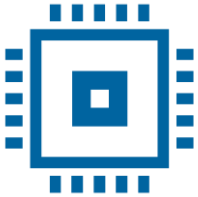


Universitatea
Transilvania
din Brașov
FACULTATEA DE INGINERIE ELECTRICĂ
ȘI ȘTIINȚA CALCULATOARELOR

PROIECT DE DIPLOMĂ

Conducător științific: CARP Marius Cătălin
Titulatura. Prof. Dr. Ing.

Absolvent:
NEAGU Lucian-Alexandru

BRAȘOV, 2023

Departamentul Electronică și Calculatoare
Programul de studii: Electronică Aplicată

NEAGU Lucian-Alexandru

Sistem de securitate cu recunoaștere facială

Conducător științific:
S. L. dr. Ing. CARP Marius Cătălin

Brașov, 2023

Cuprins

Cuprins

LISTA DE FIGURI, TABELE ȘI CODURI SURSĂ

FIGURI

TABELE

CODURI SURSĂ

LISTA DE ACRONIME

ARM – Advanced RISC Machines (o arhitectură de procesor);

AUTH – Authentication (autentificare);

CCTV – Closed Circuit Television (sistem de televiziune cu circuit închis);

CPU – Central Processing Unit (unitate centrală de procesare);

CSI – Camera Serial Interface (interfață serială pentru camere);

DIY – Do-It-Yourself;

D-PHY2 – Display PHY version 2 (versiunea 2 a interfeței fizice pentru afișaje);

ESMTP – Extended Simple Mail Transfer Protocol (protocol extins de transfer de mesaje de poștă electronică);

GPIO – General Purpose Input/Output (intrare/ieșire de uz general);

GPU – Graphics Processing Unit (unitate de procesare grafică);

HDMI – High-Definition Multimedia Interface (interfață multimedia de înaltă definiție);

HOG – Histogram of Oriented Gradients (histograma orientării gradientului);

IANA – Internet Assigned Numbers Authority (autoritatea pentru atribuirea numerelor pe internet);

I2C – Inter-Integrated Circuit (interfață integrată în circuit);

IETF – Internet Engineering Task Force (forța de lucru pentru inginerie pe internet);

IRQ – Interrupt Request (cerere de întrerupere);

LBP – Local Binary Patterns (modele binare locale);

LDA – Linear Discriminant Analysis (analiză discriminantă liniară);

LPDDR4 SDRAM – Low Power Double Data Rate 4 Synchronous Dynamic Random Access Memory (memorie dinamică sincronă de tip acces aleatoriu, cu rată dublă și consum redus de energie, versiunea 4);

MIPI – Mobile Industry Processor Interface (interfață de procesor pentru industria mobilă);

PCB – Printed Circuit Board (placă de circuit tipărit);

PID – Process Identifier;

PIR – Pasive infrared sensor (senzor pasiv cu infraroșu);

RAM – Random Access Memory (memorie cu acces aleatoriu);

SoC – System on a Chip (sistem pe un singur cip);

SMTP – Simple Mail Transfer Protocol (protocol simplu de transfer de mesaje de poștă electronică);

SPI – Serial Peripheral Interface (interfață periferică serială);

SSH – Secure Shell (shell securizat);

SSL – Secure Sockets Layer;

TLS – Transport Layer Security;

UART – Universal Asynchronous Receiver-Transmitter (receptor-transmițător asincron universal);

USB – Universal Serial Bus (magistrală serială universală);

ZIF – Zero insertion force;

1 INTRODUCERE

1.1 SCURTĂ ISTORIE A SISTEMELOR DE SECURITATE

Sistemele de securitate sunt dispozitive și programe care sunt concepute pentru a proteja persoanele, proprietatea și alte active importante împotriva accesului neautorizat, furtului, daunelor sau altor amenințări.

Aceste sisteme pot fi clasificate în funcție de diferite criterii, cum ar fi tipul de amenințări la care sunt expuse, nivelul de securitate oferit, mediul în care sunt instalate și multe altele. Acest proiect se bazează pe două tipuri de sisteme de securitate sisteme de supraveghere video și sisteme de alarmă.

Sisteme de supraveghere video sunt camere de supraveghere care înregistrează imagini video ale locațiilor protejate și le trimit către un sistem central de monitorizare sau de înregistrare. Aceste sisteme pot fi utilizate pentru a monitoriza intrările și ieșirile, pentru a detecta activitatea neobișnuită sau pentru a identifica infractorii.

Sisteme de alarmă sunt dispozitive care sunt proiectate pentru a alerta proprietarul sau autoritățile în cazul unei amenințări. Acestea pot fi sisteme de alarmă la domiciliu, sisteme de alarmă pentru afaceri sau sisteme de alarmă pentru automobile.

Aceste sisteme au evoluat semnificativ de-a lungul timpului. Istoria acestora începe încă din anii '40, primele camere video au fost dezvoltate în Germania, pentru a monitoriza lansarea rachetelor, apoi în anii '50, sistemele de supraveghere video au fost utilizate în Marea Britanie pentru a monitoriza locațiile publice. În anii '60, apare atât sistemul de supraveghere video CCTV (Closed Circuit Television), care a fost dezvoltat și a început să fie utilizat pentru a monitoriza locațiile comerciale și publice , cât și primele sisteme de alarmă cu alarma silențioasă ce au permis alertarea discreta a autorităților în caz de intrare neautorizată. O decadă mai târziu, în anii '70, tehnologia a evoluat astfel încât sistemele de alarmă puteau fi controlate de la distanță prin intermediul telefonului.

În anii '80, camerele video au devenit mai compacte și mai accesibile, ceea ce a dus la o creștere a utilizării lor în sistemele de supraveghere video, după care în anii '90, camerele video digitale au fost dezvoltate, ceea ce a permis stocarea și transferul de imagini video în format digital.

În prezent, camerele video sunt disponibile într-o varietate de dimensiuni și forme și pot fi conectate la rețele de calculatoare pentru a permite monitorizarea de la distanță a locațiilor protejate, iar sistemele de alarmă fără fir sunt din ce în ce mai populare, permițând o instalare mai ușoară și mai flexibilă.

Astfel, putem observa că tehnologia în sistemele de securitate video și a sistemelor de securitate cu alarmă s-a îmbunătățit semnificativ de-a lungul timpului, odată cu dezvoltarea tehnologiilor de captare video și a avansării tehnice a senzorilor, permițând o protecție mai bună și mai eficientă a locațiilor protejate. [14]

1.2 TEMA PROIECTULUI

Scopul acestui proiect este de a implementa un sistem de securitate cu consum de energie redus, care nu ocupă spațiu de stocare în mod inutil și care poate fi implementat relativ simplu în orice casă.

Acesta proiect are la bază un modul Raspberry Pi ce controlează o cameră și un senzor cu infraroșu care detectează întreruperea razei de lumină. Sistemul poate fi utilizat pentru a supraveghea anumite zone de interes, cum sunt punctele de acces cheie, unde senzorul infraroșu cu întrerupere poate fi montat pentru a detecta daca cineva trece prin acel punct. În urma sesizării acesta transmite un semnal la unitatea Raspberry Pi ce activează camera pentru a înregistra persoanele neautorizate, urmând ca aceste imagini captate să se întoarcă la unitatea centrală, pentru a fi procesate și stocate.

Mai departe înregistrarea este trecută printr-un program de recunoaștere facială, iar dacă sistemul nu recunoaște persoana care apare în înregistrare acesta trimite un e-mail proprietarului ce conține înregistrarea video și înștiințarea că cineva a pătruns în incintă.

1.3 OBIECTIVE

Principalele obiective software urmărite în desfășurarea proiectului au fost:

- Familiarizarea cu limbajul de programare Python (versiunea 3.10);
- Dobândirea cunoștințelor necesare pentru a utiliza acest limbaj;
- Utilizarea bibliotecii OpenCV în programul de recunoaștere facială;
- Utilizarea bibliotecii “smtplib”;
- Implementarea senzorului și camerei în codul aplicației;

Principalele obiective hardware urmărite în desfășurarea proiectului au fost:

- Înțelegerea modului de funcționare a senzorului infraroșu cu întrerupere;
- Utilizarea eficientă a camerei de supraveghere;
- Înțelegerea modului de lucru cu un sistem Raspberry Pi;

Principalele obiective privind cercetarea urmărite în desfășurarea proiectului au fost:

- Diverse sisteme de securitate și implementările acestora;
- Capacitățile tehnice ale camerei și senzorului;
- Protocolul SMTP;
- Algoritmii de recunoaștere facială;

1.4 UTILITATEA LUCRĂRII

Un astfel de sistemele de securitate video poate fi folosit în primul rând pentru protecția domiciliului, având o varietate mare de utilități și beneficii, principalul obiectiv fiind apărarea împotriva infracțiunilor. Sistemele de securitate video pot ajuta la protejarea locuinței împotriva

furturilor și a altor infracțiuni prin detectarea activităților suspecte și alertarea proprietarilor sau a autorităților, verificarea vizitatorilor și a livrărilor la domiciliu, pentru a preveni intruziunea în locuință, iar prin implementarea unui software de recunoaștere facială se poate realiza și controlul accesului în locuință sau în spațiul de muncă.

În concluzie, un astfel de sistem poate ajuta la îmbunătățirea securității și siguranței proprietății și a persoanelor din incinta acesteia.

2 ANALIZA STADIULUI ACTUAL

2.1 STADIUL ACTUAL DIN PUNCT DE VEDRE COMERCIAL

Sistemele de securitate video sunt din ce în ce mai populare și avansate din punct de vedere comercial. În prezent, există o varietate de opțiuni disponibile pentru a satisface nevoile și bugetele diferite ale afacerilor. Printre cele mai recente evoluții în domeniu se numără:

Sisteme de securitate video inteligente: Acestea utilizează algoritmi de învățare automată și inteligență artificială pentru a oferi o mai bună detecție a amenințărilor și pentru a face mai ușoară administrarea și accesarea datelor video.

Îmbunătățiri ale rezoluției: Camerele de securitate video au ajuns să posede o claritate din ce în ce mai bună, ajungând până la rezoluția de 4K, oferind o calitate superioară a imaginii.

Sisteme de stocare mai eficiente: Sistemele de stocare de date au devenit mai eficiente și mai ieftine, permițând stocarea unui volum mai mare de date video.

Integrarea cu alte tehnologii: Sistemele de securitate video sunt acum capabile să se integreze cu alte tehnologii de securitate, cum ar fi sistemele de alarmă, controlul accesului și de detectare a incendiilor.

Sisteme de securitate video în cloud: Sistemele de securitate video în cloud permit accesul la date video de oriunde și oricând, prin intermediul unei conexiuni la internet.

