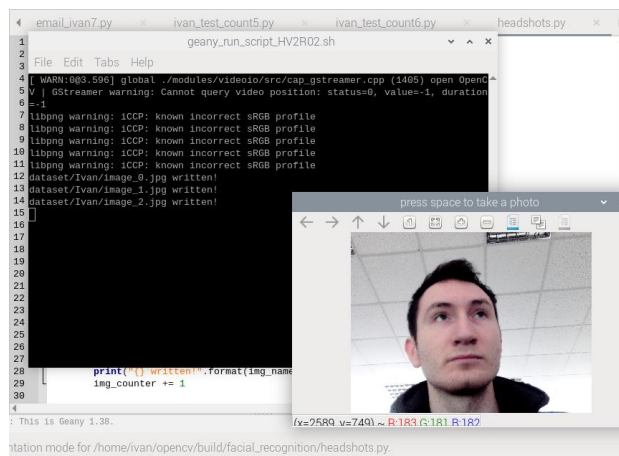
```

1 import cv2
2 name = 'Ivan' #replace with your name
3
4
5 cam = cv2.VideoCapture(0)
6
7 cv2.namedWindow("press space to take a photo", cv2.WINDOW_NORMAL)
8 cv2.resizeWindow("press space to take a photo", 500, 300)
9
10 img_counter = 0
11
12 while True:
13     ret, frame = cam.read()
14     if not ret:
15         print("Failed to grab frame")
16         break
17     cv2.imshow("press space to take a photo", frame)
18
19     k = cv2.waitKey(1)
20     if k%256 == 27:
21         # ESC pressed
22         print("Escape hit, closing...")
23         break
24     elif k%256 == 32:
25         # SPACE pressed
26         img_name = "dataset/" + name + "/image_{}.jpg".format(img_counter)
27         cv2.imwrite(img_name, frame)
28         print("{} written!".format(img_name))
29         img_counter += 1
30

```

Fig9.Redenumirea in fișierul headshots.py pe linia 2 cu numele nostru și rularea scriptului

Pe linia 3 din figura scrieți numele dumneavoastră(al folderului pe care l-ați creat și nu uitați să salvați fișierul după modificare “Ctrl+s”).Apăsați pictograma celui avion de hârtie (e marcat cu un pătrat verde în Fig9) pentru a fi ușor de identificat și pentru a rula headshots.py



```

1 email_ivan7.py
2
3 geany_run_script_HV2R02.sh
4 File Edit Tabs Help
5 [WARN:003.596] global: ./modules/videoio/src/cap_gstreamer.cpp (1405) open OpenCL
6 [GStreamer warning: Cannot query video position: status=0, values=-1, duration=0-1
7 libpng warning: iCCP: known incorrect sRGB profile
8 libpng warning: iCCP: known incorrect sRGB profile
9 libpng warning: iCCP: known incorrect sRGB profile
10 libpng warning: iCCP: known incorrect sRGB profile
11 libpng warning: iCCP: known incorrect sRGB profile
12 Dataset/Ivan/image_0.jpg written!
13 Dataset/Ivan/image_1.jpg written!
14 Dataset/Ivan/image_2.jpg written!
15
16
17
18
19
20
21
22
23
24
25
26
27
28 print("{} written!".format(img_name))
29 img_counter += 1
30
31
32 This is Geany 1.38.
33
34 station mode for /home/ivan/opencv/build/facial_recognition/headshots.py.
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
248
```

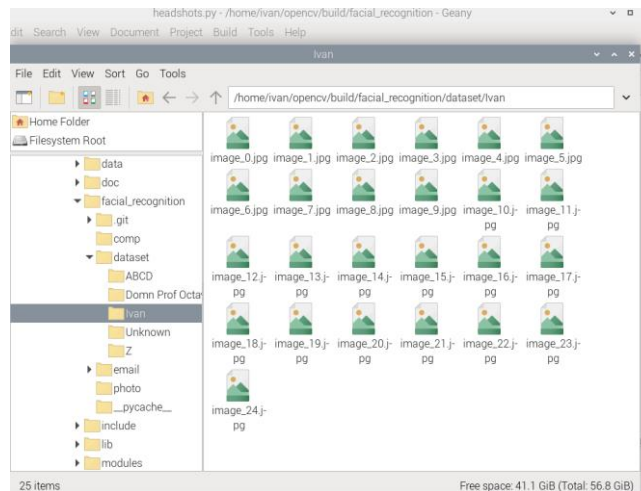



Fig11.Fotografiile pregatite pentru antrenament

Acestea sunt fotografiile de antrenament, în cazul în care doriți să faceți poze, nu uitați că "img_counter=0" este setat la 0 în "headshots.py" în cazul în care doriți să continuați să faceți fotografii, de exemplu fotografia nr 25, ideal ar fi să porniți cu counter-ul de la 25 "img_counter=25" în cazul în care lăsați acel counter la 0 se va suprascrie fiecare fotografie.

Dacă toți pașii au fost respectați trecem la antrenarea modelului

Mergem în terminal și trebuie să ajungem la folderul facial_recognition.

-ls (pentru a vedea toate folderele disponibile, putem folosi de mai multe ori această comandă pentru a vedea de fiecare dată ce foldere avem în directorul curent)

"-cd opencv "

"-ls "

"-cd build/"

"-ls"

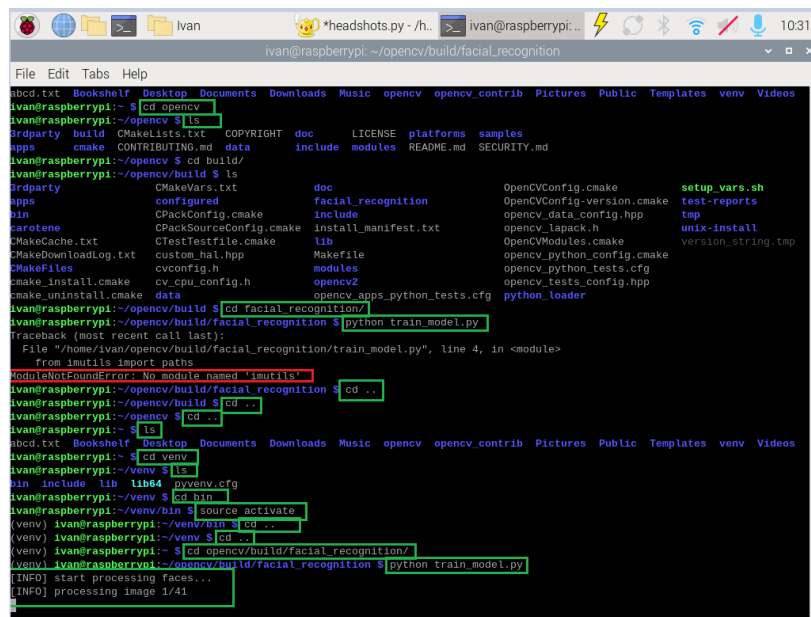
"-cd facial_recognition"

Putem folosi aceste comenzi în acest mod, dar dacă deja știm calea putem folosi direct "cd opencv/ build/facial_recognition".

Odată ce ajungem în folderul facial_recognition rulăm următoarea comandă pentru a antrena modelul:

"python train_model.py"

La rularea acestui script o să primim un mesaj în consolă „[INFO] start processing face...”, și avem o listă cu absolut ce conține fiecare folder în dataset și anume o să fie ceva asemănător „dataset/Ivan/photo...”, „Ivan” este folderul pe care l-am creat (numele folderului devine numele persoanei). Pentru fiecare imagine din folder se deschide imaginea se convertește la RGB deoarece face_recognition are nevoie de RGB, se detectează acele fețe din imagine urmând ca pe urmă să se creeze un vector numeric care descrie fața, salvează acel vector și numele persoanei într-o listă și într-un final salvează totul într-un fișier numit „encodings.pickle”, noi o să folosim acest fișier pentru recunoașterea facială în timp real.



```
ivan@raspberrypi: ~/opencv/build/facial_recognition
ivan@raspberrypi:~/opencv $ cd opencv
ivan@raspberrypi:~/opencv $ ls
apps      cmake  CONTRIBUTING.md  data      include  modules  README.md  SECURITY.md
3rdparty  build  CMakeLists.txt   COPYRIGHT doc        LICENSE    platforms  samples
ivan@raspberrypi:~/opencv $ cd build/
ivan@raspberrypi:~/opencv/build $ ls
3rdparty      CMakeVars.txt      doc                OpenCVConfig.cmake  setup_vars.sh
apps          configured          facial_recognition OpenCVConfig-version.cmake  test-reports
bin           CPackConfig.cmake  include            opencv_data_config.hpp  tmp
bartorene    CPackSourceConfig.cmake  install_manifest.txt  opencv_lapack.h          unix-install
CMakeCache.txt  CTestTestfile.cmake  lib                opencv_modules.cmake     version_string.tmp
CMakeDownloadLog.txt  custom_hal.hpp      Makefile           opencv_python_config.cmake
CMakeFiles     cvconfig.h          modules            opencv_python_tests.cfg
cmake_install.cmake  cv_cpu_config.h    opencv2            opencv_tests_config.hpp
cmake_uninstall.cmake  data              opencv_apps_python_tests.cfg  python_loader

ivan@raspberrypi:~/opencv/build $ cd facial_recognition/
ivan@raspberrypi:~/opencv/build/facial_recognition $ python train_model.py
Traceback (most recent call last):
  File "/home/ivan/opencv/build/facial_recognition/train_model.py", line 4, in <module>
    from imutils import paths
ModuleNotFoundError: No module named 'imutils'
ivan@raspberrypi:~/opencv/build/facial_recognition $ cd ..
ivan@raspberrypi:~/opencv $ cd ..
ivan@raspberrypi:~ $ ls
abcd.txt  Bookshelf  Desktop  Documents  Downloads  Music  opencv  opencv_contrib  Pictures  Public  Templates  venv  Videos
ivan@raspberrypi:~ $ cd venv
ivan@raspberrypi:~/venv $ ls
bin  include  lib  lib64  pyvenv.cfg
ivan@raspberrypi:~/venv $ cd bin
ivan@raspberrypi:~/venv/bin $ source activate
(venv) ivan@raspberrypi:~/venv/bin $ cd ..
(venv) ivan@raspberrypi:~/venv $ cd ..
(venv) ivan@raspberrypi:~/ $ cd opencv/build/facial_recognition/
(venv) ivan@raspberrypi:~/opencv/build/facial_recognition $ python train_model.py
[INFO] start processing faces...
[INFO] processing image 1/41
```

Fig12.Pașii necesari pentru a rula fișierul de antrenament train_model.py

În cazul în care o să primiți o eroare “No module named ‘imutils’” este posibil să nu fii instalat imutils și trebuie folosită din nou comanda” pip install imutils “.O altă variantă posibilă este să fii fost pe “virtual environments” când au fost instalate toate bibliotecile necesare, în cazul nostru este instalat pe venv(virtual environments).

Note de cod (train_model.py)

Set de date: train_model.py va analiza fotografiile din folderul setului de date. Organizați-vă fotografiile în dosare după numele persoanei. De exemplu, creați un dosar nou numit (numele dvs sau al persoanei) și plasați toate fotografiile feței (ale persoanei respective) în folderul pe care l-ați creat (folderul setului de date “dataset”).

Codificări: train_model.py va crea un fișier denumit encodings.pickle care conține criteriile de identificare a fețelor în pasul următor.

Metoda de detectare: Folosim metoda de detectare HOG (Histogram of Oriented Gradients).

2.9 HISTOGRAM OF ORIENTED GRADIENTS

Este un descriptor de caracteristici care este utilizat de obicei în domeniile de Vederea artificială și procesare de imagini pentru detectarea obiectelor.Pentru a fi mai ușor de procesat o imagine este procesată în alb-negru(tonuri de gri).Se calculează gradientii pentru fiecare pixel din imagine (direcția și puterea liniilor),așa ne dăm seama cum sunt orientate marginile obiectelor,vertical sau orizontal.Imaginea se împarte în bucăți mai mici iar pentru fiecare bucățică se creează o histogramă pentru a arată cât de des apar anumite orientări ale liniilor.Pentru a compensa diferențele de luminozitate și contrast bucățile mai mari combină mai multe celule și normalizează datele.La final toate informațiile sunt puse într-un vector lung care descrie imaginea urmând ca acest vector să fie folosit pentru a antrena un algoritm,un clasificator să recunoască obiectele specific.În

general HOG este folosit la găsirea oamenilor în imagini sau videoclipuri, poate de asemenea poate fi folosit și pentru alte obiecte dacă este antrenat corespunzător.[17]

Avantaje

Funcționează bine chiar și când iluminarea sau umbrele se schimbă
Este eficient pentru obiecte cu margini clare cum ar fi pietonii sau mașinile.

Dezavantaje

Poate fi lent pentru imagini mari sau în timp real
Nu funcționează la fel de bine dacă obiectele nu au margini clare

2.10 TESTAREA MODELULUI

Acum să testăm modelul pe care tocmai l-am antrenat.

Pașii pentru a crea un venv puteți urma pașii de mai jos:

Cel mai simplu este să mergem în directorul unde dorim să avem acel venv după care folosim următoarea comandă

„python -m venv venv”

putem să îi atribuim și o altă denumire, nu este neapărat venv, dar noi l-am lăsat venv pentru a fi mai intuitiv.

Pentru a verifica dacă modelul a fost antrenat corect rulăm următoarea comandă:

`python facial_req.py`

Dacă o să apară persoane necunoscute în apropiere o să se facă poză acelor persoane și o să fie mutate în folderul “Unknown”, presupunând că cunoaștem persoanele respective putem urma pașii de mai sus pentru a antrena modelul să recunoască și persoanele pe care le recunoaștem noi, doar că de data avem pozele și doar le putem muta, este și o variantă prin care putem urma metoda respectivă să facem poze apăsând tasta space, oricum în acest pas am verificat să vedem dacă am fost recunoscuți după antrenamentul pe care l-am făcut. În Fig 13 avem imaginea pe care am obținut-o după ce am rulat fișierul de recunoaștere facială pentru a vedea dacă antrenamentul pe care l-am făcut anterior a avut succes și dacă totul este în regulă și nu avem erori

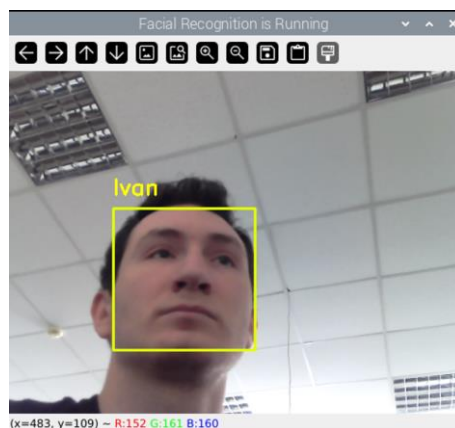


Fig13. Testare în timp real a recunoașterii faciale

După cum putem observa, modelul pe care tocmai l-am antrenat funcționează. Mai departe, o să implementăm astfel încât, dacă o persoană necunoscută este detectată, să trimită un email. Pentru a face acest lucru posibil, o să utilizăm GMAIL API.

3 GMAIL API

GMAIL API este un API RESTful care poate fi folosit pentru a accesa căsuțe poștale Gmail și trimite e-mail. Pentru majoritatea aplicațiilor web (inclusiv aplicațiile mobile), API-ul Gmail este cea mai bună alegere pentru acces autorizat la datele Gmail ale unui utilizator. API-ul Gmail oferă un acces flexibil, RESTful la căsuța de e-mail a utilizatorului, cu un natural interfață cu Fire, Mesaje, Etichete, Schițe și Istoric.

3.1 INSTALAREA ȘI NOTIFICAREA PRIN EMAIL, FOLOSIND GMAIL-API

Deschidem un browser(chrome,Microsoft Edge,Firefox) după care în bara de căutare scriem “console.cloud.google.com”. [18]

Această consolaă este o interfață web prin care utilizatorii pot gestiona și monitoriza resursele și serviciile Google Cloud Platform(GCP).Prin această consolă utilizatorii pot:

Crea și gestiona proiecte(organizează resursele în proiecte pentru o mai bună gestionare)

Avem access la servicii GCP cum ar fi Compute Engine,Cloud Storage BigQuery,Cloud Functions și multe altele(noi o să avem nevoie de GMAIL API pentru proiectul nostru)

Monitorizează resursele prin faptul că avem la dispoziție anumite instrumente pentru Cloud Monitoring și Cloud Logging pentru a urmări performanța și se pot depista problemele mult mai ușor dacă acestea apar.