Utilizarea dispozitivelor mobile: Aplicațiile mobile permit utilizatorilor să acceseze camerele de securitate video de pe dispozitive mobile, cum ar fi smartphone-uri și tablete.

Figura 2-1 Sistem de supraveghere video în cloud [15]

Stadiul actual al sistemelor de securitate video din punct de vedere comercial este unul foarte avansat, cu multe opțiuni de alegere pentru afaceri de toate dimensiunile și pentru toate nevoile de securitate. [1]

Datorită faptului că Raspberry Pi este o platformă de programare și de implementare a soluțiilor tehnice relativ ieftină și la dispoziția oricui, exista multe proiecte în care acesta poate fi implementat. Având în vedere tema acestui proiect, în continuare voi prezenta alte două proiecte simple cu funcționalități similare, găsite pe diverse pagini web destinate amatorilor.

Utilizând Pi Cam si un Raspberry Pi se poate crea un sistem pentru monitorizarea bebelușilor sau a animalelor de companie. Camera trebuie fixată într-un punct cu vedere asupra subiectului, iar apoi conectata la Raspberry. Pentru a vede înregistrările camerei trebuie doar să ne conectam la Raspberry prin SSH, extrem de simplu. [16]

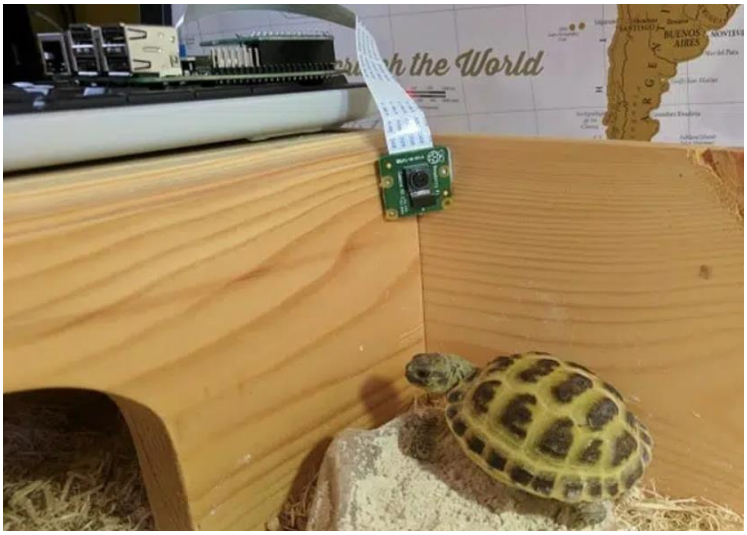


Figura 2-2 Exemplu de proiect de monitorizare cu Raspberry Pi [16]

2.2 SCURTĂ PREZENTARE A COMPONENTELOR HARDWARE ȘI SOFTWARE FOLOSITE

Conceptual proiectul are trei elemente hardware de bază Raspberry Pi, camera, senzorul IR cu întrerupere și se folosește de limbajul de programare Python pentru a realiza procesul de recunoaștere facială, precum și pentru a folosi protocolul SMTP. Mai jos voi prezenta pe scurt componentele hardware și software folosite în acest proiect.

2.2.1 Raspberry Pi 4

Raspberry Pi 4 este un computer mono-circuit de dimensiuni reduse, conceput de către Fundația Raspberry Pi. Este o versiune îmbunătățită a modelelor anterioare de Raspberry Pi, oferind o putere de calcul mai mare și caracteristici hardware îmbunătățite.

Raspberry Pi 4 rulează sistemul de operare Raspberry Pi OS, o distribuție de Linux, dar poate fi folosit cu o gamă largă de sisteme de operare. De asemenea, poate fi utilizat într-un număr mare de aplicații, inclusiv ca server, sistem de automatizare a locuinței, sistem de control al

roboților, dispozitiv de învățare și multe altele. Acesta este o alegere populară printre dezvoltatorii și entuziaștii de tehnologie, datorită dimensiunii sale reduse, costului accesibil și a flexibilității sale, permițând utilizatorilor să creeze proiecte personalizate și să își îndeplinească nevoile specifice. [2][17]

Figura 2-3 Raspberry Pi 4 [17]

2.2.2 Cameră 5MP Night Vision

Camera 5MP Night Vision pentru Raspberry Pi este o cameră suplimentară de înaltă calitate, concepută special pentru utilizarea cu Raspberry Pi. Această cameră are o rezoluție de 5 megapixeli și este capabilă să capteze imagini de înaltă calitate în lumina zilei sau în condiții de iluminare redusă.

Camera este dotată cu iluminare infraroșu pentru a permite capturarea imaginilor în întuneric. Iluminarea infraroșu se activează automat atunci când luminozitatea este insuficientă, permițând camerei să captureze imagini clare și detaliate chiar și în întuneric.

Camera se conectează la Raspberry Pi prin intermediul unui cablu cu bandă flexibilă și se poate monta cu ușurință pe placa Raspberry Pi. Camera este compatibilă cu majoritatea distribuțiilor de Linux și cu cele mai populare aplicații de fotografiere și video.

Această cameră poate fi utilizată pentru o varietate de proiecte, inclusiv monitorizarea de securitate a locuinței, captarea de imagini în timp real în condiții de iluminare redusă, captarea de imagini pentru proiecte de recunoaștere a imaginilor sau de învățare automată, și multe altele.[18]

Figura 2-4 Camera 5MP Night Vision pentru Raspberry Pi [18]

2.2.3 Senzor infraroșu cu întrerupere

IR break beam sensor, cunoscut în limba româna ca senzorul infra roșu cu întrerupere, este un dispozitiv utilizat pentru a detecta prezența sau absența obiectelor prin măsurarea modificărilor în cantitatea de lumină infraroșie detectată. Acest senzor este format dintr-un emițător de lumină infraroșie și un receptor.

Întreruperea razelor de lumină infraroșie înseamnă că un obiect a intrat în calea razei, pentru a anunța acest eveniment se trimite un semnal la placa de dezvoltare la care este conectat senzorul, de asemenea este sensibil la obiectele care se află la o distanță de până la 10 cm.

Figura 2-5 Senzor infraroșu cu detectarea întreruperi razei de lumină. [19]

Acest senzor poate fi utilizat într-o varietate de aplicații, cum ar fi sistemele de securitate, sistemele de control al accesului, jucării și multe altele. Senzorul poate fi integrat ușor în proiecte DIY (Do-It-Yourself), prin intermediul unei plăci de dezvoltare Arduino sau Raspberry Pi.

În ceea ce privește construcția, senzorul este format dintr-un LED infraroșu, un fototranzistor și un comparator. Senzorul este alimentat de la o sursă de 5V și are un consum redus de energie, făcându-l ideal pentru a fi utilizat în aplicații mobile. [3]

2.2.4 Recunoaștere facială

Recunoașterea facială este un domeniu activ și în continuă dezvoltare în cercetarea științifică. În prezent, exista numeroși algoritmi si tehnici utilizate pentru recunoaștere facială. Acestea au avantaje și limitări specifice, câteva dintre aceste avantaje ar fi identificarea rapidă, unicitatea biometrică și potențialul pentru identificare în timp real.

Prin recunoaștere facială persoanele pot fi identificate și autentificate fără necesitatea contactului fizic. Astfel poate fi utilizată într-un număr mare de aplicații și dispozitive, de la telefoane și calculatoare personale la sisteme de acces securizate și sisteme de monitorizare. Toate acestea fiind garantate de unicitatea biometrică a fiecărei persoane (deși există și excepții).

Limitările acestei tehnologii sunt aduse de sensibilitatea la variațiile de mediu, biometrică superficială și erorile de identificare și autentificare cât si de eventualele limitări morale cum sunt

protecția datelor, confidențialitatea și aspectele legale. Performanțele unui astfel de sistem pot fi influențate sau afectate de variații de iluminare, calitatea imaginilor și unghiul în care acestea sunt captate, iar dacă recunoașterea facială este superficială aceasta poate prezenta erori de neidentificare sau de identificare greșită și poate fi înșelată prin fotografii sau măști. De asemenea apar discuții despre stocarea datelor personale colectate de aceste sisteme și de folosirea lor în mod neautorizat și necorespunzător, ridicând probleme referitoare la supravegherea în masa și drepturile individuale. Probleme de acest tip sunt discutate astăzi în mass media având ca exemplu sistemul de credite sociale din China care are la bază algoritmi de recunoaștere facială și care a fost lansat oficial încă din 2014.

Beneficiile aduse de recunoașterea facială sunt extrem de utile și pot să ne îmbunătățească viața de zi cu zi, dar este important să nu pierdem din vedere limitările acesteia și să o implementăm ținând cont de normele legale și etice pentru a nu încălca drepturile și libertățile cetățenilor.

2.2.5 SMTP

SMTP (Simple Mail Transfer Protocol) este un protocol de comunicare standardizat dezvoltat în anii 1980 care este utilizat pentru livrarea și transmiterea de e-mailuri în rețeaua de internet, acesta este larg acceptat și utilizat în industria comunicațiilor. În acest domeniu există cercetări și inovații continue, dar nu asupra protocolului în sine, deoarece acesta este bine stabilit, înbunătățirile sunt aduse performanței, securității și eficienței serviciului de e-mail.[4]

3 DOCUMENTARE ASUPRA PROIECTULUI ALES

3.1 EVALUAREA SECURITĂȚII SMTP PE LINUX

Sistemul de operare Linux este considerat unul dintre cele mai sigure sisteme de operare de pe piață, chiar dacă anumiți critici argumentează că acesta ar putea fi o țintă pentru atacuri cibernetice din următoarele motive: este gratuit, open source și există multe metode și instrumente de hacking. În realitate, chiar natura sa open source îl face rezistent la astfel de atacuri deoarece problemele sunt rezolvate rapid, iar actualizările sunt disponibile pentru toți utilizatorii. În plus este mai puțin probabil ca virușii să se răspândească în rețea atunci când folosim Linux, deoarece în acest sistem de operare utilizatorii, în cele mai multe cazuri, nu dețin aplicațiile și nu pot scrie sau modifica executabilele.

Pentru a înțelege modul în care protocolul SMTP interacționează cu sistemul de operare Linux și implicit cu sistemul de operare Raspberry Pi OS, care este o distribuție a sistemului Debian care la rândul său este una dintre cele mai populare și respectate distribuții Linux, trebuie să luăm în considerare câteva informații de bază legate de acest protocol.

În primul rând, scopul SMTP este de a facilita transferul mesajelor e-mail între un client și serverul de e-mail al destinatarului. Mesajul este separat între antet și conținut, apoi se stabilește o conexiune între client și serverul expeditorului prin care mesajul este transferat cu ajutorul comenzilor și al răspunsurilor. SMTP are mai multe porturi pe care se poate realiza comunicarea, câteva dintre acestea fiind 25, 465 și 587.[4][20]

Portul 25 încă se folosește pentru comunicarea între servere, dar serviciile moderne de e-mail evită să folosească acest port deoarece furnizorii de servicii de internet și furnizorii de Cloud îl blochează pentru a reduce cantitatea de mesaje de spam.

Portul 465 a fost inițial un port criptat SMPTS (Secure) care utiliza SSL/TLS, dar IANA a atribuit un nou serviciu acestui port, așa ca acesta nu mai este utilizat pentru comunicații SMTP. În ziua de azi folosindu-se portul 587 ca port de bază pentru trimiterea e-mailurilor, acest port, însoțit de criptarea TLS asigură o transmitere securizată a mesajului, respectând ghidurile stabilite de IETF.[4][20]

Acest protocol utilizează diverse comenzi pentru gestionarea transferului de mesaje, mai jos sunt prezentate câteva comenzi de bază:

- HELO -> utilizată pentru a iniția o conexiune cu serverul;
- EHLO -> aceasta este o varianta extinsă a primei comenzi și se utilizează pentru a obține informații despre capacitățile serverului de e-mail;
- MAIL FROM -> necesară pentru a specifica adresa de e-mail a expeditorului;
- RCPT TO -> pentru specificarea adresei de e-mail a destinatarului;
- DATA -> utilizată pentru începerea transmisiei datelor;
- QUIT -> utilizată pentru încheierea sesiunii SMTP;

Autentificarea este o altă componentă importantă a acestui protocol, deoarece previne abuzurile și asigură securitatea. Aceasta se realizează prin două metode autentificarea cu parolă și autentificarea criptografică, cea din urmă utilizând certificate digitale și criptografie asimetrică pentru a verifica identitatea părților implicate în comunicare. De asemenea unele organizații utilizează servere SMTP de retransmisie (SMTP relay), acestea sunt folosite pentru a trimite mesaje către serverele de e-mail ale destinatarilor. Metoda asta este folosită în cazul în care serverul propriu nu are acces la rețeaua de internet sau este restricționat în trimiterea de e-mailuri către anumite domenii.[4][20]

O altă informație importantă legată de acest protocol este că de-a lungul timpului au apărut anumite extensii care au adăugat funcționalități suplimentare. Un bun exemplu este ESMTP (Extended SMTP) care a adăugat suport pentru autentificare, criptare, gestionarea erorilor și alte capabilități avansate. Alte câteva exemple relevante sunt AUTH (permite autentificarea utilizatorilor în timpul trimiterii e-mailului), SIZE (permite specificarea dimensiunii maxime a unui mesaj) și PIPELINING (permite trimiterea mai multor comenzi SMTP simultan), dar există și extensii specifice sau personalizate care pot fi implementate în funcție de anumite nevoi ale serverului. Toate acestea sunt doar câteva informații generale legate de SMTP ce au fost extrase din standardul RFC 5321.[4][20]

În cea ce privește implementarea și configurarea acestui protocol pe sisteme ca Linux și Raspberry Pi OS trebuie în primul rând să setăm parametrii și opțiunile specifice serverului ales, deoarece fiecare server de SMTP are propriile sale fișiere de configurare care pot fi editate. Trei astfel de servere sunt Postfix, Exim și Sendmail.

- Postfix este cunoscut pentru configurare și administrare ușoară, dar și pentru fiabilitate și securitate;
- Exim este cunoscut pentru configurarea sa flexibilă și capacitatea de a gestiona un volum mare de trafic al e-mailurilor;
- Sendmail este unul dintre primele servere SMTP disponibile, este puternic și flexibil, dar este mai dificil de învățat din punct de vedere al configurării și administrării;

Fișierele de configurare conțin setări generale ale serverului, cum ar fi metodele de autentificare, setările de securitate sau domeniile acceptate, de exemplu, în cazul serverului Postfix principalul fișier de configurare este “/etc/postfix/main.cf”. După modificarea fișierelor de configurare trebuie să reâncărcăm sau să repornim serverul SMTP pentru ca noile setări să se

aplice. De asemenea în sistemul de operare Linux trebuie să avem în considerare și noțiuni de securitate cum sunt autentificarea, autorizarea, criptarea, filtrarea și blocarea spamului.[4][20]

3.2 ALGORITMI UTILIZAȚI PENTRU RECUNOAȘTEREA FACIALĂ ÎN OPENCV

În principiu un program de recunoaștere facială este o aplicație software care identifică o persoană în baza unei fotografii sau a unui videoclip provenit de la o sursă prin intermediul unui algoritm de recunoaștere. O modalitate de identificare facială este prin compararea trăsăturilor feței cu o bază de date de imagini.

Librăria OpenCV a fost dezvoltată inițial în anul 1999 de către compania Intel, în momentul de față aceasta este o resursă open source și se află în continuă dezvoltare datorită contribuțiilor aduse de către diverși dezvoltatori și cercetători, dar și datorită contribuțiilor aduse de comunitatea de utilizatori. Există mai mulți algoritmi utilizați pentru recunoaștere facială în această librărie, dintre care cei mai utilizați și cunoscuți sunt:

- Haar Cascade;
- Eigenfaces;
- Fisherfaces;
- Local Binary Patterns Histograms;
- Histogram of Oriented Gradients;

3.2.1 Haar Cascade

Denumirea Haar Cascade provine de la “hair cascade” și definește o secvență de funcții care împreună formează o familie sau o bază de unde, aceste funcții sunt sub formă de pătrate. Algoritmul se bazează pe așa numitele ”Haar Wavelets”, care organizează pixelii din imagine în pătrate. Detectarea obiectelor de interes se face prin diferențierea intensității pixelilor. Antrenarea acestui algoritm se realizează prin două seturi de imagini, unul care conține obiectul de interes și unul care nu. Distingerea între obiectele dorite și fundalul imaginii se face în baza unui set de reguli pe care algoritmul și le-a format.[10]

Haar Cascade este utilizat în numeroase aplicații de recunoaștere facială și de detectare a obiectelor, devenind popular datorită eficienței sale în detectarea fețelor în timp real. În imaginea prezentată mai jos se poate observa modul în care pixelii din imagini sunt organizați.

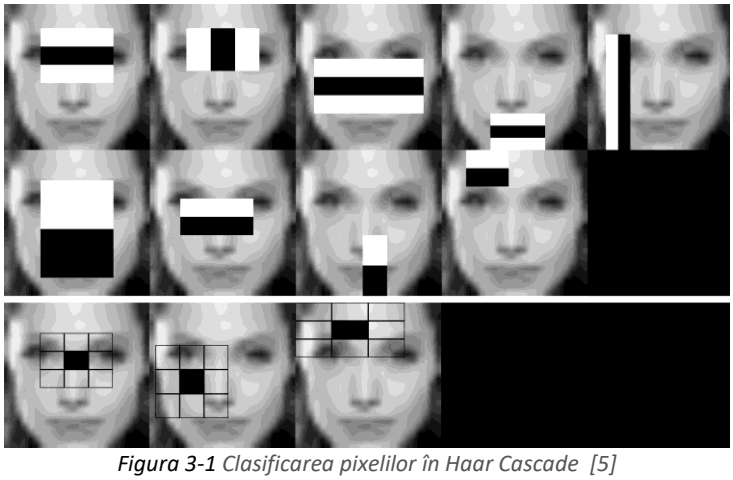


Figura 3-1 Clasificarea pixelilor în Haar Cascade [5]

În primele 4 imagini sunt arătate caracteristicile liniare, în următoarele 5 imagini caracteristicile marginale, iar în ultimele 3 sunt reprezentate caracteristicile orientate în jurul unui centru.[5]

3.2.2 Eigenfaces

Tehnica de recunoaștere facială Eigenfaces se bazează pe analizarea componentelor principale ale unei imagini. Pentru a realiza o astfel de analiză, în primul rând, se colectează un de imagini aliniate și de aceeași dimensiune ale fețelor diferitelor personae pe care vrem să le identificăm. Aceste imagini sunt prelucrate pentru a se elimina zgomotul și pentru a regla contrastul și iluminarea astfel încât imaginile să se încadreze în același standard, dar se pot aplica și tehnici de normalizare a fețelor pentru o mai bună aliniere și centrare a acestora.

Următorul pas este transformarea fiecărei imagini a unei fețe într-un vector ce conține caracteristicile respective sau într-un set de puncte de referință. Acestea sunt utilizate pentru construirea matricii de date, unde fiecare rând de date reprezintă un vector facial. Se analizează componentele matricei pentru a determina fețele-eigen. Fețele-eigen sunt vectori care capturează variația maximă în setul de date și sunt utilizate pentru a descrie fețele.

Pentru recunoașterea unei fețe algoritmul Eigenfaces compară imaginea cu fețele-eigen din setul de date și în baza vectorilor calculează distanțele intre componentele principale ale acestora, persoana cu cea mai mica distanță fiind recunoscută.

Acest algoritm facilitează recunoașterea facială în timp real deoarece reduce dimensiunea setului de date prin reprezentarea fețelor într-un spațiu cu dimensiuni minime, dar are vulnerabilități la variațile de iluminare și la schimbarea poziției feței, vulnerabilități care pot afecta performanța în diferite situații. [5][11]

3.2.3 Fisherfaces

Tehnica Fisherfaces utilizează LDA (Linear Discriminant Analysis) combinat cu Eigenfaces pentru a realiza o recunoaștere facială automată. LDA diferă de Eigenfaces prin faptul că reduce dimensionalitatea. Acesta identifică anumite direcții în spațiul caracteristicilor faciale care maximizează separabilitatea între diferitele clase de fețe și minimizează variația în cadrul fiecărei clase. Practic LDA încearcă să păstreze informațiile relevante pentru identificarea unei persoane prin discriminarea altor fețe ce nu conțin aceste informații. Fisherfaces constă în aplicarea LDA pe "fețele Eigen" existente, extrase deja de către algoritmul Eigenfaces, pentru a oferii o reprezentare compactă a caracteristicilor discriminante ale fețelor. [5][12]

3.2.4 LBP (Local Binary Patterns)

LBP este folosit pentru extragerea și reprezentarea unor caracteristici locale ale unei fețe. Toate regiunile locale sunt obținute în urma divizării imaginii faciale în regiuni mai mici cunoscute sub denumirea de ferestre sau blocuri. Metoda se bazează pe comparații locale între pixeli, constând defapt în evaloarea valorilor de intensitate ale pixelilor vecini față de cel central. Valorile sunt transformate în șiruri binare ale căror valori binare sunt comparate cu valorile binare ale șirurilor vecinilor, astfel se obține un șir de biți pentru pixelul central. Acest șir se numește model LBP. Cu ajutorul modelului LBP se construiește o histogramă corespunzătoare regiunii locale, ea reprezentând distribuția frecvențelor de apariție a diferitelor modele LBP în regiunea respectivă.