Configurează Securitatea și accesul:Putem utiliza IAM(Identity and Access Management) cu ajutorul acestui IAM putem gestiona mai efficient rolurile și permisiunile utilizatorilor.

Odată intrați pe platformă o să fie necesar să ne logăm (dacă nu suntem logați deja)pentru a putea avea access pe platforma respectivă(vezi Fig a).

În dreapta sus o să fie necesar să selectăm proiectul așa cum ne este afișat în figura b)După ce am ales proiectul apăsăm pe “New project”(dacă nu avem deja un proiect cu această aplicație) pentru a crea un nou proiect așa cum ne este afișat în figură c. O să fie necesar să ne alegem numele proiectului noi am ales “LABD003”(cum este afișat în figura d),dar se poate alege oricare alt nume. La final apăsăm butonul de „Create” pentru a crea aplicația și respectăm pașii care sunt necesari.Folosirea acestor servicii precum „Google Api”, „Google Sheets”, sunt foarte importanți,de exemplu dacă folosim „Google Sheets API” am putea să scriem în excel automat persoanele care au fost detectate în timp real pe camera respectivă cu data și ora și să fie scrise într-un tabel, noi în momentul de față o să folosim un fișier .txt pentru a scrie acest lucru.

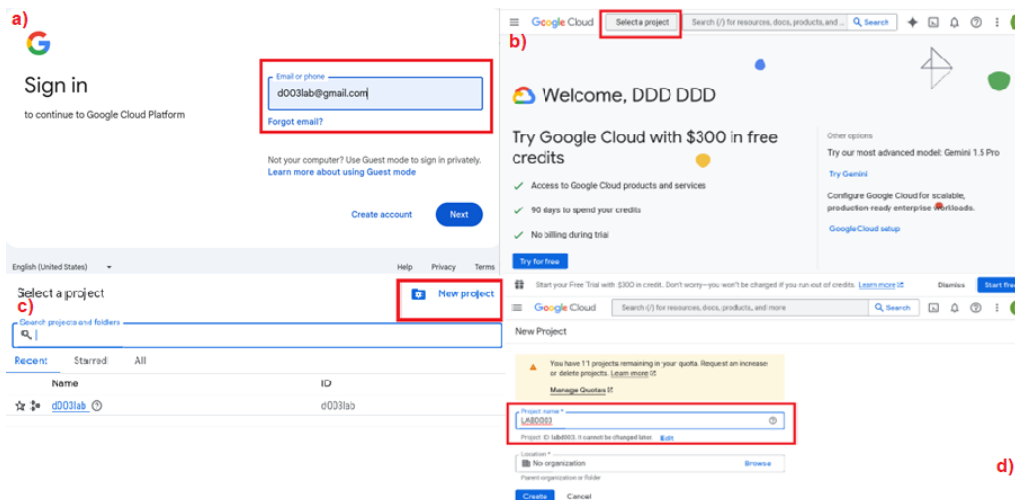


Fig14.Crearea proiectului pentru a folosi Google API

Următorul pas este să folosim un OAuth consent screen (ecranul de consimțământ OAuth) care este strâns legat de fluxul de autorizare OAuth 2.0. Acest proces permite unei aplicații să obțină access la datele unui utilizator de pe o platformă (fără a expune credențialele utilizatorului cum ar fi numele de utilizator și parola).

Funcționarea acestui OAuth este următorul: Utilizatorul acesează aplicația și începe un process care necesită access la datele sale, aplicația respectivă redirecționează utilizatorul către o platformă pentru a obține autorizarea. Utilizatorul vede acel ecran OAuth în care îi este arătat numele aplicației ce permisiuni solicită aplicația și un link către politica de confidențialitate și termenii și condițiile aplicației. Utilizatorul poate să refuze sau să acorde permisiunile. Dacă utilizatorul permite permisiunile atunci o să fie trimis un cod de autorizare (un token) de access către aplicație iar acest token este folosit pentru a face cereri API în numele utilizatorului, cu acest token aplicația accesează datele utilizatorului.

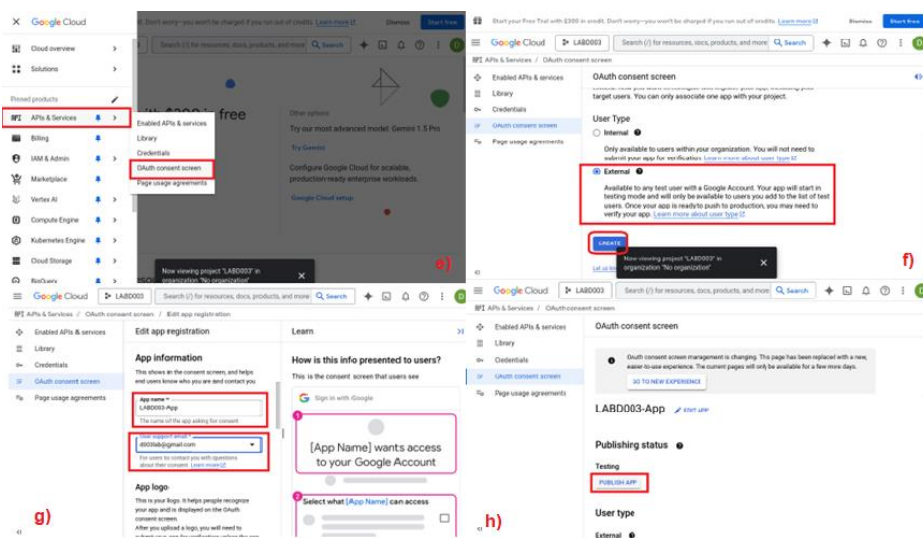


Fig15.Crearea OAuth consent screen

În Fig15. avem pașii pentru fiecare imagine pe care trebuie să îi facem mai departe pentru a crea un OAuth Client.

În fotografia „f)” am ales External deoarece aplicația este valabilă pentru teste și putem face mai multe verificări îmbunătățiri înainte de a fi lansată către producție,

Dacă alegeam internal aplicația era disponibilă doar pentru persoanele din interiorul organizației.

În fotografia f) la Developer contact information putem pune același email. După care apăsăm pe “Save and Continue” de vreo 2-3 ori și pe urmă “Back to Dashboard”

Ne reîntoarcem la „Api&Services” doar că de data asta în loc să accesăm „OAuth Content Screen” o să accesăm „Credentials”. Aici o să ne creăm Credeantial cu „CREATE CREDENTIALS”, iar acolo o să alegem „OAuth Client ID”. La „OAuth Client ID” tipul aplicației (Application Type) o să alegem Desktop App iar numele „D003LAB1”, iar la final o să dăm click pe „Create” pentru a ne crea „OAuth Client”. Dacă toți pașii au fost respectați ar trebui să ne apară acest mesaj.

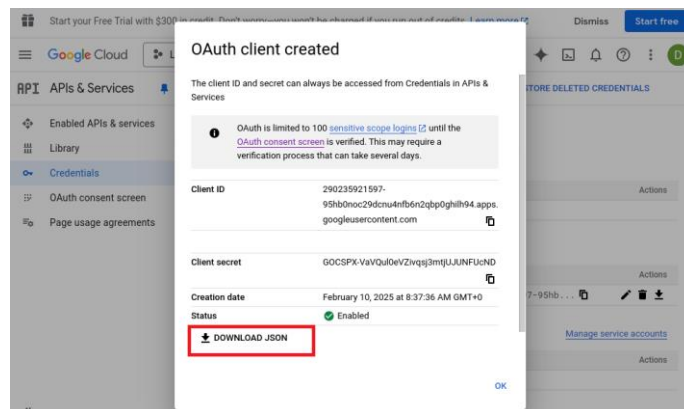


Fig16.Descarcarea fisierul JSON dupa crearea OAuth client

O să descarcăm acel JSON care o să conțină datele noastre ale contului pe care suntem în acest moment, cine are acces la aceste date practic are acces la contul nostru. Acel fișier .JSON o să-l redenumim în „D003LAB1.json”. Avem de făcut încă câțiva pași, după ce am redenumit acest fișier o să ne reîntoarcem la „Services&Api” și accesăm „Library” iar acolo o să avem o bară de search și scriem „GMAIL API” și îl activăm cu „Enable” același lucru facem și pentru „google drive API”.

Următorul pas este să facem rost de un token.json ca să facem acest lucru o să avem nevoie de un script pentru a face acest lucru.

Înainte de a executa comanda “python for_token.py” în care „for_token.py” este scriptul nostru, trebuie să folosim o comandă pentru a instala bibliotecile necesare “pip install --upgrade google-api-python-client google-auth-http2 google-auth-oauthlib”

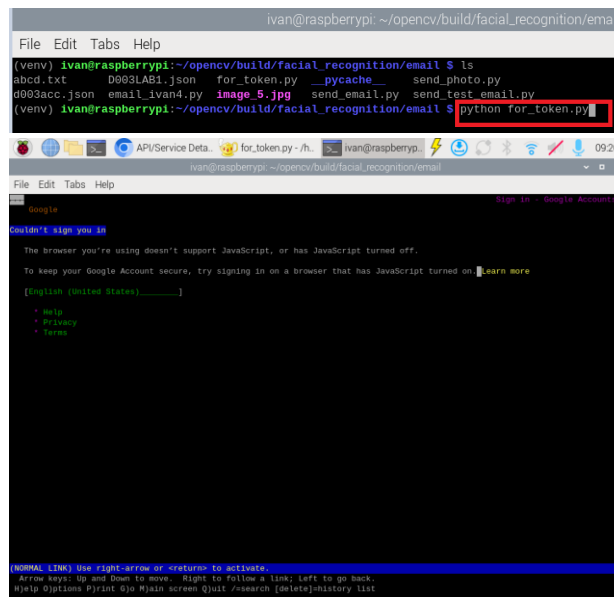


Fig17. Obținerea token.json pe Raspberry Pi

După cum putem observa, browserul nostru fie nu suportă JavaScript, fie este setat pe off. În acest caz, am folosit o altă variantă și anume: am rulat `for_token.py` pe un alt dispozitiv (laptop) pentru a obține fișierul `token.json`. Pentru a putea rula scriptul pe alt dispozitiv, vom avea nevoie ca fișierul `D003LAB1.json` să fie mutat pe acel dispozitiv și să se afle în calea corectă pentru a putea fi accesat. La variabila `CLIENT_FILE` am specificat fișierul pe care l-am descărcat și l-am redenumit în `D003LAB1.json`.

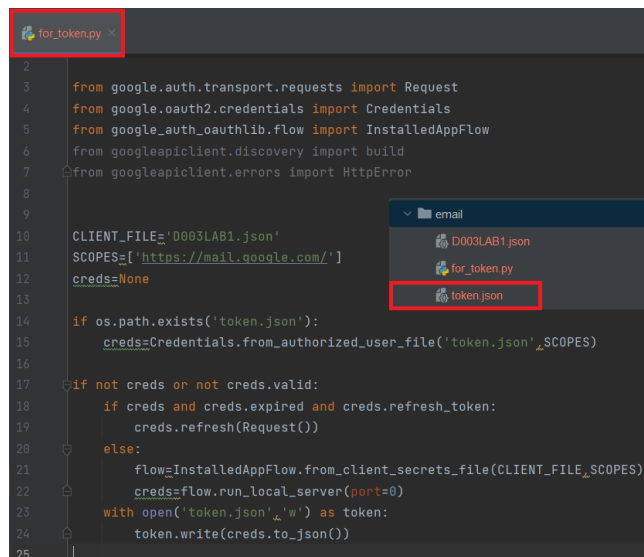


Fig18. Obținerea token.json pe alt device(windows,pycharm)

Când o să rulăm acest script o să fie necesar să ne autentificăm, după ce ne logăm pe cont, tokenul respectiv este salvat în `token.json` astfel încât utilizatorul să nu fie nevoit să se autentifice manual de fiecare dată, iar dacă acel token expiră se va reîmprospăta automat fără intervenția utilizatorului. În acest fel noi o să putem trimite email-uri automat dintr-un script de python.

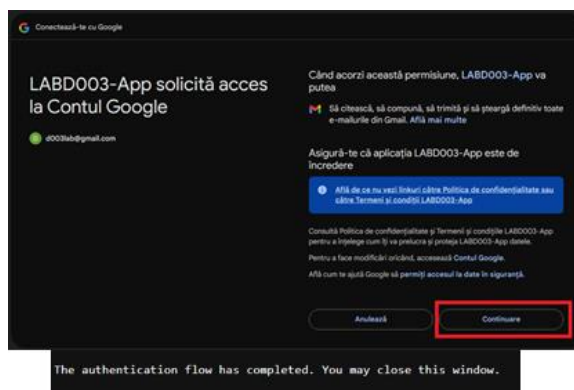


Fig19. Aplicația creată de noi solicită acces la contul nostru google

Acel token.json trebuie să îl mutăm înapoi pe celălalt dispozitiv, deoarece doar datorită acestui token putem să rămânem conectați (ne este permis să trimitem email). Acel token îl mutăm în folderul unde este și scriptul creat pentru trimiterea mesajelor când o persoană este necunoscută

4 NODE-RED DASHBOARD

Node-red este un instrument de programare care este bazat pe fluxuri, a fost dezvoltat inițial de IBM și acum este open-source. Se utilizează în general pentru conectarea dispozitivelor hardware, API-URI și servicii online și este foarte potrivit pentru aplicații IoT (Internet of Things). Node-red oferă un editor vizual bazat pe browser, care permite utilizatorilor să creeze fluxuri de date prin tragerea și plasarea nodurilor, simplificând implementarea funcționalităților complexe. Principalele caracteristici ale node-red sunt bazate pe programare vizuală, este ușor și eficient, modular, integrări ușoare

Prin programare vizuală se pot construi fluxuri de date prin tragerea și conectarea nodurilor, fără a scrie foarte mult cod, acest lucru este mult mai ușor de folosit și înțeles fără a avea la bază mult cod

Este ușor și eficient deoarece este bazat pe Node.js și consumă puține resurse fiind mult mai potrivit pentru dispozitivele embedded (Raspberry Pi)

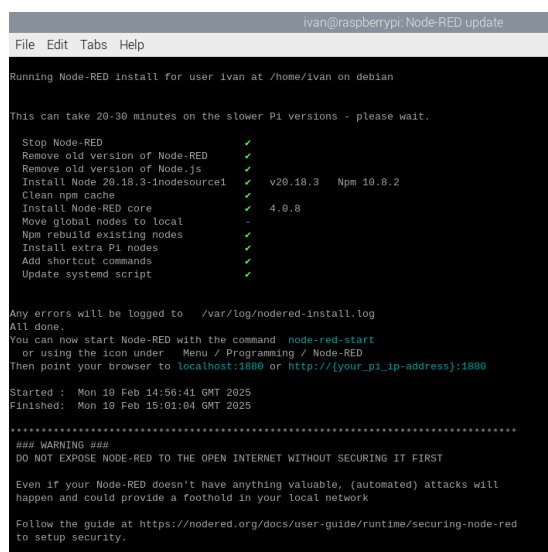
Este modular deoarece se pot extinde funcționalitățile prin instalarea de pachete suplimentare (node modules)

Când spunem integrări ușoare ne referim la faptul că suportă o gamă largă de protocoale și servicii, cum ar fi MQTT, HTTP, WebSockets și multe altele.

4.1 Instalare Node-Red pe Raspberry Pi

Folosim comanda:

```
“ bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-  
installers/master/deb/update-nodejs-and-nodered) “
```



```
ivan@raspberrypi: Node-RED update
File Edit Tabs Help

Running Node-RED install for user ivan at /home/ivan on debian

This can take 20-30 minutes on the slower Pi versions - please wait.

Stop Node-RED ✓
Remove old version of Node-RED ✓
Remove old version of Node.js ✓
Install Node 20.18.3-inodesource1 ✓ v20.18.3 Npm 10.8.2
Clean npm cache ✓
Install Node-RED core ✓ 4.0.8
Move global nodes to local -
Npm rebuild existing nodes ✓
Install extra Pi nodes ✓
Add shortcut commands ✓
Update systemd script ✓

Any errors will be logged to /var/log/nodered-install.log
All done.
You can now start Node-RED with the command node-red-start
or using the icon under Menu / Programming / Node-RED
Then point your browser to localhost:1880 or http://(your_pi_ip_address):1880

Started : Mon 10 Feb 14:56:41 GMT 2025
Finished: Mon 10 Feb 15:01:04 GMT 2025

#####
### WARNING ###
DO NOT EXPOSE NODE-RED TO THE OPEN INTERNET WITHOUT SECURING IT FIRST

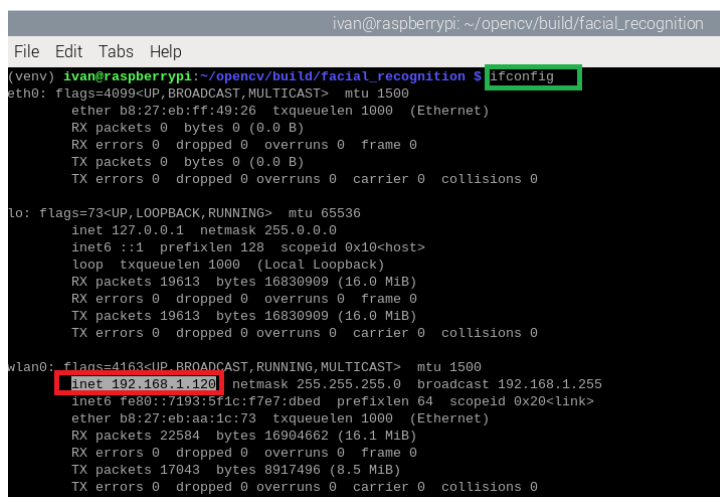
Even if your Node-RED doesn't have anything valuable, (automated) attacks will
happen and could provide a foothold in your local network

Follow the guide at https://nodered.org/docs/user-guide/runtime/securing-node-red
to setup security.
```

Fig20. Instalarea a avut succes

4.2 CONECTARE PRIN SSH LA RPI

Înainte de toate trebuie să vedem ip-ul pentru a putea să ne conectăm prin ssh la RPI. Deschidem terminalul și scriem ifconfig, cu această comandă ne sunt afișate ip -urile.



```
ivan@raspberrypi: ~/opencv/build/facial_recognition
File Edit Tabs Help
(venv) ivan@raspberrypi:~/opencv/build/facial_recognition $ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether b8:27:eb:ff:49:26 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 19613 bytes 16830909 (16.0 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 19613 bytes 16830909 (16.0 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.120 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::7193:5f1c:f7e7:dbed prefixlen 64 scopeid 0x20<link>
    ether b8:27:eb:aa:1c:73 txqueuelen 1000 (Ethernet)
    RX packets 22584 bytes 16904662 (16.1 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 17043 bytes 8917496 (8.5 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Fig21. Raspberry Pi - Adresă IP locală (192.168.1.120) afișată cu ifconfig

Înainte de a încerca să ne conectăm prin ssh este necesar să avem activată conexiunea prin ssh pentru a putea să ne conectăm pe RPI.

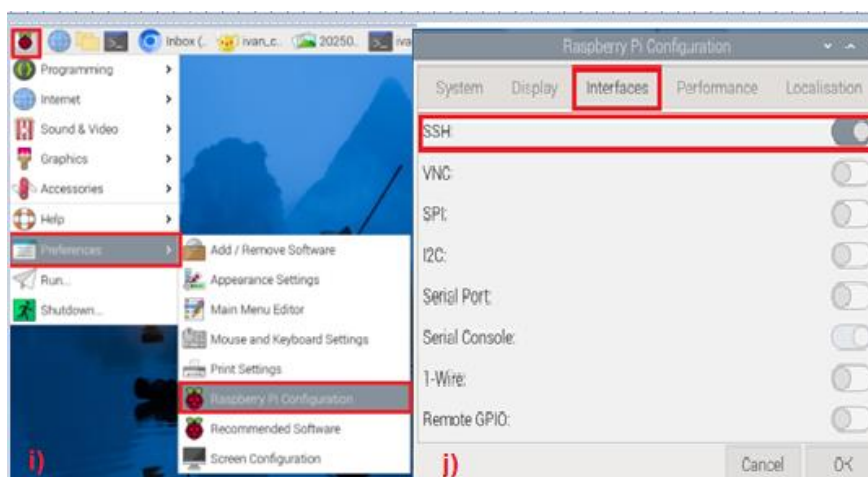
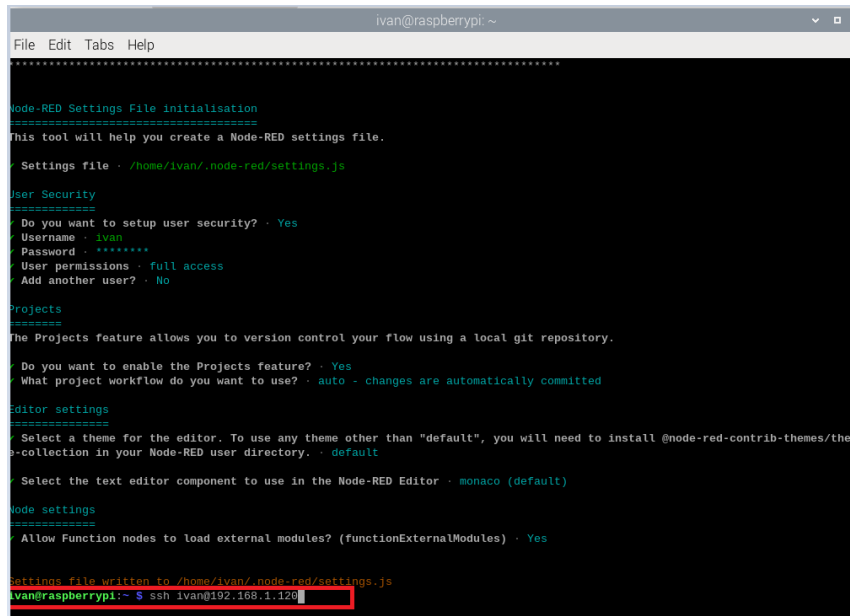


Fig22. Activare SSH pe Raspberry Pi



```

ivan@raspberrypi: ~
File Edit Tabs Help

Node-RED Settings File initialisation
=====
This tool will help you create a Node-RED settings file.

Settings file · /home/ivan/.node-red/settings.js

User Security
=====
Do you want to setup user security? · Yes
Username · ivan
Password · *****
User permissions · full access
Add another user? · No

Projects
=====
The Projects feature allows you to version control your flow using a local git repository.
Do you want to enable the Projects feature? · Yes
What project workflow do you want to use? · auto - changes are automatically committed

Editor settings
=====
Select a theme for the editor. To use any theme other than "default", you will need to install @node-red-contrib-themes/themes-collection in your Node-RED user directory. · default
Select the text editor component to use in the Node-RED Editor · monaco (default)

Node settings
=====
Allow Function nodes to load external modules? (functionExternalModules) · Yes

Settings file written to /home/ivan/.node-red/settings.js
ivan@raspberrypi:~$ ssh ivan@192.168.1.120
  
```

Fig23.Conectare la Raspberry Pi prin SSH

Nu este obligatoriu să ne conectăm prin ssh, dar dacă dorim să avem și control de la distanță, putem face acest lucru (pe windows putem folosi PuTTY)

Folosind Putty care este un client gratuit pentru Telnet, SSH și Rlogin. Aceste protocoale ne permit o comunicare la distanță cu serverele. Unele dintre acestea au și protocoale de criptare cât și port forwarding și gestionarea sesiunilor.

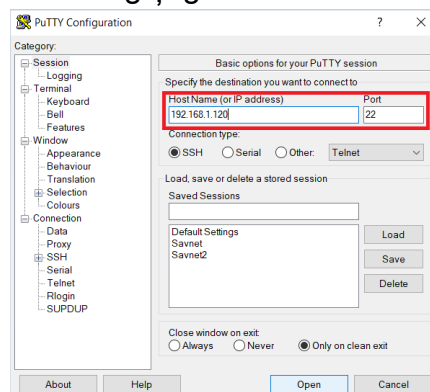
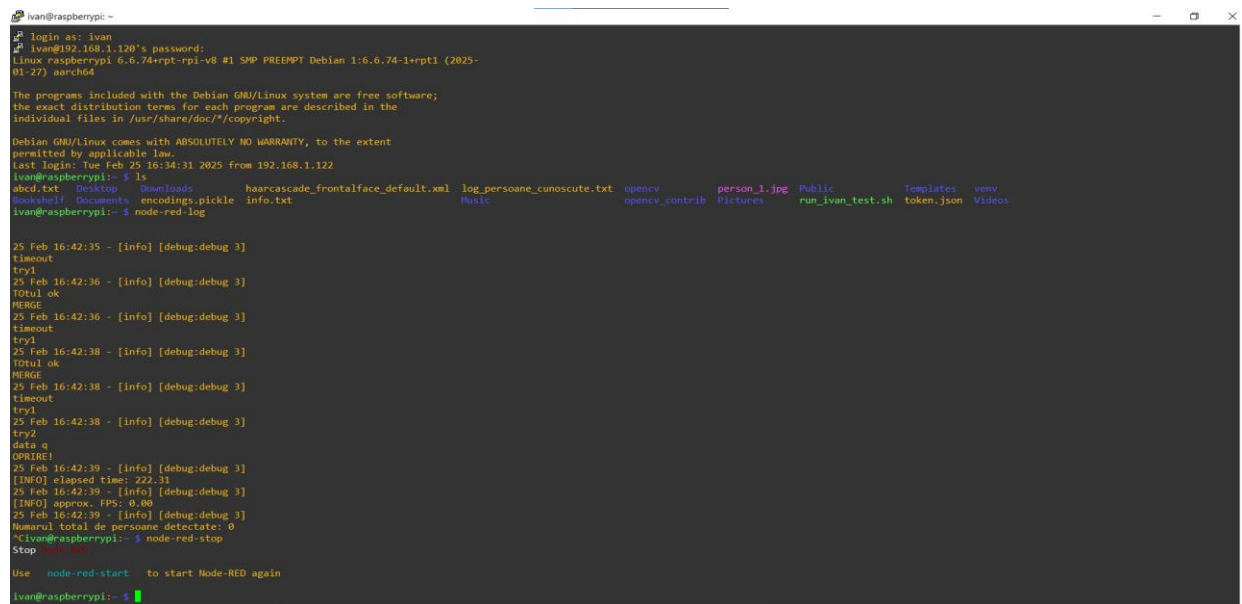


Fig24.Conectare prin SSH folosind Putty(windows)

Dacă dorim să vedem doar dashboardul și ui (user interface) în node red. Deschidem un browser (firefox, chrome, etc) și scriem 192.168.1.120:1880 acesta o să ne deschidă node-red și o să fie necesar să ne conectăm pentru a putea avea access în mediul de dezvoltare, iar pentru user interface putem scrie 192.168.1.120:1880/ui acest lucru o să ne deschidă o fereastră pentru a vedea aplicația la care are access utilizatorul (acest lucru funcționează doar dacă suntem în aceeași rețea).



```
ivan@raspberrypi:~$ ssh ivan
# login as: ivan
# ivan@192.168.1.120's password:
Linux raspberrypi 6.6.74-rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.6.74-1-rpt1 (2025-01-27) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Feb 25 16:34:31 2025 from 192.168.1.122
ivan@raspberrypi:~$ ls
abcd.txt  bookshelf  Downloads  haarcascade_frontalface_default.xml  log_persoane_cunoscut.txt  opencv  person_1.jpg  Public  Templates  venv
ivan@raspberrypi:~$ node-red-log
25 Feb 16:42:35 - [info] [debug:debug 3]
timeout
try1
25 Feb 16:42:36 - [info] [debug:debug 3]
TOTUL ok
HERGE
25 Feb 16:42:36 - [info] [debug:debug 3]
timeout
try1
25 Feb 16:42:38 - [info] [debug:debug 3]
TOTUL ok
HERGE
25 Feb 16:42:38 - [info] [debug:debug 3]
timeout
try1
25 Feb 16:42:38 - [info] [debug:debug 3]
try2
data q
OPRIRE!
25 Feb 16:42:39 - [info] [debug:debug 3]
[INFO] elapsed time: 222.31
25 Feb 16:42:39 - [info] [debug:debug 3]
[INFO] approx. FPS: 0.00
25 Feb 16:42:39 - [info] [debug:debug 3]
Numarul total de persoane detectate: 0
ivan@raspberrypi:~$ node-red-stop
Stop node-red
ivan@raspberrypi:~$ node-red-start to start Node-RED again
ivan@raspberrypi:~$
```

Fig25. SSH Terminal – Conectare la Raspberry Pi și oprire Node-RED

Prin ssh putem avea access la fișiere la rularea scripturilor și desigur la depanarea problemelor în caz că o să fie nevoie de asistență de la distanță dar pentru a putea face acest lucru este necesar ca să avem port forwarding configurat.

4.3 FIREWALL

Un firewall este un sistem de Securitate al rețelei care controlează și monitorizează traficul de rețea atât intern cât și extern pe baza unor reguli de Securitate. Această crează un paravan/barieră de protecție între o rețea internă de încredere și o rețea externă rău dăunătoare. Implementarea acestui firewall poate fi atât software cât și hardware sau o combinație între cele două. Există mai multe tipuri de firewall-uri cu inspecție bazată pe starea conexiunilor, proxy, filtrare de pachete.

Firewall-ul pentru filtrarea pachetelor analizează fiecare pachet de date care trece prin ele și apoi le filtrează pe baza unor parametri precum adresă ip sursă și destinație numere de porturi și tipul de protocol. Aceste firewall-uri sunt simple și rentabile dar ele nu pot examina conținutul pachetelor ceea ce le face mai puțin eficiente împotriva atacurilor sofisticate. Avem și câteva comenzi pentru firewall:

`sudo ufw enable`-activează firewall-ul

“Firewall is active and enabled on system startup”

`sudo ufw disable`-dezactivează firewall-ul

Dacă activăm firewall-ul nu o să ne mai putem conecta de pe alt dispozitiv prin ssh numai dacă permitem ca pe portul 22(al ssh) să se poată face conexiunea cu comanda `sudo ufw allow 22/tcp`

Dacă dorim să blocăm porturile sau să revenim la setările implicite putem folosi “sudo ufw reset”, această comandă va șterge toate regulile și va dezactiva UFW și o să fie necesar să activăm din nou manual prin “sudo ufw enable”. O altă variantă ar fi să vedem regulile folosind “sudo ufw status numbered” această comandă ne va afișa toate regulile și pe ce porturi permit comunicarea. Dacă dorim să închidem un anumit port pur și simplu ștergem regula respectivă care aparține de portul respectiv folosind “sudo ufw delete” și scriem numărul acelei reguli pe care o dorim ștersă, iar acest lucru va închide portul respectiv, putem să vedem situația utilizând din nou sudo ufw status. (poza). Activarea acestui firewall și blocarea tuturor porturilor nu ne permite să facem o cerere către ip-ul 192.68.1.120 cu portul 1880 de pe alt dispozitiv chiar dacă suntem în aceeași rețea, deoarece odată activat firewall toate porturile sunt blocate, trebuie doar să folosim aceea comandă “sudo ufw allow” specificat portul și protocolul de comunicație așa cum am făcut anterior pentru portul 22. Deci dacă o să dorim să avem access la portul 1880 o să folosim “sudo ufw allow 1880/tcp” acel lucru ne permite să facem o cerere chiar dacă firewall-ul este activat.

4.4 SSH

Ssh(Secure Shell) este un protocol la nivel de rețea în care permite comunicarea între două sisteme și se asigură că datele trimise/primate sunt criptate în tot acest timp. Se utilizează în special pentru conectarea la servere/dispozitive la distanță, la configurarea și administrarea dispozitivelor de rețea și la transferul de fișiere. Folosim în general ssh deoarece toate datele sunt criptate prevenind astfel interceptarea datelor, utilizatorii se pot autentifica cu parole sau chei SSH și se asigură că toate datele nu sunt modificate în timpul transferului. SSH utilizează în mod implicit portul 22.

4.5 SERVER NODE-RED

Pornire server node-red:

Comanda în terminal: node-red-start,

Opre server node-red:

Comanda în terminal: node-red-stop

Pentru a vedea ce se întâmplă(log)

Comanda în terminal: node-red-log

Odată ce am pornit serverul node-red o să ne afișeze pe ce adresă URL o să trebuiască să intrăm, în cazul nostru pe 127.0.0.1:1880/ dacă dorim să vedem și partea de user interface scriem 127.0.0.1:1880/ui.

Înainte de toate o să avem nevoie să instalăm node-red-dashboard. Mergem la cele trei liniițe din colțul din dreapta sus pe urmă “Manage Pallette”, intrăm pe install și scriem

"node-red dashboard" și de acolo îl instalam pentru a putea folosi aceste butoane(pentru user interface) în proiectul nostru.(Vezi Fig.26)

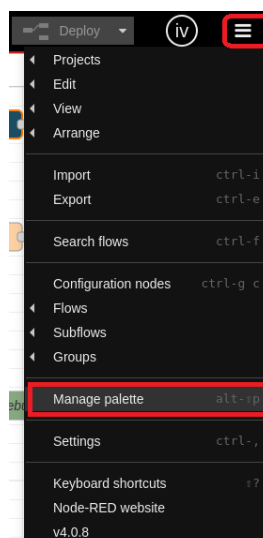


Fig26. Meniul principal node-red

În node-red am implementat un buton "START" care ne pornește aplicația creată pentru recunoașterea fețelor.



Fig27. Buton pornire aplicație pentru recunoașterea fețelor

Acest buton trimite un semnal către pythonshell iar „pythonshell in” este configurat cu o cale absolută pentru a porni fișierul nostru python(ivan_c7BUN.py) pentru facial recognition. Python shell este configurat și în așa fel încât să intre într-un mediu virtual(venv) și de acolo să acceseze fișierul (ivan_c7).

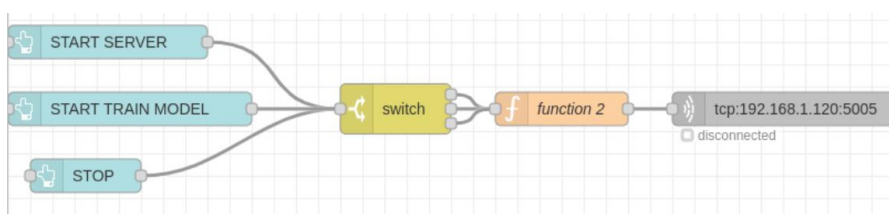


Fig28. Pornirea și oprirea atât a serverului TCP cât și pentru antrenarea modelului

În Fig28. avem 3 butoane, un buton pentru a porni server-ul un buton pentru a porni antrenarea modelelor și un buton pentru a opri server-ul. Odată ce server-ul este oprit se va opri orice aplicație(facial recognition sau pentru antrenarea modelelor).

La fel ca pentru primul buton de “START” avem și un buton “ care pornește aplicația pentru antrenarea modelelor.



Fig29.Pornirea aplicației pentru antrenarea modelui

La fel ca la primul buton,avem un pythonshell în care este specificată calea către fișierul nostru(train_model1.py) pentru a porni fișierul respectiv.

În nodul “switch” avem 3 reguli pentru fiecare comandă:

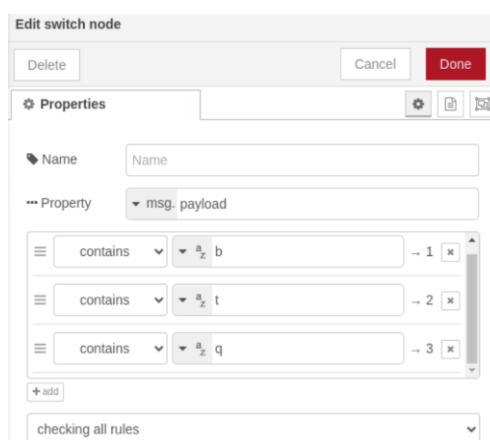


Fig30.Configurare nod switch

Prima comandă este pentru primul buton(START SERVER care conține în payload un string “b”)

A doua comanda este pentru al doilea buton(START TRAIN MODEL care conține în payload un string “t”)

A treia comandă este pentru al treilea buton(STOP care conține în payload un string “q”)

Un payload conține informația utilă sau esențială.Într-un pachet de date,un payload este conținutul real, un mesaj sau un fișier.

Într-un exploit,payload-ul este partea de cod sau instrucțiuni care execută efectiv acțiunea malicioasa sau dorită, după ce o vulnerabilitate a fost exploatată

În API-uri,un payload se referă la datele care sunt trimise sau primite între un client și server.Aceste date sunt de obicei JSON,XML sau form-data și conțin informațiile necesare pentru a efectua o anumită acțiune dorită.

În nodul “function 2” avem puse condiții pentru toate aceste stringuri:

```
if (msg.payload=== "b"){
    global.set('serverStatus',true);
    msg.payload='b';
    return msg;
}else if (msg.payload=== "q"){
    global.set('serverStatus',false)
    msg.payload="q";
    return msg;
}else if (msg.payload=== "t"){
    if (global.get('serverStatus')===true){
        return msg;
    }else{
        msg.payload="Serverul nu este pornit";
        node.error("Serverul nu este pornit",msg)
        return null;
    }
}
```

Fig31.Configurare nod „function 2”

În figura 3 este codul pentru “function 2”, în această funcție sunt verificate condițiile:

Dacă prin switch trece un payload care conține un string “b” atunci o să setăm variabila “statusServer” pe true” și trimitem payload ul mai departe.

Dacă pe switch trece un payload care conține un string “q” atunci setam variabila “statusServer” pe false.

Dacă prin payload trece un payload cu litera “t” și serverStatus este setat pe True atunci trimitem mesajul respectiv(stringul “t”) altfel o să primim un mesaj cum că serverul nu este pornit.

Pe scurt putem trimite acel mesaj “t” pentru a porni antrenamentul numai dacă variabila globală StatusServer este setată pe True, altfel nu se pot trimite mesaje deoarece server-ul nu este pornit.

Toate aceste mesaje se trimit prin TCP(Transmission Control Protocol) folosind nodul din node-red tcp request care este configurat în felul următor:

Fig32.Configurare nod tcp request

Adresa ip 192.168.1.120 este adresa ip a raspberry pi.

```
server=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
server.bind(("0.0.0.0",5005))
server.listen(1)

conn, addr=server.accept()
```

Fig33.Fragment de cod extras din fișierul pentru recunoașterea facială.

În fișierele (ivan_c7bun.py, train_model1.py) au fost scrise aceste linii de cod ca în figura ZX5.În care se creează un obiect de tip socket și este configurat astfel încât să folosească ipv4 (socket.AF_INET) și (socket.SOCK_STREAM) pentru conexiuni TCP care este un protocol de comunicare și se asigură că datele sunt trimise și primite în ordine fără erori.Linia de cod "server.bind(("0.0.0.0",5005))" înseamnă că ascultă pe toate interfețele și toate aceste interfețe sunt disponibile adică server-ul va accepta orice adresă ip a calculatorului.Portul "5005" este la portul pe care server-ul ascultă,putem considera acest port ca fiind o ușă prin care datele intra și ies.Următoarea linie "server.listen(1)" specifică numărul maxim de conexiuni.

Următoarea linie "conn,addr=server.accept()",această comandă o putem considera blocantă,în sensul că programul va aștepta până când cineva(client) se va conecta la server până atunci nu poate trece mai departe,print "conn" este un obiect de tip socket prin care putem primi și trimite date de la client,iar addr este un tuplu care conține portul și adresa ip a clientului.

Deci noi în momentul în care pornim aplicația(ivan_c7bun.py, train_model1.py) ambele aplicații înainte de toate așteaptă să se facă conexiunea deoarece avem aceea comandă blocantă care specifică clar că până nu este realizată o conexiune nu se poate merge mai departe cu rularea programului.Noi în momentul în care apăsăm butonul "START SERVER") trimitem un payload "b"(putem trimite orice altceva dacă edităm funcția(function 2),urmând ca acel semnal să treacă prin switch,function 2 și ajunge la nodul tcp request.În momentul acela noi cu nodul "tcp request" facem o cerere către serverul "192.168.1.120:5005".Dacă suntem în aceeași rețea putem de asemenea să facem același lucru(cu condiția ca aplicația să fie pornită),intrăm pe un browser și scriem "192.168.1.120" ,putem de pe orice alt dispozitiv să facem același lucru dar cu condiția să fim în aceeași rețea.Și așa am făcut o cerere către serverul respectiv,de asemenea putem și din python să facem același lucru atâta timp cât este în aceeași rețea(Fig33)

```
import requests
ip='192.168.1.120'
port=5005
url=f"http://{ip}:{port}"
response=requests.get(url)
```

Fig34. Cerere din python catre server

Dacă am folosi nodul “tcp out” în locul “tcp request” cu o configurație ca în figura Fig34. am avea foarte multe erori în log(node-red-log), multe erori de reconectare, deoarece acest nod odată ce pică conexiunea va încerca tot timpul să se reconecteze la un interval de 10 secunde(funcționează la fel ca routerele), toate aceste erori ar dispărea odată ce noi am trimite un semnal către acest nod red să se conecteze la server și este obligatoriu ca una dintre aceste aplicații să ruleze în acel moment deoarece cum am precizat în figura ZX5 programul nu poate funcționa dacă nu se realizează o conexiune. Odată ce conexiunea este realizată vor dispărea toate aceste erori care ne apar în log(vezi Fig.35)

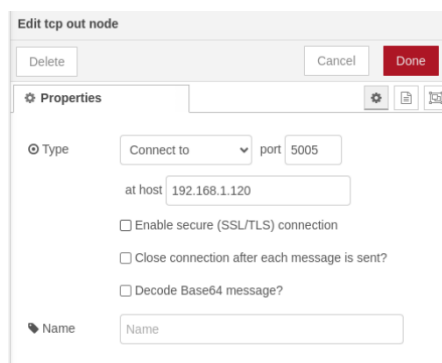


Fig35. Configurare nod TCP OUT