Următorul pas este concatenarea histogramelor pentru a se obține o reprezentare globală ale caracteristicilor LBP ale feței, iar pentru clasificare și recunoașterea fețelor în baza reprezentărilor LBP este nevoie să se utilizeze un algoritm de clasificare sau o rețea neuronală.

Tehnica de recunoaștere facială LBP este eficientă datorită robusteții sale la variații ale iluminării și expresii faciale, dar și pentru că poate captura textura feței, care poate fi folosită în combinație cu alte caracteristici pentru a se obține o reprezentare mai completă. [5][13]

3.2.5 HOG (Histogram of Oriented Gradients)

În principal HOG este folosit pentru detectarea obiectelor, a persoanelor și a fețelor, acesta fiind folosit în combinație cu alți algoritmi pentru realizarea procesului de recunoaștere facială. Principalele avantaje ale acestui descriptor față de alți descriptori este că are rezultate aproape perfecte în detectarea persoanelor, chiar dacă fundalul este complex, și rămâne constant chiar și în fața schimbărilor geometrice și fotometrice. Singurul dezavantaj major fiind faptul că este sensibil la schimbările de orientare ale obiectelor.

În acest proiect am utilizat algoritmul HOG în conceperea programului de antrenament pentru algoritmul de recunoaștere facială. Pe baza teoriei vectorilor, HOG este împărțit în trei etape. Prima etapă este analiza inițială a imaginii relevante, fața, care implică izolarea aproximativă a fundalului. Pentru realizarea acestui proces se ia în considerare doar magnitudinea vectorului, nu și orientarea acestuia, deoarece acest proces caută diferențele de magnitudine ale vectorilor din imagine. Pentru a calcula magnitudinea se poate utiliza ecuația 1.

$$m(u,v) = \sqrt{f_u(u,v)^2 + f_v(u,v)^2} \quad (1)$$

Din această ecuație matematică obținem magnitudinea *m* a unui vector de caracteristici în punctul *(u, v).* Vectorul *m(u, v)* este format din două componente, $f_u(u, v)$ orientată în direcția *u* și $f_v(u, v)$ orientată în direcția *v.* În figura 3-2 este prezentat un exemplu de histogramă aproximativă obținută în urma utilizării ecuației 1. În stânga este imaginea care va fi procesată, iar în dreapta histograma gradienților orientați.[9]

Figura 3-2 Exemplu de histogramă a gradienților [23]

După ce s-a făcut estimarea poziției obiectului în imagine se antrenează software-ul pentru a identifica mai ușor poziția obiectului. Acum se ia în considerare și direcția vectorului, care a fost omisă în etapa anterioară. Pentru a îndeplini această sarcină se folosește ecuația 2.[9]

$$\theta(u,v) = tan^{-1}\frac{f_v(u,v)}{f_u(u,v)} \qquad (2)$$

Ecuația 2 demonstrează că se poate utiliza arctangenta pentru a obține un unghi în conformitate cu componentele $f_u(u, v)$ și $f_v(u, v)$. Cum același punct va avea mai mulți vectori, ecuația 2 oferă o explicație detaliată a direcției vectorului, care este utilizată pentru a afișa punctul ca o variație continuă a gradientului în funcție de direcția vectorului. A treia etapă este construirea histogramelor de orientare, unde sunt utilizați gradienții calculați anterior. Întreaga imagine este împărțită în celule mici, pentru fiecare celulă calculându-se histograma de orientare a gradienților.

3.3 RECUNOAȘTERE FACIALĂ UTILIZÂND OPENCV ȘI PYTHON PE RASPBERRY PI

În timpul vieții un om întâlnește un număr semnificativ de persoane, iar capacitatea de a diferenția și recunoaște fețele acestor persoane pe care le întâlnim este una dintre cele mai importante și impresionante calități ale creierului uman, așa că dezvoltarea de sisteme capabile să realizeze un astfel de proces complex stârnește interes în comunitatea cercetătorilor și dezvoltatorilor din diverse domenii. Pentru ca o mașină să posede capacitatea de a identifica și localiza o persoană doar pe baza trăsăturilor faciale sunt necesare numeroase calcule matematice, deoarece fața este multidimensională. Prin urmare, scopul efortului colectiv este de a crea un sistem de recunoaștere a feței mai bun, care să fie util într-o varietate de contexte. Una dintre multele metode de implementare constă în utilizarea bibliotecii OpenCV pe un computer mono-circuit Raspberry Pi folosind limbajul de programare Python, pentru a realiza calculul de recunoaștere a imaginii. Acest sistem fiind bazat pe trei componente majore: un set de date ce conține fotografii cu fețele persoanelor care vrem să le identificăm, o secțiune de antrenare a programului și o secțiune care realizează procesul de recunoaștere.

Setul de date poate reprezenta atât un fișier ce conține imagini încărcate manual de către programator ale persoanei pe care vrem să o identificăm și este denumit cu numele persoanei respective, cât și un cod Python care activează camera și este programat în așa manieră încât să permită utilizatorului să captureze câteva imagini pe care programul să le salveze într-un fișier asemănător cu cel menționat mai sus. Un astfel de cod trebuie să deschidă o fereastră în care să se afișeze imaginile care sunt captate de cameră și să conțină comenzi care să-i permită utilizatorului să salveze imagini în diferite unghiuri, precum și o comandă pentru închiderea ferestrei.[6][7]

Secțiunea de antrenament are un rol fundamental deoarece formează cadrul în baza căruia se va face recunoașterea facială. Primul pas reprezintă convertirea imaginilor din RGB în nuanțe de gri, proces numit grayscaling, această transformare este necesară pentru că în general datele privind culorile unei imagini pot îngreuna procesul de înregistrare al unor detalii semnificative. În plus, convertirea imaginilor în nuanțe de gri îmbunătățește procesul de calcul al punctelor de interes făcându-l mai eficient și mai rapid prin omiterea datelor nesemnificative. După realizarea procesului de antrenare sistemul are informațiile necesare pentru a face comparații și distingeri între diferite imagini.[6][7]

Codul pentru realizarea procesului de recunoaștere facială preia imaginile care sunt înregistrate în timp real de cameră și le afișează într-o fereastră. Aceste imagini sunt comparate cu informațiile obținute în timpul procesului de antrenament iar dacă datele se potrivesc peste fața persoanei respective se va așeza o ramă pe care va scrie numele persoanei respective. În cazul în care aceasta nu este recunoscută în locul în care ar trebui să apară numele va apărea "unknown" sau orice alt mesaj a prevăzut programatorul în cod. Codul poate folosi orice algoritm de

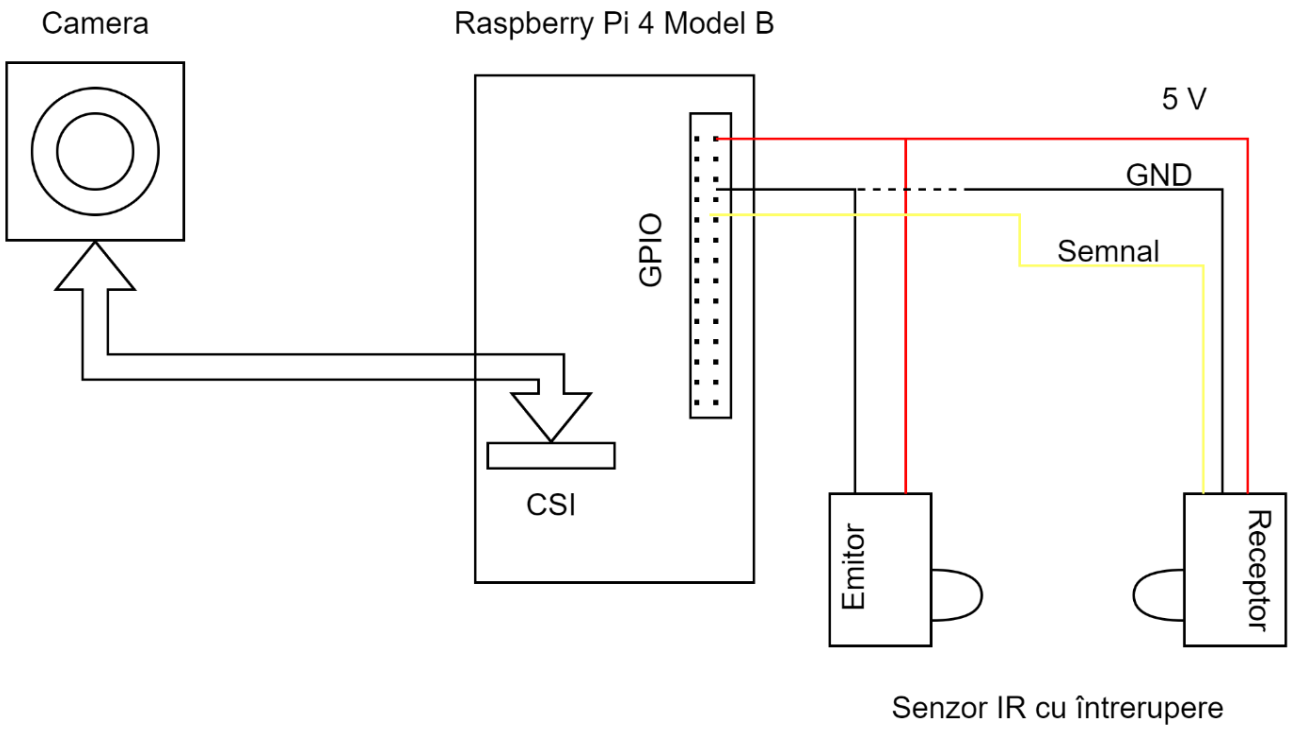
recunoaștere facială compatibil cu biblioteca OpenCV, dar în general modul acesta de implementare este des folosit și apare în multiple exemple disponibile online. [7][8]

4 ARHITECTURA HARDWARE

Arhitectura hardware descrie dispunerea și organizarea fizică a sistemelor informatice. Pentru ca un computer sau un alt dispozitiv electronic să funcționeze, diferite componente hardware trebuie să fie conectate și aranjate în moduri specifice. În acest capitol voi descrie capabilitățile și funcționarea componentelor hardware ale sistemului meu precum și modul în care acestea interacționează pentru îndeplinirea scopului proiectului.