```
27 Feb 11:18:06 - [info] [tcp out:bdd122b08ead52e2] connecting to 192.168.1.120:5005
27 Feb 11:18:06 - [info] [tcp out:bdd122b08ead52e2] error: Error: connect ECONNREFUSED 192.168.1.120:5005
27 Feb 11:18:06 - [info] [tcp out:bdd122b08ead52e2] connection lost to 192.168.1.120:5005
27 Feb 11:18:16 - [info] [tcp out:bdd122b08ead52e2] connecting to 192.168.1.120:5005
27 Feb 11:18:16 - [info] [tcp out:bdd122b08ead52e2] error: Error: connect ECONNREFUSED 192.168.1.120:5005
27 Feb 11:18:16 - [info] [tcp out:bdd122b08ead52e2] connection lost to 192.168.1.120:5005
27 Feb 11:18:16 - [info] [debug:debug 1]
hello
27 Feb 11:18:26 - [info] [tcp out:bdd122b08ead52e2] connecting to 192.168.1.120:5005
27 Feb 11:18:26 - [info] [tcp out:bdd122b08ead52e2] error: Error: connect ECONNREFUSED 192.168.1.120:5005
27 Feb 11:18:26 - [info] [tcp out:bdd122b08ead52e2] connection lost to 192.168.1.120:5005
27 Feb 11:18:36 - [info] [tcp out:bdd122b08ead52e2] connecting to 192.168.1.120:5005
27 Feb 11:18:37 - [info] [tcp out:bdd122b08ead52e2] error: Error: connect ECONNREFUSED 192.168.1.120:5005
27 Feb 11:18:37 - [info] [tcp out:bdd122b08ead52e2] connection lost to 192.168.1.120:5005
27 Feb 11:18:43 - [info] [debug:debug 1]
Trebuie pornit serverul
27 Feb 11:18:47 - [info] [tcp out:bdd122b08ead52e2] connecting to 192.168.1.120:5005
27 Feb 11:18:47 - [info] [tcp out:bdd122b08ead52e2] connected to 192.168.1.120:5005
27 Feb 11:18:47 - [info] [debug:debug 1]
```

Fig36. Output obținut utilizând nodul TCP OUT

Acest lucru se întâmplă deoarece în cazul routerelor există un TTL(Time to live”) care tot timpul verifică la fiecare 10 secunde dacă un alt router este în viață,iar după 4 încercări nereușite el își va schimba ruta automat.

Creare folder:



Fig.37 Configurație în node-red pentru creare Folder

Pentru a crea un folder am folosit 3 noduri în node-red, primul nod este un nod de tip “text input”, în care utilizatorul va scrie numele folderului pe care dorește să îl creeze. Cel

de-al doilea nod este un nod de tip “fs mkdir”(file system) și care va utiliza comanda "mkdir" pe system pentru a creea respectivul folder în acest folder scriem și calea unde dorim să salvăm acel folder.Iar cel de-al treilea nod “Path” este de tip “text” în care ne va afișa pe ecran calea completă unde a fost creat folderul respectiv. Acest lucru funcționează în felul următor.Utilizatorul scrie numele care să îi fie atribuit acelui folder, acest mesaj(msg.payload) va fi trimis către nodul numărul doi care va apela comanda “mkdir” pe sistem la calea pe care am specificat-o noi, iar mesajul va fi trimis mai departe către nodul numărul 3 cu calea și numele ales de utilizator și ne va fi afișată calea completă și numele ales de utilizator unde a fost create respectivul folder și sub ce denumire.

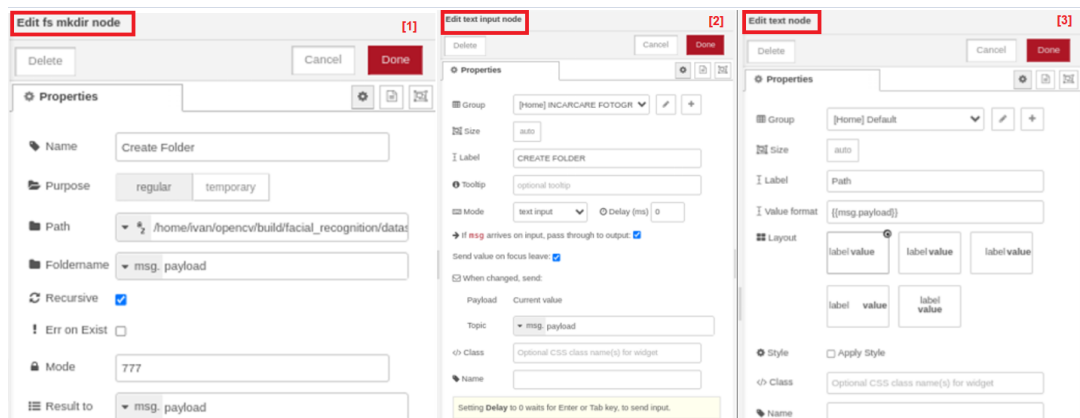


Fig38. Configurațiile nodurilor *fs mkdir*, *text input*,*text*

Alegere folder:



Fig39. Configurație node red pentru *Alegere folder*

Avem 3 noduri unul,primul “*CHOOSE FOLDER*” este de tip “*text input*” al doilea nod este unul de tip “*function*”,iar al treilea este unul de debug(nu este obligatoriu).În primul nod utilizatorul scrie folderul în care dorește să încarce fotografiile respective pentru a le putea antrena mai departe.Astfel spus,utilizatorul după ce a creat folderul(în sec. *CREARE FOLDER*), poate alege chiar folderul respectiv pentru a încarca fotografiile sau dacă dorește un alt folder care este deja,doar îl scrie și apasă tasta “Enter”.Al doilea nod preia textul/mesajul (ce a scris utilizatorul) în primul nod și creează o variabilă globală numită “*Variabila_Input*” ,care va fi folosită pentru a încarca mai multe fotografii la folderul ales de către utilizator,unde va dorii să antreneze aceste modele.

Incarcare fotografie:



Fig40.Configurație node red pentru *încărcare fotografie*

Pentru a încărca o fotografie o să folosim 5 noduri,upload,join,funcția “base 64”,funcția “function 3” și ultimul este un nod de tip “write file” ,pentru a scrie fișiere.

În primul nod după ce o fotografie este încărcată aceasta va trimite un mesaj(*msg.payload*) care conține toate datele acelei fotografii,toate acele date care aparțin de fotografia respectivă vor fi transferate sub formă binară către nodul “join”.Nodul join va combina fiecare *msg.payload* primit și va crea un Buffer cu acele mesaje.În funcția “base64” transformăm conținutul mesajului (care poate să fie sub formă de text,date binare,numere) într-un format mai special (*base64*).În această funcție mesajele pe care le primim o să transforme acele date binare sub formă de string și sunt encodeate în base64.În “function 3” transformăm datelele respective înapoi în formatul original urmând a fi trimise mai departe către nodul 5 care cu ajutorul acelor date pe care le-a primit să scrie imaginea sub format .jpg.Această metodă nu este foarte bună deoarece avem o singură imagine care se salvează sub aceeași formă “*povv.jpg*” iar imaginea respectivă va fi suprascrisa mereu dacă o să încărcăm altă fotografie [19].Acest node “upload” nu a fost pus în dashboard pentru că aceea fotografie s-ar fi suprascris mereu și nu putea să încarce mai multe fotografii odată,dar a fost un început bun pentru a vedea cum stau lucrurile.

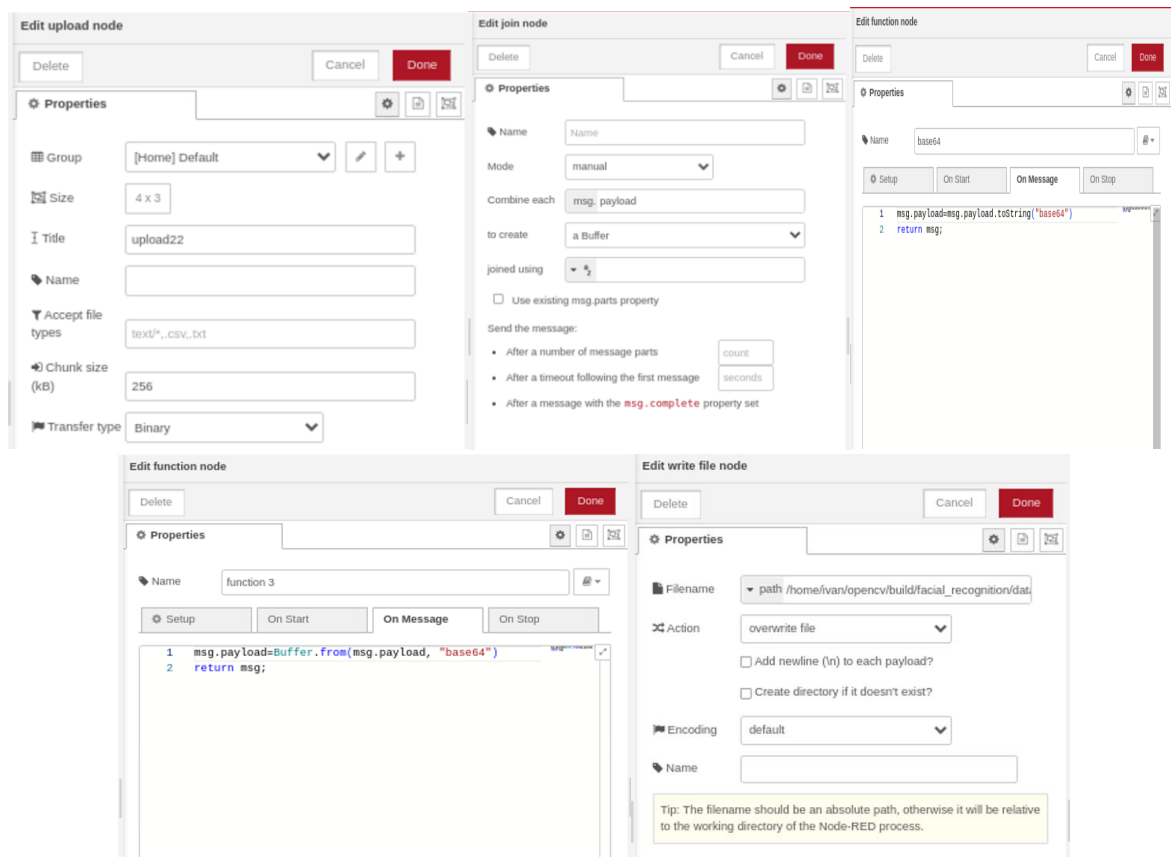


Fig41.Configurațiile nodurilor *upload,join,base64,function 3,write file*

4.6 MULTI-UPLOAD

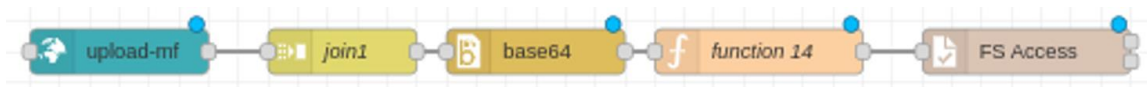


Fig42. Configurație node red pentru *MULTI-UPLOAD*

În această secțiune avem 5 noduri (upload-mf,join1,base64,function 14, FS Access).

În primul nod se încarcă fotografiile toate acele fotografii vor fi trimise sub formă binară către join1. Nodul al doilea are rolul de a combina fiecare mesaj primit cu datele respective și a crea un Buffer cu acele date urmând ca mai departe datele care sunt în Bufferul respectiv să fie trimise către nodul al treilea (base 64) care are rolul de a converti toate acele date din buffer-ul respectiv în base64. În nodul nr 4(function 14) am creat un script care să ne permită să extragem "Variabila_Input", variabilă care este creată de către utilizator. Dacă utilizatorul nu o să selecteze o variabilă valabilă (denumirea unui folder în care se dorește trimiterea acelor fotografii) atunci v-a da eroare. În cazul în care utilizatorul alege un folder valid în care dorește ca fotografiile să fie trimise atunci fotografiile vor fi primite sub formă de mesaj(msg.payload) cu datele care au fost convertite în base64 (nodul nr 3), și vor fi salvate într-o variabilă numită "base64Data", tot în această funcție o să generăm un nume unic pentru fiecare fotografie folosind timestamp și un număr aleatoriu, și desigur o să alegem și o cale în care dorim ca fotografiile să fie trimise. În final o să convertim fotografia din base64 în formatul original și trimitem un (msg.payload) către nodul numărul "FS Access". În care îi este specificată calea completă (full path) și mai trimitem un (msg.filename) care este scrierea fișierului (fotografia) în care a fost salvată, deoarece fotografia a fost salvată într-un buffer (function 14) urmând ca mai departe să fie trimisă sub formă de msg.filename către "FS Access".

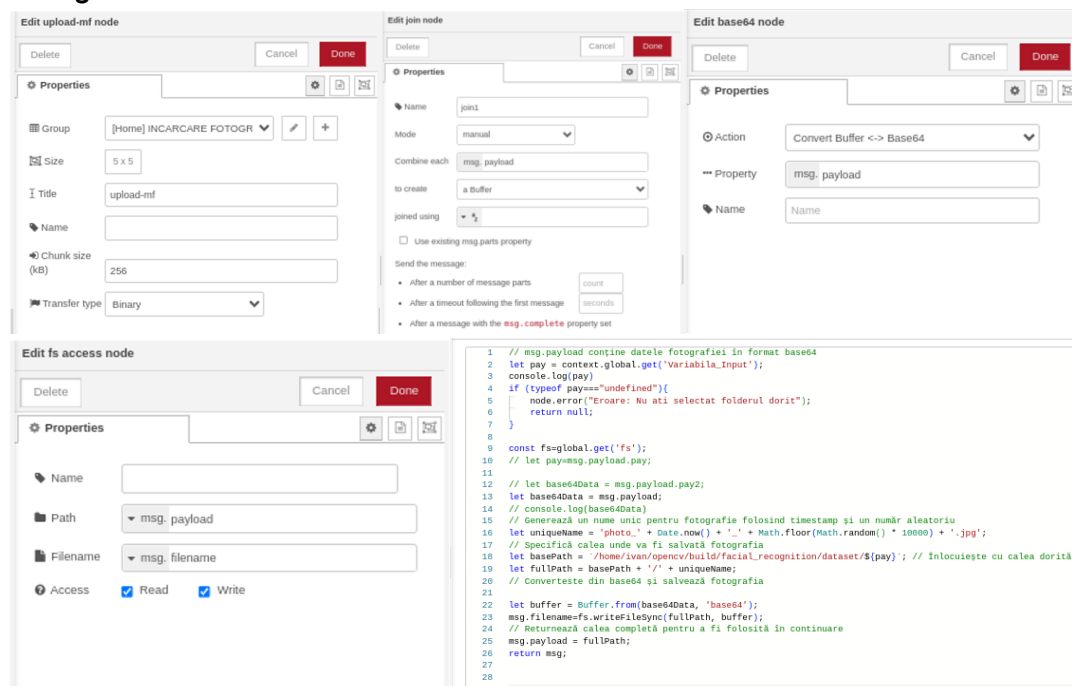


Fig43. Configurație pentru nodurile *upload-mf,join,base64,fs acces, function 14*

A screenshot of a web browser interface. The address bar shows the URL 'photo_1741186435129_2015.jpg'. The navigation bar includes buttons for 'Image', 'Edit', 'View', 'Go', and 'Help'. Below this is a toolbar with icons for back, forward, home, and search, along with a search input field. A red error banner is displayed at the bottom of the browser window, containing a red circle with a white minus sign, the text 'Could not load image 'photo_1741186435129_2015.jpg'.', and the message 'Error interpreting JPEG image file (Not a JPEG file: starts with 0x2f 0x68)'. A 'Retry' button is located on the right side of the error banner.

The screenshot displays two parts of the application. On the left, a code editor shows JavaScript code for handling a message payload. The code defines a function to save a file by generating a unique filename based on the current date and time, converting the base64-encoded payload to a buffer, and writing it to a directory named 'build/facial_recognition'. On the right, a 'Edit write file node' dialog box is shown. It has fields for 'Filename' (set to 'msg.filename') and 'Action' (set to 'overwrite file'). There are also checkboxes for adding newlines and creating directories if they don't exist. A tip at the bottom states: 'Tip: The filename should be an absolute path, otherwise it will be relative to the working directory of the Node-RED process.'

The diagram shows a workflow with the following steps:

- Afisare folder** (light blue box) connects to **FS List** (brown box).
- Afisare fisiere** (light blue box) connects to **FS List** (brown box).
- Two **FS List** boxes connect to a central **list** box (yellow box).
- The **list** box connects to **function 20** (orange box).
- function 20** connects to **storage folders** (orange box).

- 38 -

În care utilizatorul se răzgândește să treacă din enable în disable acesta va primi din nou un mesaj, și o să fie 2 căi identice, doar că unul este enable și altul este disable, dar fiind faptul că sunt 2 căi identice se consideră duplicat și ambele căi o să fie șterse din array-ul respectiv. Acest lucru se întâmplă deoarece dacă un utilizator dorește să șteargă un folder și îl pune pe enable și pe urmă se răzgândește și îl pune înapoi pe disable, practic calea respectivă cu folderul respectiv este „duplicate” deoarece sunt duplicate, doar că un mesaj este trimis cu “True” în momentul în care bifează acel folder pe enable și dacă utilizatorul se răzgândește și îl pune din True pe False va trimite un alt mesaj cu calea folderului respective pe False, acestea fiind spuse o să fie 2 foldere identice în acel array, aceste este motivul pentru care ștergem toate căile care sunt duplicate. Acest array va fi salvat într o variabilă globală numită “*folders*”. În Fig45 avem configurația pentru Afișare folder fs list, list iar pentru funcțiile cele 2 „function 20, storage folder” se găsesc în ANEXĂ

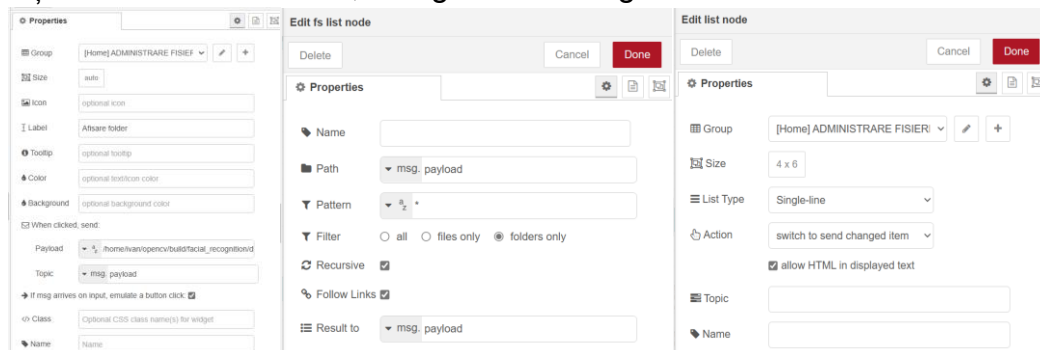


Fig47. Configurația nodurilor pentru Afișare folder fs list, list

4.8 ȘTERGERE FOLDER



Fig48. Configurația în node red pentru ȘTERGERE FOLDER

Avem 3 noduri (ȘTERGERE FOLDER, FUNCTION 21, EXEC),

Primul nod trimite un payload cu “DELETE”, iar acest payload o să fie folosit în funcție ca o condiție în momentul în care este apăsat. În această funcție (function 21), pentru început o să extragem aceea variabilă globală (acel array cu căile respective), aceea variabilă globală se numește “*folders*” și o să conțină un array cu toate căile respective pentru fiecare folder care a fost enable de către utilizator). În momentul în care se apasă butonul de ștergere se trimite acel payload de “delete” și noi avem o condiție în cazul în care se întâmplă acest lucru în funcția respectivă să se șteargă toate căile respective din acel array (în principiu golește întreg array ul) și îl salvează în aceeași variabilă globală “*folders*” doar că de data aceasta acel array va fi gol, urmând ca mai departe să se trimită un payload către msg.payload către modul exec care va executa o comandă de ștergere a tuturor folderelor respective care au fost în acel array. Înainte de a șterge complet acel array am salvat acel array în (msg.payload) care în același timp va trimite și o comandă de ștergere a acelor foldere (o să se vadă în poză la ce fac referire). Când acel mesaj msg.payload ajunge la nodul exec se execută comandă de ștergere a acelor foldere, deoarece în acel msg.payload

au fost toate căile ale folderelor necesare pentru a fi șterse care au fost pe modul enable de către utilizator.

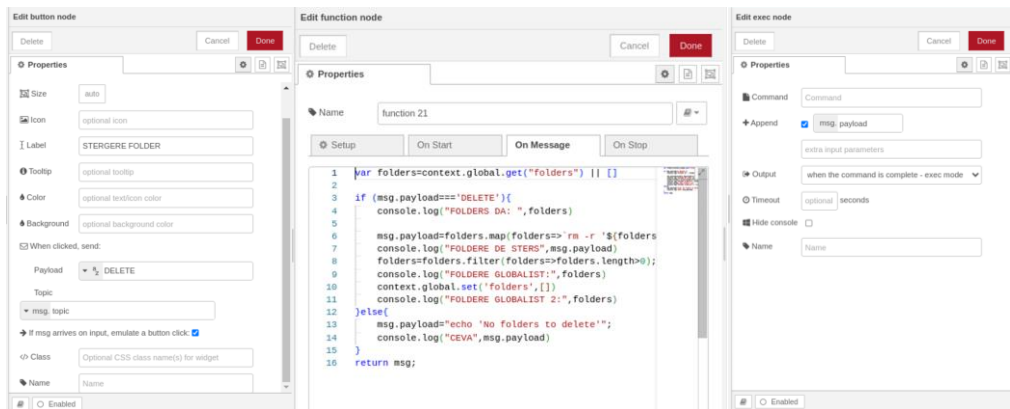


Fig49. Configurația nodurilor pentru ȘTERGERE FOLDER, FUNCTION 21, EXEC

4.9 INTRĂRI/IEȘIRI

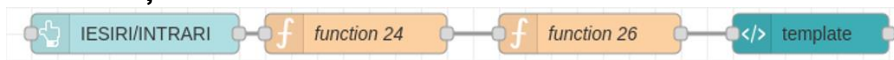


Fig50. Configurația în node red pentru INTRĂRI/IEȘIRI

Avem 4 noduri (IEȘIRI/INTRĂRI, function 24, function 26, template). Primul nod trimite un semnal către nodul nr 2 în momentul în care este apăsat. În funcția 24 o să se citească un fișier cu persoanele cunoscute cât și cele necunoscute cu data și ora la care au intrat sau au ieșit dintr-o încăpăre, va citi fiecare linie din fișier și va trimite atât un payload cât și un topic. Acel payload va conține linia la care a citit din acel fișier și topicul va conține "new_line". Atât payload-ul cât și topicul o să fie trimise către funcția 26, tot în această funcție o să verificăm dacă au schimbări în fișier, în momentul în care există schimbări în fișier o să citească linia la care a avut loc schimbarea și o va trimite mai departe (va fi apelată funcția (processNewLine) din nou). Funcția 26 nu face nimic altceva decât că primește payload-ul și topicul și îl trimite către template. În template se verifică dacă acel topic este de tip string și dacă este "new_line", dacă acel topic este sub această denumire atunci se intra în acel "if" și se crează un nou element (o nouă linie) unde va fi plasat acel payload (care conține data și ora la care a intrat o persoană în încăpăre). Această funcție de tip scope tot timpul va aștepta mesaje și va actualiza orice mesaj care va veni din fișierul respectiv astfel încât să vedem în dashboard data și ora la care persoanele au intrat sau au ieșit din încăpăre.



Fig51. Configurația nodurilor pentru IEȘIRI/INTRĂRI, function 24, function 26, template.


```

1
2 <div style="font-family: Arial, sans-serif; padding:10px;">
3   <h3>Continut fisier:</h3>
4   <ul id="new-line"></ul>
5 </div>
6
7 <script>
8   (function(scope){
9     scope.$watch('msg',function(msg)
10    {
11      if(msg && msg.topic=="new_line"){
12        let ul=document.getElementById("new-line");
13        let li=document.createElement("li");
14        li.textContent=msg.payload;
15        ul.appendChild(li);
16      }
17    })
18  })(scope);
19
20 </script>
21
22
23
24

```

Fig52.Cod pentru afișarea mesajelor în timp real într-o listă HTML

Afișarea la nivel local a unui flux video și folosirea Flask pentru transmiterea fluxului de date către node-red.

Avem doua scripturi în care folosim o funcție "cv2.imshow()" din OpenCV care ne afișează fluxul video la nivel local, iar în al doilea script folosim un server Flask pentru a transmite acel flux către interfață Node-Red Dashboard astfel încât dacă cineva dorește să vadă acel flux video de pe un telefon sau un alt laptop să poată să aibă acces la acel flux video.

Pentru primul script este foarte simplu,avem funcția respectivă care ne deschide o fereastră pentru a vizualiza fluxul video.Pentru cel de-al doilea script se creează serverul.se pornește serverul, iar întregul flux video este transmis pe o ruta (/video_feed),iar noi folosim în dashboard folosim un template și în template introducem sursa rutei respective,astfel încât ceea ce se transmite pe /video_feed să se transmită și în Node-Red Dashboard.

Edit template node > Text editor

```

1 

```

Fig53.Exemplu de integrare a sursei video în Node-RED

Vizualizare Dashboard:

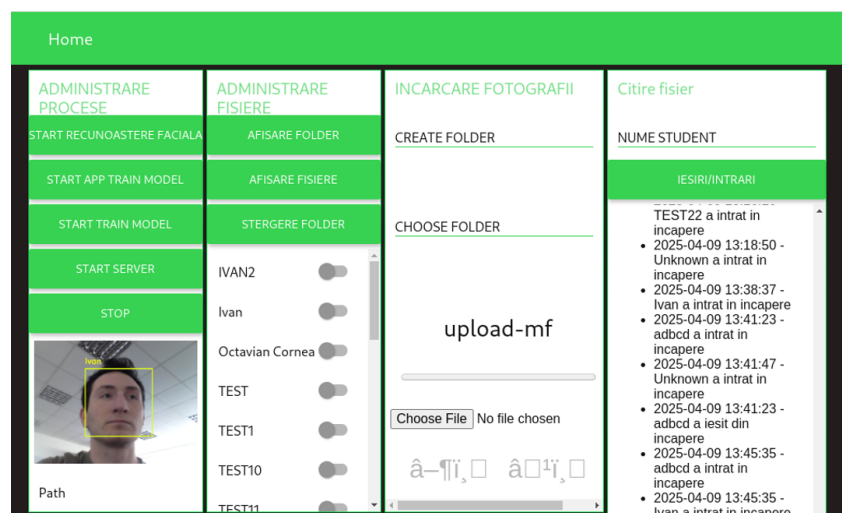


Fig54.Interfață sistemului pentru recunoaștere facială

Secure node-red dashboard:

Intrăm în terminal și scriem `"cd home/ivan/.node-red/"`, intrăm în `settings.js` cu `"sudo nano settings.js"` pentru a putea modifica fișierul, tastăm `"ctrl+w"` pentru a căuta `"httpNodeAuth"`, odată ajunși acolo o să comentăm această linie, în loc de `"user"` alegem userul nostru iar la `"pass"` o să trebuiască să generăm noi o parolă. Deschidem un alt terminal și scriem `"node-red admin hash-pw"`, o să alegem o parolă dorită și o să primim un hash al parolei respective, practic se generează un hash pentru parola aleasă de noi. O să copiem acest hash și îl punem în dreptul lui `"pass"`. Apăsăm `"Ctrl+x"` și `y` pentru a salva modificările făcute, după care apăsăm tasta `"enter"`. După ce am făcut acest lucru o să dăm un restart la server-ul node-red (scriem în terminal `"node-red"`) sau scriem (`"node-red-stop"`, `"node-red-start"`). După care putem intra pe `"192.168.1.120:1880/ui"` pentru a vedea dacă acest lucru a avut effect.

4.10 ÎNREGISTRAREA STUDENȚILOR

Dacă dorim să înregistrăm studenții la nivel local în loc să încarcăm fotografiile necesare pentru antrenarea modelelor putem face într-un mod mai simplu astfel încât fiecare student să își scrie numelele și să își facă câteva fotografii.

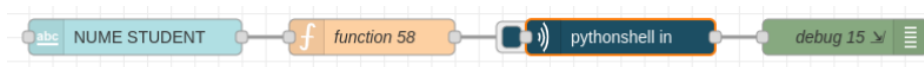
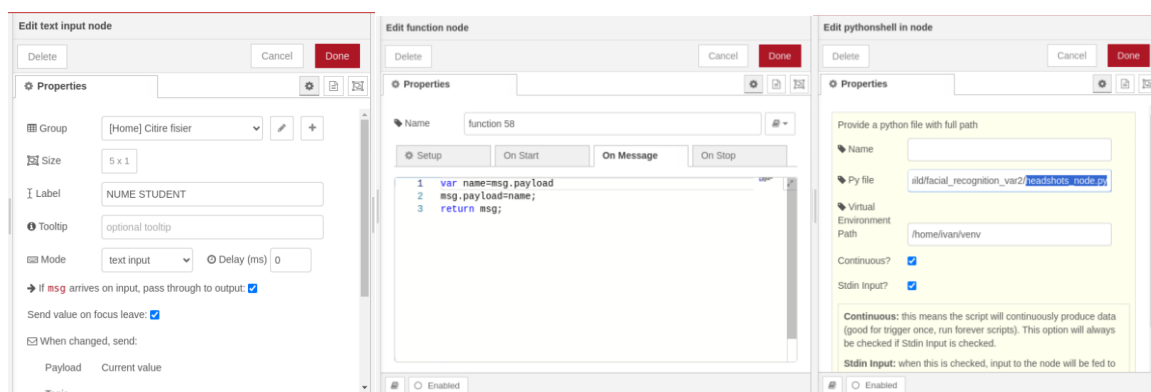


Fig55. Configurația în node red pentru *NUME STUDENT*

Avem 4 noduri („*NUME STUDENT*”, „*function 58*”, „*pythonshell in*”, „*debug 15*”)

Utilizatorul (*studentul*) își scrie propriul nume variabila numelui respectiv se va salva în „*function 58*” și va fi trimisă către *pythonshell* unde va porni scriptul respectiv (*headshots_node.py*) care va lua variabila respectivă salvată (numele studentului) și va crea un folder cu numele studentului în directorul „*dataset*”, urmând ca toate fotografiile făcute să fie încărcate automat în directorul respectiv, apăsând tasta „*ESC*” închidem întreg procesul. După ce toți studenții au fost înregistrați următorul pas este să antrenăm modelul prin pornirea fișierului de antrenare (*Start app for train*) cât și pornirea serverului (*start server*) în acest fel, după ce toate aceste modele au fost antrenate putem porni camera și server-ul pentru a vedea dacă totul funcționează. În figura de mai jos sunt configurațiile în node red pentru aceste noduri.

Înregistrarea studenților este necesară doar în primul laborator pentru a colecta datele respective (*fotografiile*) astfel încât pe baza acestor date să putem antrena modele respective pentru orele următoare în acest fel putem urmări în timp real ora și data la care studentul intră în sala de curs/laborator, iar pentru restul laboratoarelor nu o să mai fie necesar să facem acest lucru doar în cazul în care un student nu a fost la primul laborator o să fie necesară înregistrarea.

Fig56. Configurația nodurilor pentru pornirea scriptului *headshots_node.py*

5 RECUNOAȘTEREA FACIALĂ, LOCAL CU OPENCV, ONLINE CU FLASK

În această secțiune sunt prezentate două fișiere(*ivanc7_bun.py*,*ivanc12.py*, fiecare având același scop și anume să realizeze recunoaștere facială, atât primul fișier cât și cel de-al doilea diferă în primul rând la transmiterea datelor pentru vizualizare. Primul dintre ele(*ivanc7_bun.py*) rulează local (în sensul că o să deschidă o fereastră GUI local pe dispozitivul pe care îl folosim fie pe calculator sau pe laptop), fereastra respectivă se deschide prin intermediul bibliotecii OpenCV folosind funcția „*cv2.imshow()*”, acest *imshow* este cel care deschide acea fereastră GUI doar în momentul în care avem și „*device=0*”(dacă îl rulăm din dashboard). Cel de-al doilea fișier(*ivanc12.py*) utilizează un server Flask pentru a reda aceeași funcționalitate într-un mediu web, oferind posibilitatea de a accesa și vizualiza recunoașterea facială dintr-un browser, de pe orice dispozitiv conectat în rețea.

Rularea primului fișier din terminal ar trebui să funcționeze fără să aibă probleme cu deschiderea acelei ferestre GUI cu funcția *cv2.imshow()*. Problema apare în momentul în care încercăm să rulăm acel fișier din dashboard prin *pythonsheel* iar rezolvarea acestei probleme pe windows este să scriem la începutul fișierului (după importuri) comanda următoare " *os.environ['DISPLAY']=':0'* ". Această comandă trece DISPLAY-UL pe 0, el fiind inițial „None” când îl pornim din dashboard, practic noi când îl punem pe 0 îi spunem să ne deschidă aceea fereastră grafică pe primul ecran, dacă mai avem un ecran putem să îi spunem să își deschidă pe al doilea punând acel display pe 1, este foarte important de reținut că această comandă se utilizează doar pentru a putea alege unde să se deschidă ferestrele grafice, pe ce ecran.

În cel de-al doilea fișier (*ivanc12.py*) se creează o aplicație web utilizând Flask în care o să definim un endpoint „*video_feed*”, și avem o funcție care generează în mod continuu cadrele video procesate și se trimite cadru cu cadru către browser într-un format special (stream de imagini jpeg, unul după altul), în acest fel putem vedea în timp real și de la distanță ceea ce vede camera. Acest lucru este mai avantajos dacă dorim să ne conectăm de la distanță putem face asta (nu în cazul nostru deoarece nu avem configurat un port forwarding pentru a putea accesa serverul din afara rețelei), dar îl putem accesa local dacă suntem pe același router.

5.1 COMPARAREA METODELOR DE AFIȘARE A FLUXULUI VIDEO ÎN PYTHON PRIN CV2.IMSHOW() ȘI FLASK, EVIDENȚIIND AVANTAJELE ȘI DEZAVANTAJELE FIECĂREI ABORDĂRI

În cadrul aplicațiilor care implică procesare video în Python fie că este vorba de supraveghere video, detecția mișcării, recunoașterea facială sau alte aplicații din domeniul viziunii artificiale, este esențial ca fluxul video să fie afișat într-un mod eficient și adaptat în așa fel încât să servească unui scop aplicația respectivă. Există mai multe modalități prin

care un flux video poate fi afișat, însă cele mai frecvente le am amintit amintit anterior(utilizarea funcției `cv2.imshow()` din biblioteca OpenCV,transmiterea fluxului video printr-o aplicație web realizată cu Flask).Fiecare dintre aceste metode are propriile caracteristici, avantaje și limitări.Alegerea uneia sau alteia depinde în mare măsură de de contextul în care este utilizată aplicația. Dacă aplicația rulează local și este destinată unui singur utilizator o fereastră locală cu „cv2.imshow” este adesea suficientă.În schimb dacă fluxul video trebuie accesat de la distanță de pe alte dispozitive sau de mai mulți utilizatori simultan,o soluție de tip web streaming cu flask devine mult mai potrivită. În continuare vom analiza avantajele și dezavantajele fiecărei metode.

5.2 AFIȘAREA CU CV2.IMSHOW()

Această este o metodă clasică și rapidă oferită de OpenCV pentru afișarea cadrelor video direc într-o fereastră locală.

Avantaje:

Implementarea este foarte simplă și rapidă,ideală pentru teste locale,prototipuri sau aplicații de laborator

Performanța este ridicată deoarece totul se desfășoară local fără latențe provocate de rețea sau transmisii externe

Oferă control complet asupra ferestrei de afișare, permițând redimensionarea, suprapunerea de text sau grafice,capturarea tastelor și alte funcționalități utile pentru dezvoltare

Dezavantaje

Este limitată la utilizarea locală nefiind posibil accesul la fluxul video de pe alte dispozitive sau prin rețea

Poate avea probleme de compatibilitate cu anumite medii de lucru ,cum ar fii Jupyter Notebook sau Visual Studio Code unde ferestrele de tip GUI nu funcționează întotdeauna corect.

Nu este potrivită pentru aplicații care trebuie să fie accesibile de mai mulți utilizatori sau pentru soluții scalabile în medii de producție.

5.3 AFIȘAREA CU FLASK(STREAMING VIDEO ÎN BROWSER)

Această abordare presupune transmiterea fluxului video prin intermediul unei aplicații web construite cu Flask,ceea ce permite afișarea acestuia direct într-un browser.

Avantaje:

Fluxul video poate fi accesat de la distanță de pe orice dispozitiv conectat la rețea,indiferent că este vorba despre un telefon, o tabletă sau un alt computer

Este ideal pentru aplicații web sau proiecte IoT, unde interfața trebuie să fie accesibilă prin browser și să nu depinde de o interfață locală.

Permite scalabilitate mult mai bună, fiind capabil să servească mai mulți utilizatori simultan.

Dezavantaje:

Configurarea este mai complexă, necesitând cunoștințe despre Flask, HTML, precum și despre modul în care se transmite fluxul video prin tipul de conținut „multipart/x-mixed-replace”.

Comparativ cu afișarea locală, poate apărea o întârziere ușoară în redarea video cauzată de transmiterea prin rețea și prin protocoalele HTTP.

Consumul de resurse este mai mare, mai ales dacă aplicația transmite simultan către mai mulți clienți ceea ce implică o procesare video și un trafic de date suplimentar pe server.

6 ACCESAREA ȘI ADMINISTRAREA DE LA DISTANȚĂ A UNUI SERVER NODE-RED DASHBOARD

În contextul actual în care tehnologia evoluează rapid iar conectivitatea a devenit un element definitoriu al societății moderne, nevoie de a controla sisteme de la distanță este o necesitate în cele mai multe cazuri, fie că vorbim de aplicații industriale, educaționale, casnice, gestionarea eficientă a informațiilor în timp real joacă un rol esențial în creșterea performanței și în optimizarea proceselor.

În acest sens mi-am propus să explorez potențialul platformei Node-RED un instrument open-source dezvoltat de IBM, care facilitează automatizarea proceselor printr-o abordare vizuală bazată pe fluxuri logice. Platforma se remarcă prin simplitate, flexibilitate și o comunitate activă, ceea ce o face ideală pentru implementări rapide chiar și pe echipamente cu resurse limitate precum Raspberry Pi [20]. Aceste fiind spuse în această secțiune dorim să expunem public Node-Red Dashboard folosind Port Forwarding și ngrok.

6.1 PORT FORWARDING

Atunci când dorim să facem un server accesibil pe internet este modul în care funcționează rețelele locale și implicit routerele. Dispozitivele dintr-o rețea locală (calculator, raspberry pi, laptop) nu sunt vizibile direct din exterior deoarece routerul le ascunde în spatele unei adrese ip publice folosind NAT (Network Address Translation).

În acest caz intervine port forwarding-ul care este o metodă prin care îi putem adresa direct routerului ce ar trebui să facă de exemplu „tot ce vine pe portul X din afară (de pe internet) trimite către dispozitivul Y din rețeaua locală pe portul Z. Practic deschidem un port între exterior și un anumit serviciu din rețeaua noastră internă.

Dorim să facem același lucru, și anume să facem serverul nostru 192.168.1.120 (Raspberry Pi, portul 1889) public, astfel încât să îl putem accesa și de pe internet, nu doar local. Am intrat în setările de configurare ale routerului 192.168.1.1 (putem scrie această adresă în browser, iar apoi ni se va deschide o fereastră în care trebuie să introducem user-ul și parola). Ca să facem acest lucru, este necesar să fim conectați la routerul la care dorim să facem port forwarding (Fig. 57).

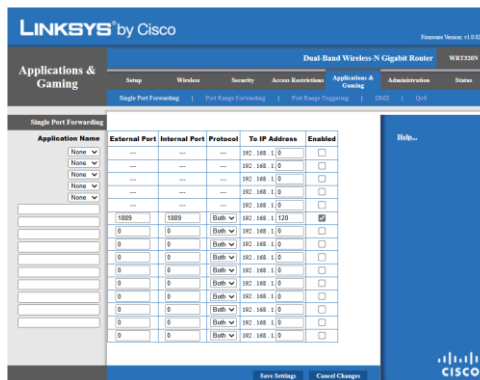


Fig57. Configurare port forwarding

Intrăm pe <https://whatismyipaddress.com/> pentru a obține IP-ul public al rețelei (de pe care noi o să încercăm să accesăm serverul Node-RED). În cazul nostru, IP-ul public este „193.226.9.80”. Cu acest IP încercăm pe browser să intrăm folosind portul 1889, astfel: 193.226.9.80:1889/ui. Ce am observat este că nu funcționează deoarece suntem în spatele unui CG-NAT (Carrier Grade Network Address Translation)[21], iar IP-ul nostru public, pe care l-am obținut, nu este IP-ul real care ne scoate în afara rețelei către internet. Acest lucru l-am observat deoarece am utilizat comanda pe Windows: `tracert 8.8.8.8` și pe Debian/Linux `traceroute 8.8.8.8`. Această comandă ne ajută să vedem care sunt „hops” — punctele de acces prin care trece traficul de internet pentru a ajunge la destinație.