Configurația hardware a unui sistem reprezintă fundația peste care se amplasează software-ul și este definitorie pentru limitele și posibilitățile acestuia. Un design eficient și ergonomic poate duce la o performanță superioară și o mai bună utilizare a resurselor.

În figura 4-1 este prezentată schema bloc a proiectului din punctul de vedere al configurației hardware.



Figură 4-1 Schema bloc a proiectului

Imaginea următoare (figura 4-2) este o reprezentare grafică a configurației hardware a sistemului, în ea putem observa conectarea senzorilor la alimentare și la ground, conectarea firului de semnal al senzorului la pinul GIPO17 precum și conectarea camerei la CSI (Camera Serial Interface).

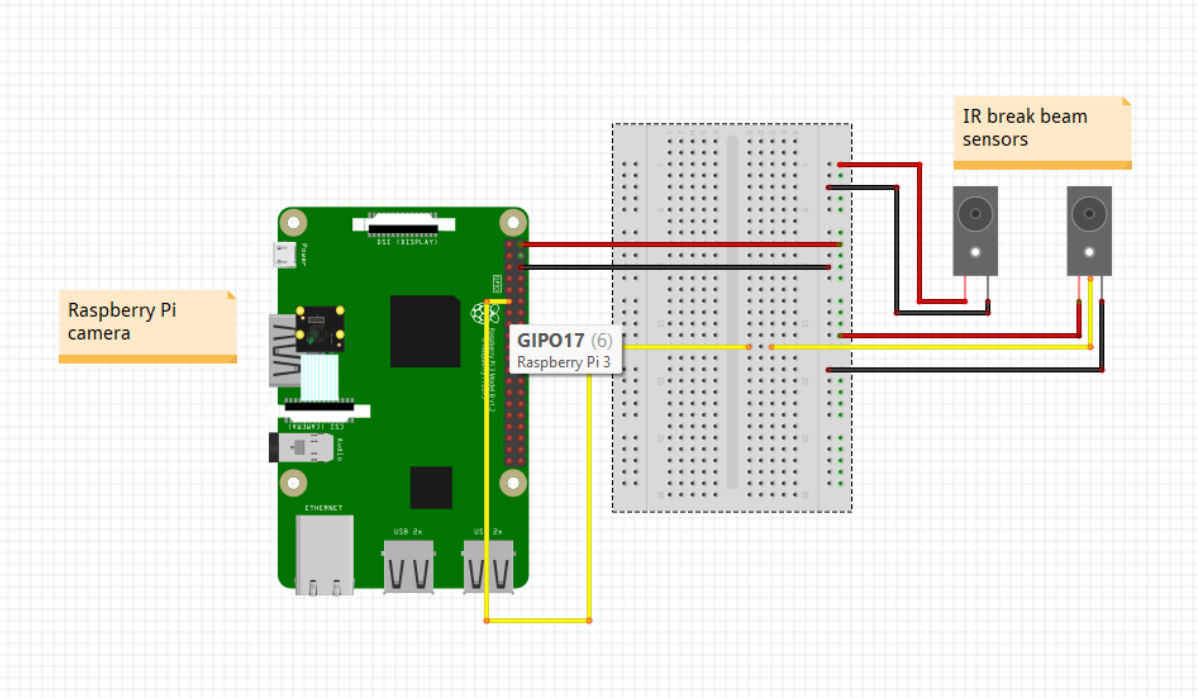


Figura 4-2 Reprezentare grafică a configurației hardware

4.1 SPECIFICAȚIILE COMPUTERULUI RASPBERRY PI 4 MODEL B

În acest proiect am folosit un Raspberry Pi 4 Model B, acesta este ultimul model din seria de computere mono-circuit Raspberry Pi și este un sistem pe un singur cip (SoC) Broadcom BCM2711 care încorporează un procesor ARM Cortex-A72 quad-core cu frecvența de 1,5 GHz ce îi oferă computerului o performanță și eficiență mai bună comparativ cu modelele anterioare.

Din punct de vedere al stocării principalul suport este cardul micro SD ce se inserează într-un slot specific, prezentat în figura 4-4, iar din punct de vedere al configurației de memorie RAM acest model beneficiază de 2GB LPDDR4 SDRAM cu performanțe bune și o lățime de bandă de memorie mare. Avantajele principale aduse de tehnologia LPDDR4 SDRAM sunt viteza mare de transfer a datelor și consumul redus de energie fără a compromite capacitatea de stocare.

În ceea ce ține de procesarea grafică dispunem de un GPU VideoCore VI integrat. GPU-ul suportă OpenGL ES 3.0 și are o capacitate de decodare video la rezoluția de 4K. În plus dispune de doi conectori micro-HDMI care permit ieșirea pe două ecrane la rezoluții de până la 4K, dar având în vedere că în acest proiect am folosit o cameră cu rezoluția de 5MP nu am putut exploata unitatea de procesare grafică la capacitatea ei maximă.

Această plăcuță este populară în cercurile programatorilor și dezvoltatorilor de aplicații, deoarece acest computer portabil oferă o bază puternică pentru o varietate de proiecte, motiv pentru care este apreciat atât de profesioniști, cât și de educatori și entuziaști.

În figura 4-3 sunt prezentate porturile și conectivitățile Raspberry Pi 4 Model B.

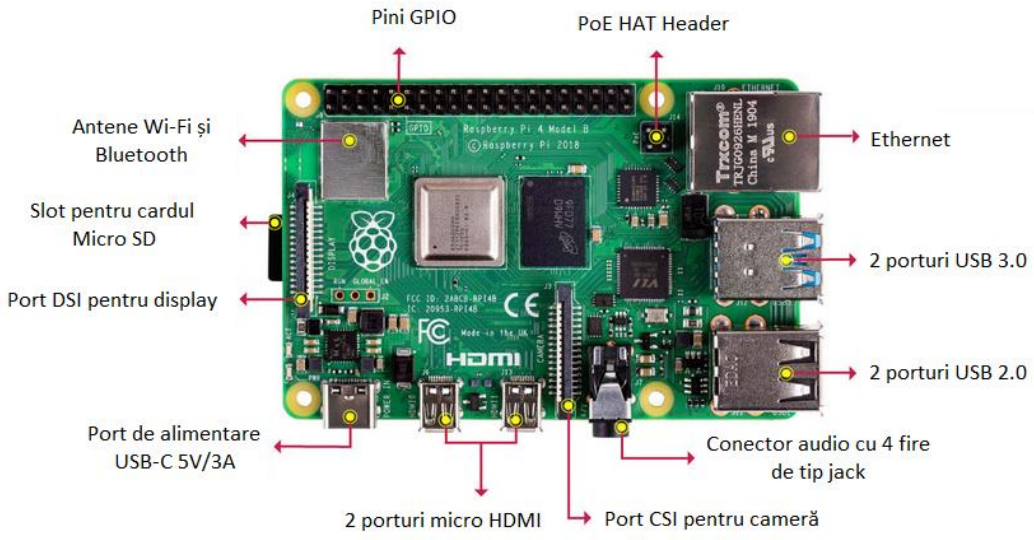


Figura 4-3 Porturile Raspberry Pi 4 Model B

Opțiunile de conectivitate ne oferă de asemenea numeroase avantaje pentru dezvoltarea de aplicații de rețea și IoT (Internet of Things) deoarece sistemul dispune de:

- Wi-Fi dual-band (2,5 GHz și 5 GHz) ;
- Bluetooth 5.0 ;
- Gigabit Ethernet ;
- USB 2.0 și USB 3.0 ;

Conexiunea cu dispozitivele electronice externe se realizează prin intermediul GPIO (General Purpose Input/Output) care dispune de 40 de pini și suportă protocoale precum I2C, SPI, UART și altele.

Raspberry Pi 4 poate rula mai multe tipuri de sisteme de operare, în principal sunt folosite cele oficial (Raspberry Pi OS) sau versiuni de Linux cum sunt Ubuntu și Fedora, dar există și sisteme de operare personalizate folosite pentru aplicații de nișă, care pot opera pe Raspberry fără probleme.

Relevante pentru acest sub capitol sunt și specificațiile microprocesorului ARM Cortex-A72 a cărui nucleu de înaltă performanță a fost creat de ARM Holdings. Acesta a fost dezvoltat pe baza predecesorului său, ARMv8-A, și este utilizat pe scară largă într-o serie de sisteme integrate.

Cortex-A72 poate rula atât software pe 32 de biți, cât și pe 64 de biți. Spre deosebire de predecesorul sau, Cortex-A57 , A72 este mai eficient și optimizat pentru calcule de înaltă performanță. Datorită capacităților sale superioare de execuție în afara ordinii, dimensiunii mai

mari a cache-ului și a resurselor mai mari în general, acesta îmbunătățește performanța sistemului în care lucrează atunci când se confruntă cu aplicații solicitante.[24]

Microprocesorul poate fi implementat în mai multe configurații de nuclee, cel implementat pe acest model de Raspberry Pi este un quad core, cu viteză de procesare crescută și o procesare paralelă mai buna, față de un model cu un singur nucleu sau un dual core. Design-ul Cortex-A72 are un nivel ridicat de pipe-line care permite procesarea concomitentă a mai multor instrucțiuni în diferite etape de pipe-line. Prin urmare, avem posibilitatea unui randament mai mare și o mai bună utilizare a resurselor.[24]

Construcția sa îl face eficient din punct de vedere energetic și în același timp păstrează capabilități excelente de performanță. Acest lucru se face prin gestionarea avansată a energiei, scalarea dinamică a tensiunii și frecvenței, plus menținerea unui consum redus de energie atunci când se află în stare de inactivitate.[24]

Tabelul 4-1 Specificații ARM Cortex-A72 [24]

Categorie	Specificații
ARM ISA (Arhitectura setului de instrucțiuni)	32/64-biți
Nr. maxim de instrucțiuni (pe ciclu de ceas)	8 ops
Maximum Pipeline Length (nr. maxim de etape)	20 etape
Integer Pipeline Length (etape prin care trece o instrucțiune de tip întreg)	14 etape
Penalizarea greșelii de prezicere a ramurii	15 cicluri
Integer Add	2
Integer Mul	1
Unități de încadrare/stocare (Load/Store)	1 +1 (Dedicated L/S)
Unități de branch	1
Floating-Point/NEON Arithmetic Logic Units	2x128-biți
L1 Cache	48KB I$ + 32KB D$
L2 Cache	512KB – 4MB

Specificațiile prezentate în tabelul 4-1 reprezintă mare parte din capabilitățile microprocesorului ARM Cortex-A72, care este prezent pe computerul mono-circuit Raspberry Pi 4 folosit în acest proiect.