```
Tracing route to dns.google [8.8.8.8]
over a maximum of 30 hops:
  0  0 ms  0 ms  0 ms  192.168.1.1
  1  2 ms  2 ms  40 ms 192.168.1.1
  2  4 ms 13 ms  9 ms 10.22.1.1
  3  3 ms  2 ms  2 ms 10.22.0.1
  4  3 ms  3 ms  3 ms 172.16.89.1
  5  6 ms  3 ms  3 ms 193.226.9.65
  6  5 ms  3 ms  5 ms 217.73.174.189
  7 16 ms 15 ms
```

Fig58.Tracing route

6.2 NGROK

Ngrok este un instrument care creează tuneluri securizate între un server local și internet oferind astfel șansa de a face vizibil un server local pe internet cu tot cu un link public.

În cadrul acestui proiect am dorit să expun Node-Red Dashboard pe internet astfel încât oricine să poată avea acces atâta timp cât user ul și parola sunt corecte. Utilizând ngrok nu a mai fost necesar pentru a configura routerul. Cu ajutorul unei comenzi foarte simple(`ngrok http 1889`) am reușit să creez un tunel securizat între portul local folosit de node-red și internet. După rularea acelei comenzi am primit un link public, accesibil de pe orice dispozitiv cu conexiune la internet.

N. CONCLUZII

Scopul prezentei lucrări de disertație a fost de a explora și integra tehnologii moderne în vederea implementării unui sistem complet de recunoaștere facială pentru detectarea prezenței studenților. Lucrarea este structurată pe mai multe segmente, fiecare abordând aspecte specifice ale dezvoltării, de la procesarea imaginilor până la gestionarea comunicațiilor și interacțiunea cu utilizatorul final.

În prima parte a lucrării s-a abordat dezvoltarea interfeței grafice și a algoritmilor de detecție facială. Pentru acest lucru s-a utilizat biblioteca OpenCV împreună cu funcția `cv2.imshow()`, care a permis afișarea în timp real a fluxului video (la nivel local) și evidențierea persoanelor detectate. Acest proces a reprezentat fundamentul sistemului, demonstrând capacitatea de prelucrare rapidă și precisă a imaginilor.

A doua parte a lucrării s-a concentrat pe integrarea și gestionarea fluxului de date. Pentru a facilita monitorizarea și controlul acestui flux s-a folosit Node Red Dashboard, care a transformat întregul proces de prelucrare într-o soluție accesibilă și interactivă. De asemenea, integrarea cu framework-ul Flask a permis transmiterea cadrelor video către un dashboard web, oferind o viziune unitară și ușor de accesat a întregului sistem.

În a treia etapă s-a pus accentul pe partea de alertare și securitate. Folosirea GMAIL API a permis configurarea unui mecanism automat de trimiterea prin e-mail a imaginii cu persoanele necunoscute către administrator în cazul detectării unor persoane necunoscute. Tot acest proces a fost implementat și optimizat pe Raspberry Pi, demonstrând eficiență și portabilitatea soluției, precum și capacitatea de a opera într-un mediu cu resurse limitate.

În urma rezultatelor obținute, s-a demonstrat că integrarea armonioasă a unor tehnologii diverse, cum ar fi OpenCV, Node-RED, Flask și GMAIL API, poate conduce la realizarea unui sistem robust și flexibil de monitorizare a prezenței. Pe lângă creșterea nivelului de securitate, soluția implementată oferă posibilitatea extinderii și adaptării la alte aplicații din domeniul automatizării și procesării imaginii. Astfel lucrarea nu doar că valorifică impactul rețelelor de calcul și al tehnologiilor web, dar reprezintă și un exemplu concret de aplicare a acestor tehnologii în contextul real, cu beneficii semnificative atât în industrie cât și în mediul academic.

7 BIBLIOGRAFIE

- [1] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," Mar. 2015, doi: 10.1109/CVPR.2015.7298682.
- [2] J. Deng, J. Guo, J. Yang, N. Xue, I. Kotsia, and S. Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," Jan. 2018, doi: 10.1109/TPAMI.2021.3087709.
- [3] "Recunoașterea facială: tehnologie, aplicații și implicațiile etice", Accessed: Mar. 09, 2025. [Online]. Available: <https://revistaverso.ro/recunoasterea-faciala-tehnologie-aplicatii-si-implicatiile-etice/>
- [4] "Comisia Europeană anunță interzicerea a opt utilizări ale inteligenței artificiale." Accessed: Mar. 14, 2025. [Online]. Available: <https://evz.ro/comisia-europeana-anunta-interzicerea-a-opt-utilizari-ale-inteligentei-artificiale.html>
- [5] "Broad-scale applications of the Raspberry Pi: A review and guide for biologists." Accessed: Mar. 17, 2025. [Online]. Available: <https://besjournals.onlinelibrary.wiley.com/doi/full/10.1111/2041-210X.13652>
- [6] "Raspberry Pi OS." Accessed: Feb. 06, 2025. [Online]. Available: <https://www.raspberrypi.com/software/>
- [7] "Install OpenCV on Jetson Nano", Accessed: Feb. 08, 2025. [Online]. Available: <https://qengineering.eu/install-opencv-on-jetson-nano.html>
- [8] "Jagtap, A. M., Kangale, V., Unune, K., & Gosavi, P. (2019, February). A Study of LBPH, Eigenface, Fisherface and Haar-like features for Face recognition using OpenCV. In 2019 International Conference on Intelligent Sustainable Systems (ICISS)(pp. 219-224). IEEE".
- [9] "Sigut, J., Castro, M., Arnay, R., & Sigut, M. (2020). OpenCV basics: a mobile application to support the teaching of computer vision concepts. IEEE Transactions on Education, 63(4), 328-335".
- [10] "Adusumalli, H., Kalyani, D., Sri, R. K., Pratapteja, M., & Rao, P. P. (2021, February). Face Mask Detection Using OpenCV. In 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)(pp. 1304-1309). IEEE".
- [11] "What is face detection and how does it work?", Accessed: Apr. 03, 2025. [Online]. Available: <https://www.techtarget.com/searchenterpriseai/definition/face-detection?>
- [12] "Face Recognition using Open Source Computer Vision Library (OpenCV) with Python", Accessed: Apr. 05, 2025. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/9964836>
- [13] "Adusumalli, H., Kalyani, D., Sri, R. K., Pratapteja, M., & Rao, P. P. (2021, February). Face Mask Detection Using OpenCV. In 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)(pp. 1304-1309). IEEE".
- [14] "Sriratana, W., Mukma, S., Tammarugwattana, N., & Sirisantamrid, K. (2018, July). Application of the OpenCV-Python for Personal Identifier Statement. In 2018

International Conference on Engineering, Applied Sciences, and Technology (ICEAST)(pp. 1-4). IEEE”.

[15]

“James, C., & Nettikadan, D. (2019, April). Student monitoring system for school bus using facial recognition. In 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)(pp. 659-663). IEEE”.

[16] “Swap Memory: What It Is, How It Works, and How to Manage It.”

[17] “Histogram of oriented gradients”, Accessed: Mar. 03, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients

[18] “Getting Started With Google APIs For Python Development.” Accessed: Feb. 03, 2025. [Online]. Available: <https://www.youtube.com/watch?v=PKLG5pfs4nY>

[19] “Node Red Dashboard Image Upload and save to file or database.” Accessed: Apr. 06, 2025. [Online]. Available: <https://www.youtube.com/watch?v=LsOFOPaUxsw>

[20] “Node-RED.” Accessed: Jan. 09, 2025. [Online]. Available: <https://en.wikipedia.org/wiki/Node-RED>

[21] “What is Carrier-grade NAT (CGN/CGNAT)?” Accessed: Apr. 24, 2025. [Online]. Available: <https://www.a10networks.com/glossary/what-is-carrier-grade-nat-cgn-cgnat/>

8 ANEXE

8.1 Codul sursă pentru recunoașterea facială (var1 fără Flask)

```
#ivan_c7bun.py
# Importa pachetele necesare
from imutils.video import VideoStream
from imutils.video import FPS
import face_recognition
import imutils
import pickle
import time
import cv2
import base64
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.base import MIMEBase
from email import encoders
from google.oauth2.credentials import Credentials
from googleapiclient.discovery import build
from datetime import datetime, timedelta
import socket
import os
import pprint
print("DISPLAY =",os.environ.get('DISPLAY'))
env_var=os.environ
os.environ['DISPLAY']=':0'
pprint.pprint(dict(env_var),width=1)
server=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
server.bind(("0.0.0.0",5005))
server.listen(2)

conn, addr=server.accept()
# Configuratii initiale
SCOPES = ['https://mail.google.com/']
TOKEN_PATH = 'token.json'
credentials = Credentials.from_authorized_user_file('token.json', SCOPES)

# Creeaza serviciul Gmail
service = build('gmail', 'v1', credentials=credentials)

def trimite_email_cu_poza(img_name):
    expeditor = "d003lab@gmail.com"
```

```
destinatar = "d003lab@gmail.com"  
subiect = "Persoana necunoscuta in D101"  
text_email = "Poza atasata: "
```

```
mesaj = MIMEMultipart()  
mesaj['to'] = destinatar  
mesaj['from'] = expeditor  
mesaj['subject'] = subiect  
mesaj.attach(MIMEText(text_email, 'plain'))
```

```
cale_pozei = f"/home/ivan/opencv/build/facial_recognition/{img_name}"  
nume_pozei = f"{img_name}"
```

```
with open(cale_pozei, 'rb') as fisier:  
    continut_poza = fisier.read()
```

```
atasament = MIMEBase('image', 'jpeg')  
atasament.set_payload(continut_poza)  
encoders.encode_base64(atasament)  
atasament.add_header('Content-Disposition', f'attachment; filename={nume_pozei}')  
mesaj.attach(atasament)
```

```
mesaj_codificat = base64.urlsafe_b64encode(mesaj.as_bytes()).decode()
```

```
try:  
    mesaj_trimis = service.users().messages().send(userId="me", body={'raw':  
mesaj_codificat}).execute()  
    print(f"Email trimis cu succes! ID mesaj: {mesaj_trimis['id']}")  
except Exception as error:  
    print(f"A aparut o eroare: {error}")
```

```
# Variabile globale pentru contorizare  
people_count = 0  
known_face_encodings = []  
known_face_names = [] # Lista pentru numele fetelor cunoscute
```

```
# Initializeaza 'currentname' pentru a detecta o persoana noua  
currentname = "unknown"  
encodingsP = "encodings.pickle"  
cascade = "haarcascade_frontalface_default.xml"
```

```
print("[INFO] loading encodings + face detector...")  
data = pickle.loads(open(encodingsP, "rb").read())
```

```
detector = cv2.CascadeClassifier(cascade)

# Incarca encodările și numele fetelor cunoscute
data_encodings = data["encodings"]
data_names = data["names"]
known_face_encodings.extend(data_encodings)
known_face_names.extend(data_names)

print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
print("ceva1")
time.sleep(2.0)

fps = FPS().start()
print("ceva")
# Dictionar pentru ultima detectare a fiecărei persoane cunoscute
last_seen = {}

def log_intrare(ume):
    with open("log_persoane_cunoscute.txt", "a") as fisier_log:
        data_ora = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        fisier_log.write(f"{data_ora} - {ume} a intrat in incapere\n")
        print("INTRARE",last_seen)

def log_iesire(ume):
    with open("log_persoane_cunoscute.txt", "a") as fisier_log:
        data_ora = datetime.now()-timedelta(minutes=2)
        data_ora_completa=data_ora.strftime("%Y-%m-%d %H:%M:%S")
        fisier_log.write(f"{data_ora_completa} - {ume} a iesit din incapere\n")
        print("IESIRE",last_seen)

while True:
    conn.settimeout(1)
    print("timeout")
    try:
        print("try1")
        data=conn.recv(1024).decode().strip()
        print("try2")
        print("data",data)
        if data=="q":
            print("OPRIRE!")
            break
    except socket.timeout:
        pass
```

```
frame = vs.read()
frame = imutils.resize(frame, width=500)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

rects = detector.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30,
30),
                                flags=cv2.CASCADE_SCALE_IMAGE)
boxes = [(y, x + w, y + h, x) for (x, y, w, h) in rects]
encodings = face_recognition.face_encodings(rgb, boxes)
names = []
print("TOTUL OK")
for encoding in encodings:
    matches = face_recognition.compare_faces(known_face_encodings, encoding,
tolerance=0.6)
    name = "Unknown"

    if True in matches:
        matchedIdxs = [i for (i, b) in enumerate(matches) if b]
        counts = {}

        for i in matchedIdxs:
            name = known_face_names[i]
            counts[name] = counts.get(name, 0) + 1

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

        # Inregistrare intrare daca e prima detectare
        if name not in last_seen:
            log_intrare(name)

        # Actualizare ultima detectare
        last_seen[name] = datetime.now()

    else:
        known_face_encodings.append(encoding)
        known_face_names.append(name)
        people_count += 1
        print(f'Noua persoana detectata! Total persoane: {people_count}')

    img_name = f"person_{people_count}.jpg"
    cv2.imwrite(img_name, frame)
    trimite_email_cu_poza(img_name)
```



```
names.append(name)

# Verifica persoanele care au iesit
current_time = datetime.now()
for name in list(last_seen.keys()):
    if (current_time - last_seen[name]) > timedelta(minutes=2):
        log_iesire(name)
        del last_seen[name]

for ((top, right, bottom, left), name) in zip(boxes, names):
    cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 225), 2)
    y = top - 15 if top - 15 > 15 else top + 15
    cv2.putText(frame, name, (left, y), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255,
255), 2)
    print("MERGE")

cv2.imshow("Facial Recognition is Running", frame)
key = cv2.waitKey(1) & 0xFF
if key == ord("q"):
    break
conn.close()
server.close()
fps.stop()
print(f"[INFO] elapsed time: {fps.elapsed():.2f}")
print(f"[INFO] approx. FPS: {fps.fps():.2f}")
print(f"Numarul total de persoane detectate: {people_count}")

cv2.destroyAllWindows()
vs.stop()
```

8.2 Codul sursă pentru recunoașterea facială (var2 cu Flask)

```
# Importa pachetele necesare
from imutils.video import VideoStream
from imutils.video import FPS
import face_recognition
import imutils
import pickle
import time
import cv2
```

```
import base64
import os
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.base import MIMEBase
from email import encoders
from google.oauth2.credentials import Credentials
from googleapiclient.discovery import build
from datetime import datetime, timedelta
import socket
from flask import Flask, Response, request
import warnings
import logging
import sys
import threading
print("START")
# Crează serverul Flask pentru streaming video
app = Flask(__name__)
warnings.filterwarnings("ignore", message="This is a development server.")
log = logging.getLogger('werkzeug')
log.setLevel(logging.ERROR)

# Variabilă globală pentru controlul închiderii
should_exit = False

# Configurări inițiale
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(("0.0.0.0", 5005))
server.listen(1)

conn, addr = server.accept()

SCOPES = ['https://mail.google.com/']
TOKEN_PATH = 'token.json'
credentials = Credentials.from_authorized_user_file('token.json', SCOPES)
service = build('gmail', 'v1', credentials=credentials)

def trimite_email_cu_poza(img_name):
    try:
        if not os.path.exists(img_name):
            print(f"[EROARE] Fișierul {img_name} nu există!")
            return
```

```
expeditor = "d003lab@gmail.com"  
destinatar = "d003lab@gmail.com"  
subiect = "Persoana necunoscuta in D101"  
text_email = "Poza atasata: "
```

```
mesaj = MIMEMultipart()  
mesaj['to'] = destinatar  
mesaj['from'] = expeditor  
mesaj['subject'] = subiect  
mesaj.attach(MIMEText(text_email, 'plain'))
```

```
with open(img_name, 'rb') as fisier:  
    continut_poza = fisier.read()
```

```
atasament = MIMEBase('image', 'jpeg')  
atasament.set_payload(continut_poza)  
encoders.encode_base64(atasament)  
atasament.add_header('Content-Disposition', f'attachment; filename={img_name}')  
mesaj.attach(atasament)
```

```
mesaj_codificat = base64.urlsafe_b64encode(mesaj.as_bytes()).decode()
```

```
mesaj_trimis = service.users().messages().send(userId="me", body={'raw':  
mesaj_codificat}).execute()  
print(f"[SUCCES] Email trimis cu ID: {mesaj_trimis['id']}")
```

```
except Exception as error:  
    print(f"[EROARE] A aparut o eroare la trimiterea emailului: {error}")
```

```
# Variabile globale pentru contorizare  
people_count = 0  
known_face_encodings = []  
known_face_names = []  
currentname = "unknown"  
encodingsP = "encodings.pickle"  
cascade = "haarcascade_frontalface_default.xml"
```

```
print("[INFO] loading encodings + face detector...")  
data = pickle.loads(open(encodingsP, "rb").read())  
detector = cv2.CascadeClassifier(cascade)
```

```
# Incarcă encodările și numele fetelor cunoscute
data_encodings = data["encodings"]
data_names = data["names"]
known_face_encodings.extend(data_encodings)
known_face_names.extend(data_names)

print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)

fps = FPS().start()
last_seen = {}

def log_intrare(num):
    with open("log_persoane_cunoscute.txt", "a") as fisier_log:
        data_ora = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        fisier_log.write(f"{data_ora} - {num} a intrat in incapere\n")
        print(f"[LOG] INTRARE: {num}")

def log_iesire(num):
    with open("log_persoane_cunoscute.txt", "a") as fisier_log:
        data_ora = datetime.now() - timedelta(minutes=2)
        data_ora_completa = data_ora.strftime("%Y-%m-%d %H:%M:%S")
        fisier_log.write(f"{data_ora_completa} - {num} a iesit din incapere\n")
        print(f"[LOG] IESIRE: {num}")

def stop_flask():
    func = request.environ.get('werkzeug.server.shutdown')
    if func is None:
        print("[EROARE] Nu pot opri serverul Flask - nu rulează în modul de dezvoltare?")
    else:
        func()
    print("[INFO] Serverul Flask s-a oprit!")

def cleanup_resources():
    print("[INFO] Se închid resursele...")
    conn.close()
    server.close()
    fps.stop()
```

```
cv2.destroyAllWindows()
vs.stop()
print("[INFO] Toate resursele au fost închise!")
```

```
def gen_frames():
    global should_exit, people_count
    while not should_exit:
        conn.settimeout(1)
        try:
            data = conn.recv(1024).decode().strip()
            if data == "q":
                print("[INFO] Oprește cerută de client!")
                should_exit = True
                conn.close()
                server.close()
                os._exit(0) # Oprește imediată a aplicației (kill instantaneu)
                threading.Thread(target=stop_flask).start()
                break
        except socket.timeout:
            pass

    frame = vs.read()
    frame = imutils.resize(frame, width=500)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    rects = detector.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5,
minSize=(30, 30),
                                flags=cv2.CASCADE_SCALE_IMAGE)
    boxes = [(y, x + w, y + h, x) for (x, y, w, h) in rects]
    encodings = face_recognition.face_encodings(rgb, boxes)
    names = []

    for encoding in encodings:
        matches = face_recognition.compare_faces(known_face_encodings, encoding,
tolerance=0.6)
        name = "Unknown"

        if True in matches:
            matchedIdxs = [i for (i, b) in enumerate(matches) if b]
            counts = {}
```

```
for i in matchedIdxs:
    name = known_face_names[i]
    counts[name] = counts.get(name, 0) + 1

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

if name not in last_seen:
    log_intrare(name)

    last_seen[name] = datetime.now()

else:
    people_count += 1
    img_name = f"person_{people_count}.jpg"

    success = cv2.imwrite(img_name, frame)
    if not success:
        print(f"[EROARE] Nu am putut salva imaginea {img_name}!")
        continue

    print(f"[DETECTIE] Noua persoana detectata! Salvata ca {img_name}")
    known_face_encodings.append(encoding)
    known_face_names.append(name)

    trimite_email_cu_poză(img_name)

names.append(name)

current_time = datetime.now()
for name in list(last_seen.keys()):
    if (current_time - last_seen[name]) > timedelta(minutes=2):
        log_iesire(name)
        del last_seen[name]

for ((top, right, bottom, left), name) in zip(bboxes, names):
    cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 225), 2)
    y = top - 15 if top - 15 > 15 else top + 15
    cv2.putText(frame, name, (left, y), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255,
255), 2)

_, jpeg = cv2.imencode('.jpg', frame)
frame = jpeg.tobytes()
```

```
yield (b'--frame\r\n'  
      b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')
```

```
@app.route('/video_feed')  
def video_feed():  
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')
```

```
if __name__ == '__main__':  
    print("[INFO] Serverul Flask a pornit!")  
    try:  
        app.run(host='0.0.0.0', port=5000, threaded=True)  
    finally:  
        os._exit(0) # Oprește imediată a aplicației (kill instantaneu)
```

8.3 Cod sursă pentru colectarea imaginilor

```
#headshots_node.py  
import cv2  
import os  
import sys  
print("DISPLAY =",os.environ.get('DISPLAY'))  
env_var=os.environ  
os.environ['DISPLAY']=':0'  
#name = sys.stdin.readline().strip()  
name=input().strip()  
if not name:  
    print("Error")  
    sys.exit(1)  
cam = cv2.VideoCapture(0)  
  
cv2.namedWindow("press space to take a photo", cv2.WINDOW_NORMAL)  
cv2.resizeWindow("press space to take a photo", 500, 300)  
  
folder_path=os.path.join("dataset",name)  
if not os.path.exists(folder_path):  
    os.makedirs(folder_path)  
    print(f'Folder '{folder_path}' creat.")  
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 = f"{folder_path}/{name}_{img_counter}.jpg"
    cv2.imwrite(img_name, frame)
    print("{} written!".format(img_name))
    img_counter += 1
```

```
cam.release()
```

```
cv2.destroyAllWindows()
```

8.4 Cod sursă pentru antrenarea modelelor

```
#train_model1.py
#!/usr/bin/python

# import the necessary packages
from imutils import paths
import face_recognition
# import argparse
import pickle
import cv2
import os
import socket

print("Trebuie pornit serverul")
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(("0.0.0.0", 5005))
server.listen(1)
conn, addr = server.accept()
```

```
stop=False
start_server=False
while True:
    if stop:
        break
    conn.settimeout(1)
    print("timeout")
    try:
        print("try1")
        data1=conn.recv(1024).decode().strip()
        if data1=="b":
            start_server=True

        print("try2")
        print("data", data1)

        if data1 == "q":
            start_server=False
            print("OPRIRE! PROGRAM SI SERVER")
            break
        if stop:
            break

        if data1 == "t" and start_server==True:
            # our images are located in the dataset folder
            print("[INFO] start processing faces...")
            imagePaths = list(paths.list_images("dataset"))
            print("imagePaths",imagePaths)
            # 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)))
                print("PROCESING 1")
                conn.settimeout(1)
                try:
                    data2=conn.recv(1024).decode().strip()
                    print("DATA 2 PART1",data2)
                    if data2 == "q":
```

```
        stop=True
        print("OPRIRE ANTRENAMENT!!")
        break
except socket.timeout:
    pass
name = imagePath.split(os.path.sep)[-2]

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

# detect the (x, y)-coordinates of the bounding boxes
boxes = face_recognition.face_locations(rgb, model="hog")

# compute the facial embedding for the face
encodings = face_recognition.face_encodings(rgb, boxes)
print("PROCESING 2")
# 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}
with open("encodings.pickle", "wb") as f:
    f.write(pickle.dumps(data))
```

```
except socket.timeout:
    pass
conn.close()
server.close()
```

8.5 Funcțiile din Node-RED (var2)

```
//function 29
if (msg.payload=== "b"){
    global.set('serverStatus',true);
    msg.payload='b';
    return msg;
}else if (msg.payload=== "q"){
    global.set('serverStatus',false)
    msg.payload="q";
```

```
    return msg;
  }else if (msg.payload=== "t"){
    if (global.get('serverStatus')===true){
      return msg;
    }else{
      msg.payload="Serverul nu este pornit";
      node.error("Serverul nu este pornit",msg)
      return null;
    }
  }
}
```

//Set input variable

```
context.global.set('Variabila_Input', msg.payload);
return msg;
```

//Convertirea datelor din base64 si salvare fotografie

```
// msg.payload conține datele fotografiei în format base64
let pay = context.global.get('Variabila_Input');
console.log(pay)
if (typeof pay=== "undefined"){
  node.error("Eroare: Nu ati selectat folderul dorit");
  return null;
}

const fs=global.get('fs');
// let pay=msg.payload.pay;

// let base64Data = msg.payload.pay2;
let base64Data = msg.payload;
// console.log(base64Data)
// Generează un nume unic pentru fotografie folosind timestamp și un număr aleatoriu
let uniqueName = 'photo_' + Date.now() + '_' + Math.floor(Math.random() * 10000) + '.jpg';
// Specifică calea unde va fi salvată fotografia
let basePath = `/home/ivan/opencv/build/facial_recognition_var2/dataset/${pay}`; //
Înlocuiește cu calea dorită
let fullPath = basePath + '/' + uniqueName;
// Converteste din base64 și salvează fotografia

let buffer = Buffer.from(base64Data, 'base64');
msg.filename=fs.writeFileSync(fullPath, buffer);
// Returnează calea completă pentru a fi folosită în continuare
```

```
msg.payload = fullPath;  
return msg;
```

//function 48

```
var folders=Array.isArray(msg.payload)?  
msg.payload:[msg.payload];  
folders=folders.map(f=>({  
  path:`/home/ivan/opencv/build/facial_recognition_var2/dataset/${f.title}`,  
  isChecked:f.isChecked  
}))  
folders=folders.filter((item,index,arr)=>arr.findIndex(f=>f.path===item.path)===index);  
console.log("FOLDER F 20 filter",folders)  
  
msg.payload=folders.map(f=>f.path);  
console.log("msg.payload: ",msg.payload)  
return msg;
```

//storage folders

```
var folders=context.global.get("folders") || [] //folderele salvate global  
console.log("PRIMUL",folders)  
var newFolder=msg.payload //folder curent primit  
folders.push(newFolder)  
context.global.set("folders",folders);  
///folders=folders.filter((item,index,self)=>self.findIndex(f=>f.path===item.path)===index);  
console.log("FOLDER 2",folders)  
var counts=folders.reduce((acc,path)=>{  
  acc[path] = (acc[path] || 0)+1;  
  return acc;  
},{});  
var duplicates=Object.keys(counts).filter(path=>counts[path]>1);  
console.log("Duplicate folders: ",duplicates)  
  
folders=folders.filter(folderArray=>!duplicates.includes(folderArray[0]));  
context.global.set("folders",folders)  
console.log("Folders fara duplicate ",folders)  
return null // nu trimite mai departe doar salveaza
```

//function 49

```
var folders=context.global.get("folders") || []

if (msg.payload=== 'DELETE'){
  console.log("FOLDERS DA: ",folders)

  msg.payload=folders.map(folders=>`rm -r '${folders}'`).join(' && ');
  console.log("FOLDERE DE STERS",msg.payload)
  folders=folders.filter(folders=>folders.length>0);
  console.log("FOLDERE GLOBALIST:",folders)
  context.global.set('folders',[])
  console.log("FOLDERE GLOBALIST 2:",folders)
}else{
  msg.payload="echo 'No folders to delete'";
  console.log("CEVA",msg.payload)
}
return msg;
```

//function 52

```
const fs = global.get('fs');
const
filePath="/home/ivan/opencv/build/facial_recognition_var2/log_persoane_cunoscute.txt";
let currentLineIndex=0;
function processNewLines(){
  fs.readFile(filePath,"utf8",(err,data)=>{
    if(err){
      console.error("Eroare",err);
      return null
    }
    const lines=data.split("\n").map(line=>line.trim()).filter(line=>line.length>0);
    while (currentLineIndex<lines.length){
      let msg={
        topic:"new_line",
        payload:lines[currentLineIndex]
      }
      node.send(msg);
      currentLineIndex++;
    }
  })
}

fs.watch(filePath,(eventType)=>{
```

```
    if(eventType==="change"){  
        processNewLines();  
    }  
});  
processNewLines()
```

//function 53

```
msg.topic="new_line";  
msg.payload=msg.payload;  
return msg;
```

//template

```
<div style="font-family: Arial, sans-serif; padding:10px;">  
    <h3>Continut fisier:</h3>  
    <ul id="new-line"></ul>  
</div>  
<script>  
    (function(scope){  
        scope.$watch('msg',function(msg)  
        {  
            if(msg && msg.topic==="new_line"){  
                let ul=document.getElementById("new-line");  
                let li=document.createElement("li");  
                li.textContent=msg.payload;  
                ul.appendChild(li);  
            }  
        })  
    })(scope);  
</script>
```

//template src(pentru integrare flask in Node Red)

```

```

//function 58

```
var name=msg.payload  
msg.payload=name;
```


return msg;

**DECLARAȚIE DE AUTENTICITATE A
LUCRĂRII DE FINALIZARE A STUDIILOR***

Subsemnatul/Subsemnata GHEȚANCA IOAN

Legitimat(ă) cu C.I. seria TZ nr. 935882
CNP 199111224 2265

autorul lucrării PLATFORMĂ INTELIGENTĂ PENTRU URMĂRIREA
ACCESULUI ÎN LABORATOARELE DE CERCETARE
elaborată în vederea susținerii examenului de finalizare a studiilor de
MASTER organizat de către Facultatea de
Inginerie Electrică și Energetică, din cadrul Universității Politehnica Timișoara, sesiunea
Iunie a anului universitar 2024-2025, coordonator
CONF. DR. ING. OCTAVIAN CORNOȘ hând în considerare prevederile Capitolului V – Măsuri
privind asigurarea originalității lucrărilor din Regulamentul privind organizarea și desfășurarea
examenelor de licență/diplomă și disertație în Universitatea Politehnica Timișoara, aprobat prin
HS nr. 30/21.03.2024 și cunoscând faptul că în cazul constatării ulterioare a unor încălcări ale
normelor de etică/a faptului că diploma a fost obținută prin mijloace frauduloase, voi suporta
sanctiunile legale prevăzute de Legea nr. 199/2023 – Legea Învățământului Superior, declar
pe proprie răspundere, că:

- această lucrare este rezultatul propriei activități intelectuale;
- lucrarea nu conține texte, date sau elemente de grafică din alte lucrări sau din alte surse fără ca acestea să nu fie citate, inclusiv situația în care sursa o reprezintă o altă lucrare/alte lucrări ale subsemnatului;
- sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor;
- această lucrare nu a mai fost prezentată în fața unei alte comisii de examen/prezentată public/publicată de licență/diplomă/disertație;
- în elaborarea lucrării am utilizat instrumente specifice inteligenței artificiale (IA) și anume _____ (denumirea) _____ (sursa), pe care le-am citat în conținutul lucrării/nu am utilizat instrumente specifice inteligenței artificiale (IA)¹.

Declar că sunt de acord ca lucrarea să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop.

Timișoara,

Data 13.06.2025 Semnătura GP

*Declarația se completează de student, se semnează olograf de acesta și se inserează în lucrarea de finalizare a studiilor, la sfârșitul lucrării, ca parte integrantă.
¹ Se va păstra una dintre variante: 1 - s-a utilizat IA și se menționează sursa 2 – nu s-a utilizat IA