4.2 CONECTORUL ȘI INTERFAȚA CSI

Modulul de cameră pentru Raspberry Pi poate transfera datele prin intermediul unui bus extrem de rapid (CSI) direct către procesor. Această operațiune este realizată prin intermediul unui cablu cu 15 pini, cunoscut sub numele de cablu panglică sau cablu flexibil, care se conectează la un soclu, numit ZIF 15, montat pe suprafața plăcii. Benzile de date de pe magistrală oferă o lățime de banda de aproximativ 2 Gbps, ceea ce reprezintă o rezoluție aproximativă de 5 MP, permițând înregistrarea de video-uri la rezoluția de 1920 pixeli × 1080 pixeli cu 30 de cadre pe secundă.

Specificațiile acestei interfețe descriu stratul fizic (D-PHY2), iar schema de semnalizare a acestuia este cunoscuta sub numele de semnalizare diferențială de joasă tensiune. Sistemul fiind destinat aplicațiilor de joasă tensiune (1,2 V), care permit o viteză de transfer limitată practic la 1 Gbps pe o bandă, dar cum modelul conceput pentru Raspberry Pi este prevăzut cu două benzi se ajunge la viteza de transfer de 2 Gbps. În realitate, rata de transfer de date poate varia mult și depinde de calitatea interconexiunilor, așadar valorile rămân teoretice și aproximative.[21]

MIPI (Mobile Industry Processor Interface) este un set de standarde creat de o organizație internațională care se ocupă cu dezvoltarea de interfețe și protocoale în industria electronica, în special pentru dispozitivele mobile. MIPI CSI este una dintre cele mai cunoscute și utilizate interfețe în domeniul camerelor, oferind o conectivitate eficientă între componente.

În tabelul 4-2 am enumerat pini de conectare cu numele și scopurile lor respective.

Tabelul 4-2 Conectarea pinilor pentru interfața CSI [22]

Pin	Nume	Scop
1	Ground	Ground
2	CAM1_DN0	Data Lane 0
3	CAM1_DP0	
4	Ground	Ground
5	CAM1_DN1	Data Lane 1
6	CAM1_DP1	
7	Ground	Ground
8	CAM1_CN	MIPI Clock
9	CAM1_CP	
10	Ground	Ground
11	CAM_GPIO0	
12	CAM_GPIO1	
13	SCL0	I^2C Bus
14	SDA0	
15	+3.3 V	Alimentare

4.3 SENZORUL IR BREAK BEAM

În acest proiect am ales să folosesc un senzor infraroșu cu detectarea întreruperii razei de lumină deoarece este o metodă simplă de a detecta mișcarea. Principiul de funcționare este simplu, avem două capete unul care are rolul de emițător și unul care are rolul de receptor. Raza de lumină proiectată de emițător este o radiație în infraroșu, o radiație electromagnetică cu lungimea de undă mai mare decât cea a luminii vizibile, prin urmare este invizibilă, dar nu suficient de mare pentru a fi periculoasă omului. Emițătorul este compus din două fire, cel roșu pentru alimentare respectiv cel negru pentru împământare și are rolul de a trimite fasciculul de lumină,

comparativ cu acesta receptorul are trei fire și este sensibil la lumina transmisă de emitor. Al treilea fir al receptorului , de culoare galbenă, are rolul de a trimite semnalul care anunță întreruperea razei de lumină către microcontrolerul pe care este montat.

Spre deosebire de senzorii PIR, detectarea întreruperii razei este mai rapidă, permite un control mal bun al zonei în care dorim să detectăm mișcarea și este mai ieftină comparativ cu folosirea unui modul de tip sonar. Dar are și dezavantaje, cum sunt distanța de transmisie limitată și necesitatea de a fi poziționați față în față, într-un unghi în care emițătorul să-și proiecteze raza astfel încât să nu depășească zona de sensibilitate a receptorului.

Senzorul pe care îl folosesc în acest proiect are două versiuni, una cu LED de 3 mm și una cu LED de 5 mm. Am ales varianta cu 5 mm deoarece are o rază de funcționare mai mare, până la 50 de cm, în timp ce varianta de 3 mm are o rază de funcționare de doar 25 de cm. Alimentarea se poate realiza la 3,3 V sau la 5 V , am decis sa-l alimentez la 5 V pentru a obține o rază de acțiune mai bună. Important de menționat este și faptul că ieșirea receptorului este un tranzistor cu colector deschis. Acest fapt determină nevoia unei rezistențe de pull-up dacă dorim să citim semnal digital de la senzor, pentru a face asta se poate conecta un rezistor de 10 KOhm între firul roșu de alimentare al receptorului și firul galben de semnal. Însă am decis să nu folosesc această modalitate deoarece Raspberry Pi are încorporată capacitatea de a activa o astfel de rezistență de pull-up.

4.4 TESTAREA COMPONENTELOR

Înainte de a implementa sistemul pe care mi-am propus să-l realizez primul obiectiv a fost să testez componentele hardware. Testarea este un proces de evaluare și verificare, în acest caz, al unei componente hardware, pentru a identifica posibile defecțiuni sau erori și a le remedia înainte de a începe dezvoltarea proiectului. Pentru a testa componentele folosite am utilizat câteva coduri simple în Python, care să-mi arate că piesele lucrează în parametri normali. Precum și niște noțiuni de bază legate de Raspberry Pi, plus un program de diagnoză pentru cardul micro SD.

4.4.1 Testarea Raspberry Pi 4 Model B

Am început testarea plăcuței printr-o simplă verificare vizuală a acesteia, pentru a vedea dacă lipsesc componente. După ce am confirmat integritatea structurală a Raspberry Pi-ului, am introdus cardul micro SD pe care se află sistemul de operare și am alimentat plăcuța. Am folosit un adaptor micro HDMI pentru a conecta Raspberry Pi-ul la un monitor și astfel am confirmat atât funcționarea sistemului de operare cât și a interfeței de ieșire video. Următorul pas a fost să verific celelalte interfețe de comunicare USB, Ethernet, Wi-Fi și Bluetooth, toate s-au comportat normal. Pentru verificarea memoriei și a procesorului m-am folosit de aplicațiile htop și Raspberry Pi Diagnostics. În figura 4-4 se poate observa că CPU-ul, memoria RAM și cardul SD funcționează în parametri normali.

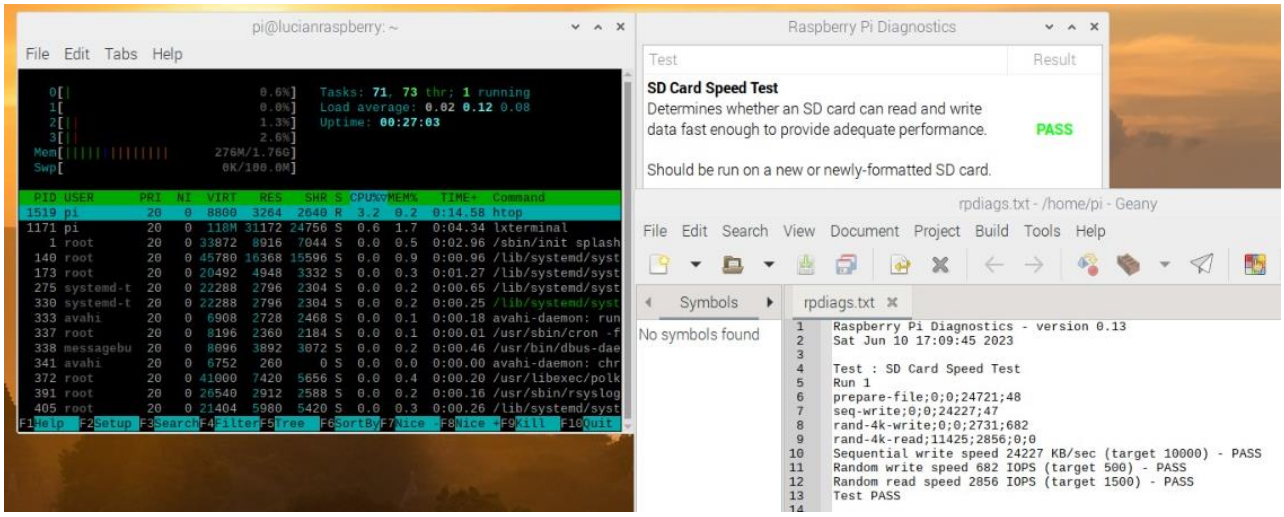


Figura 4-4 Parametri resurselor de sistem

4.4.2 Testarea senzorului infraroșu

Senzorul infraroșu cu detectarea întreruperii razei de lumină l-am testat folosind un cod simplu în Python. Scopul acestui cod este să afișeze un mesaj în terminal care să specifice starea în care se află senzorul.

```
import RPi.GPIO as GPIO
# Definirea pinului
BEAM_PIN = 17
# Funcția pentru schimbarea stării pinului
def break_beam_callback(channel):
    if GPIO.input(BEAM_PIN):
        print("beam unbroken")
    else:
        print("beam broken")

GPIO.setmode(GPIO.BCM)
# Configurarea pinului BEAM_PIN ca intrare, cu rezistența de pull-up activată
GPIO.setup(BEAM_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
# Detecția evenimentului pentru schimbarea stării
GPIO.add_event_detect(BEAM_PIN, GPIO.BOTH, callback=break_beam_callback)

message = input("Press enter to quit\n\n")
GPIO.cleanup()
```

Codul 4-1 Cod pentru testarea senzorului

La trecerea unui obiect prin fața senzorului a starea pinului BEAM_PIN se modifică și afișează mesajul ”beam broken”, iar când obiectul dispare se afișează mesajul ”beam unbroken”. Rezultatul afișat în terminal se poate observa în figura 4-5.

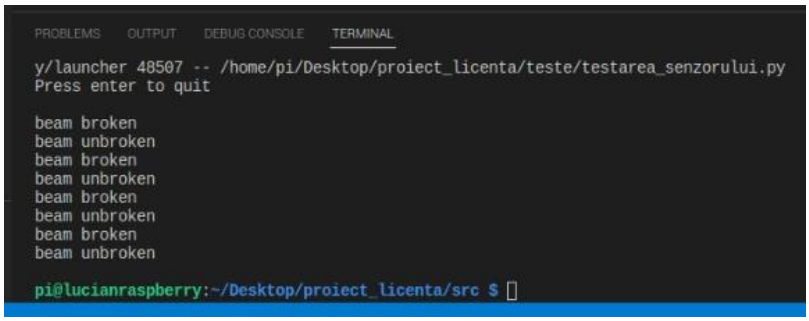


Figura 4-5 Rezultatul din terminal pentru codul de testare al senzorului

4.4.3 Testarea camerei

Pentru a testa camera am folosit de asemenea un cod simplu în Python și am utilizat bibliotecile OpenCV și imutilis, pentru a afișa imaginile capturate de cameră într-o fereastră.

```
import cv2
import imutils

# Instanțe pentru capturarea video
cap = cv2.VideoCapture(0)

while True:
    # Citirea unui frame din captură
    ret, frame = cap.read()


    # Verificarea dacă citirea frame-ului a fost reușită
    if not ret:
        print("Eroare la citirea frame-ului")
        break

    # Afișarea frame-ului într-o fereastră
    frame = imutils.resize(frame, width=500)
    rotated_frame = cv2.rotate(frame, cv2.ROTATE_180)
    cv2.imshow("Camera", rotated_frame)

    # Designarea tastei 'q' pentru încheierea programului
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Eliberarea resurselor și închiderea ferestrelor
cap.release()
cv2.destroyAllWindows()
```

Codul 4-2 Cod pentru testarea camerei

Datorită modului în care am camera poziționată pe macheta proiectului aceasta filmează invers, cu susul în jos, pentru a rezolva această problemă am folosit funcția cv2.rotate pentru a rotii imaginea la 180 de grade. În figura 4-6 este prezentată fereastra deschisă de codul prezentat anterior.

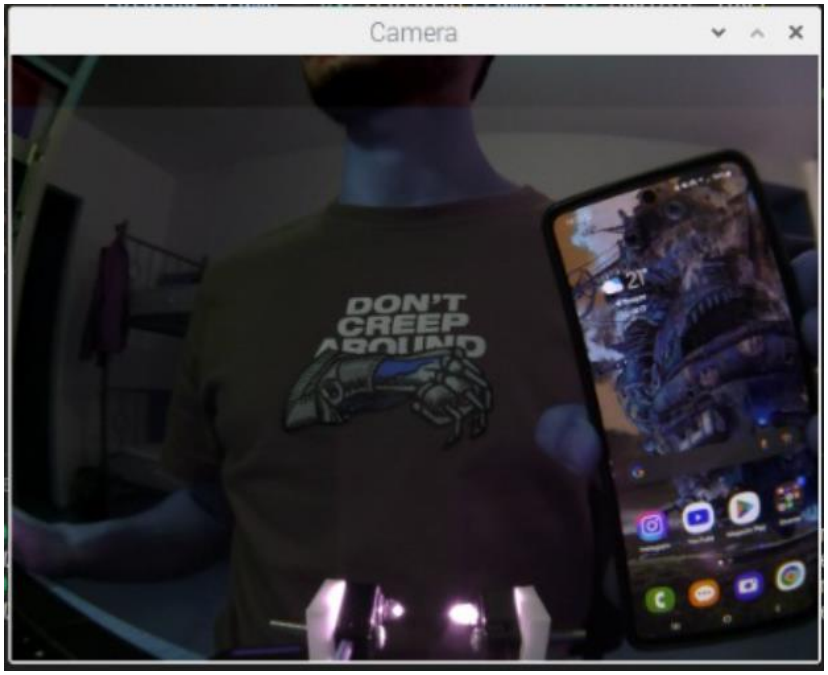


Figura 4-6 Fereastra deschisă de codul de testare al camerei

4.5 IMPLEMENTAREA HARDWARE

Implementarea hardware a proiectului a fost relativ simplă, constând doar în conectarea camerei la portul CSI și în conectarea senzorului la alimentare, masă și la pinul GPIO 17 pentru transmiterea semnalului digital către procesor. Folosirea unui breadboard nu era neapărat necesară, deoarece există suficienți pini de alimentare și masă, dar am preferat să nu ocup acești pini în mod inutil.

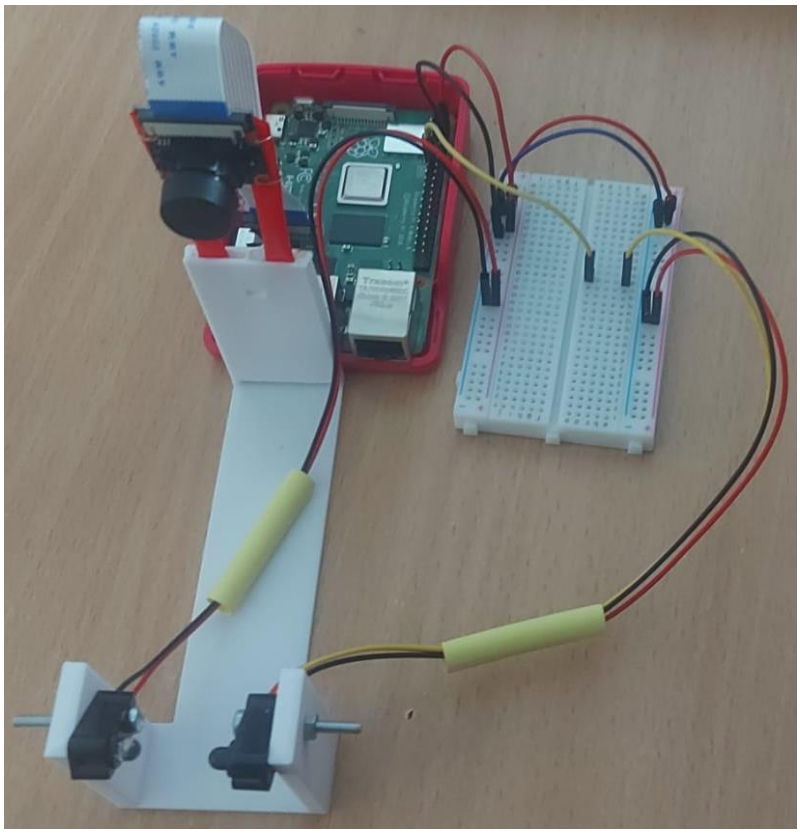


Figura 4-7 Implementarea hardware a proiectului

5 ARHITECTURA SOFTWARE

Întreaga dispunere și organizare a unui sistem software se numește arhitectură software. Scopul acesteia este de a descrie conexiunile dintre elementele unui program și modul în care acestea lucrează împreună pentru a îndeplini un anumit scop, pentru a produce funcționalitatea aplicației. Arhitectura software se ocupă cu aspecte precum gestionarea datelor, performanța, comunicarea, securitatea, fiabilitatea și altele, toate acestea reprezentând planul de creare, întreținere și dezvoltare a programului.

Responsabilitatea software-ului este de a face alegeri structurale care determină modul în care diversele componente ale sistemului, fie ele hardware sau software, interacționează și cooperează între ele pentru îndeplinirea unui anumit scop.

În acest proiect am realizat un sistem de securitate cu recunoaștere facială care să se activeze prin detectarea mișcării la pătrunderea intrusului în zona de interes pe care acest sistem o protejează. Funcționalitatea acestui program poate fi explicată printr-o diagramă logică.

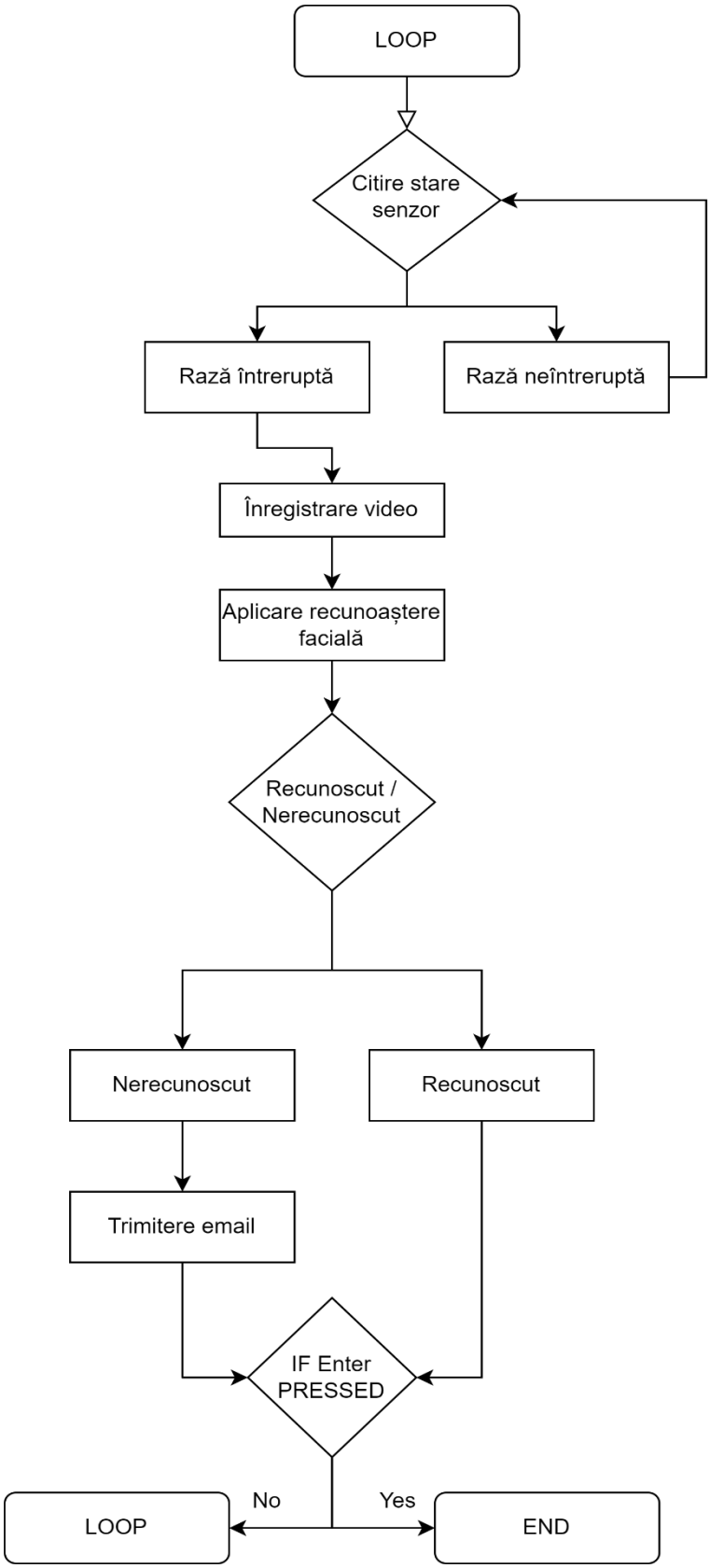


Figura 5-1 Diagrama logică a programului

În diagrama prezentată anterior am încercat să prezint funcționalitatea programului folosit pentru acest proiect. O dată ce sistemul este activat acesta așteaptă să primească semnalul de întrerupere a razei de la senzor, semnalul funcționează ca un declanșator pentru cameră. După ce camera a fost activată videoclipul pe care aceasta îl înregistrează este salvat într-un fișier. Următorul pas pe care programul îl parcurge este să preia acest videoclip și să-l proceseze cu ajutorul algoritmului de recunoaștere facială.

Secvența din cod care se ocupă cu recunoașterea facială realizează acest proces cu ajutorul datelor obținute prin programul de antrenament. În urma comparării datelor se pot obține două rezultate, fiecare afectând diferit parcurgerea următoarelor lini de cod.

Dacă persoana din înregistrare este recunoscută de algoritm sistemul decide că nu există niciun pericol pentru locația pe care o protejează, așa că se întoarce la starea sa inițială în care așteaptă să primească semnalul de activare de la senzor.

În cazul în care algoritmul nu recunoaște persoana din imaginile preluate de cameră, decide că locația a fost invadată și apelează programul “mail.py”. Acest program are datoria de a prelua videoclipul ce conține imaginile cu persoana neidentificată și de a concepe un e-mail destinat proprietarului, care să conțină videoclipul în care se află intrusul și înștiințarea că securitatea locației a fost compromisă. După îndeplinirea misiunii, la fel ca în primul caz prezentat, sistemul se întoarce în starea de așteptare, pregătit să se activeze iar la primirea unui nou semnal de la senzor.

5.1 CAPTURAREA IMAGINILOR PENTRU ANTRENAREA ALGORITMULUI

Pentru a putea antrena algoritmul de recunoaștere facială avem nevoie de un set de informații în baza cărora să putem efectua comparații. Informațiile necesare sunt extrase dintr-un fișier cu imagini încărcate în prealabil. Pentru crearea unui astfel de fișier putem utiliza două metode. Prima metodă este extrem de simplă și constă în încărcarea manuală a pozelor în directorul respectiv. A doua metodă constă în crearea unui cod în limbajul de programare Python care să manipuleze camera pentru a captura și stoca poze cu persoana pe care dorim ca algoritmul să o recunoască în fișierul din care algoritmul de antrenament își va extrage imaginile.

Prima parte a codului are scopul de a stabilii poziția camerei, rezoluția la care este făcută poza și framerate-ul, precum și definirea unui counter care să numere câte capturi sunt făcute.

```
cam = PiCamera()
cam.rotation = 180
cam.resolution = (512, 304)
cam.framerate = 10
rawCapture = PiRGBArray(cam, size=(512, 304))
img_counter = 0
```

Codul 5-1 Cod pentru setarea capturii

Următoarea secvență din cod deschide o buclă while ce are condiția true, deci o buclă infinită care poate fi închisă doar de către utilizator la apăsarea unei taste definite în cod, care activează comanda break.

```
while True:
    for frame in cam.capture_continuous(rawCapture, format="bgr", use_video_port=True):
        image = frame.array
        cv2.imshow("Press Space to take a photo", image)
        rawCapture.truncate(0)
```

Codul 5-2 Cod pentru deschiderea ferestrei de vizualizare

Această primă parte a secvenței deschide o fereastră în care utilizatorul poate să vadă imaginile care sunt preluate în timp real de camera, scopul acestei ferestre de vizualizare este să ajute utilizatorul să captureze imagini ale feței sale în unghiurile pe care acesta și le dorește.

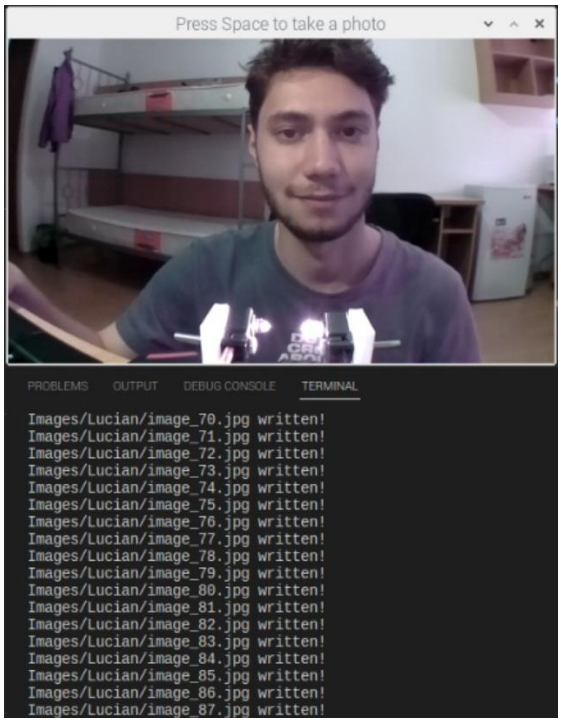


Figura 5-2 Fereastra deschisă pentru efectuarea de capturi

A doua parte a secvenței definește tasta SPACE pentru efectuarea de capturi și tasta q pentru închiderea ferestrei, de asemenea salvează imaginile în fișierul specificat și le numește în funcție de valoarea counter-ului.

```
k = cv2.waitKey(1)
        rawCapture.truncate(0)
        if k == ord('q'): # q pressed
            break
        elif k == 32:
            # SPACE pressed
            img_name = "Images/"+ name +"/image_{}.jpg".format(img_counter)
            cv2.imwrite(img_name, image)
            print("{} written!".format(img_name))
            img_counter += 1

    if k == ord('q'):
        print("q hit, closing...")
        break
```

Codul 5-3 Cod pentru capturarea imaginii și închiderea ferestrei

5.2 ANTRENAREA ALGORITMULUI

Această parte a proiectului este extrem de importantă pentru algoritmul de recunoaștere facială, deoarece programul de antrenament extrage din imaginile capturate de codul prezentat anterior datele necesare algoritmului pentru compararea imaginilor.

Utilizând modulul face_recognition din Python programul găsește și examinează fețele dintr-o colecție de fotografi pentru a extrage trăsăturile faciale sub formă de codificări. Codificări pe care le salvează într-un fișier numit “encodings.pickle”. Din acest fișier algoritmul de identificare facială își va extrage datele. Pentru a extrage codificările se folosesc câteva funcții, acestea sunt prezentate în secvența de mai jos din codul 5-4.

```
boxes = face_recognition.face_locations(rgb, model="hog") # detectarea locațiilor fețelor
encodings = face_recognition.face_encodings(rgb, boxes)   # extrage codificările fețelor
knownEncodings.extend(encodings)                          # lista codificărilor extrase
knownNames.extend([name] * len(encodings))                # adăugarea numelui
```

Codul 5-4 Cod pentru extragerea codificărilor

În figura 5-3 este prezentat un exemplu de codificare extrasă dintr-o imagine. După cum se poate observa datele sunt colectate într-o matrice cu 32 de rânduri și 4 coloane, fiecare element al matricei reprezentând câte o caracteristică sau trăsătură a feței prezentate în imagine.

```
Encoding for image_43.jpg:
[-1.72970012e-01  3.97321470e-02  6.24977909e-02 -5.49131185e-02
 -1.99997708e-01 -3.24790105e-02 -1.43982843e-02 -5.55847846e-02
  1.37369305e-01 -4.35434468e-02  2.42114589e-01 -6.48507029e-02
 -2.36658365e-01 -3.72820795e-02 -5.85849546e-02  1.03359029e-01
 -5.46045825e-02 -8.45270008e-02 -7.26068988e-02 -1.18294790e-01
  6.49584681e-02  1.05041333e-01 -1.50065729e-02  3.68686393e-02
 -6.78955764e-02 -3.75700653e-01 -1.43718496e-01 -6.71573281e-02
  8.31674263e-02 -1.09781608e-01  2.21909210e-02  6.32698014e-02
 -1.13860130e-01 -1.09608002e-01  1.96018443e-03  6.86299801e-02
 -1.33500010e-01 -9.25178677e-02  2.18779206e-01  1.93663351e-02
 -1.89318225e-01 -4.84391395e-03  7.71745443e-02  2.95079976e-01
  8.96249488e-02  4.34609130e-02  5.31374924e-02 -7.12689832e-02
  1.11580208e-01 -2.68467546e-01  4.52049002e-02  1.82663426e-01
  1.60384014e-01  3.09415460e-02  1.33585408e-01 -1.14425384e-01
  5.15383333e-02  1.55383527e-01 -2.07026720e-01  1.03992619e-01
  7.24406019e-02 -7.18023479e-02 -2.26820521e-02 -7.29773268e-02
  1.74912170e-01  8.31668377e-02 -6.82971179e-02 -1.17809504e-01
  1.71747595e-01 -1.55855775e-01 -8.11152756e-02  1.29420534e-01
 -9.70780700e-02 -2.48912603e-01 -2.87658691e-01  1.27771944e-01
  4.08377379e-01  1.47731885e-01 -1.39050081e-01  2.96995938e-02
  1.56607665e-02 -2.11510845e-02  9.51167867e-02  5.95659800e-02
 -1.48513749e-01 -3.20775285e-02 -1.43780127e-01  1.08317100e-02
  1.81064397e-01 -4.34750728e-02 -3.03259352e-04  2.49478430e-01
  9.77207869e-02  1.04115389e-01  7.28707388e-02  9.56703126e-02
 -9.32892337e-02 -5.34914620e-03 -9.20402259e-02  7.06422478e-02
  1.42392084e-01 -1.21375494e-01  1.20411329e-02  6.22744113e-02
 -1.44996196e-01  1.68581232e-01 -4.09368277e-02 -5.26115857e-02
 -7.53153563e-02 -8.02219883e-02 -1.02616049e-01  1.99574847e-02
  1.82789475e-01 -2.33740047e-01  1.44570321e-01  1.75608978e-01
  1.00093922e-02  1.75943285e-01  1.37518868e-01  8.71229544e-03
 -4.68667597e-04 -9.52517018e-02 -1.59431636e-01 -7.34101981e-02
  8.60151500e-02 -9.16480497e-02 -3.83815058e-02  4.18971665e-02]
```

Figura 5-3 Exemplu de codificare extrasă

Pentru obținerea acestor seturi de caracteristici am folosit face_recognition din OpenCV și implicit algoritmul HOG (Histogram of Oriented Gradients) pentru detectarea regiunilor feței prin identificarea variațiilor de intensitate și de orientare a muchiilor.

5.3 ALARMA SILENȚIOASĂ PRIN UTILIZAREA PROTOCOLULUI SMTP

Pentru a crește eficiența sistemului meu de securitate am implementat și un program de declanșare al unei alarme silențioase. O alarmă silențioasă diferă de una normală prin faptul ca nu produce zgomote puternice și se rezumă la metode mai discrete de alertă. O metodă de acest fel este notificarea proprietarului prin trimiterea unui e-mail către acesta.

Am îndeplinit această sarcină prin utilizarea protocolului SMTP, în baza căruia am realizat un cod în limbajul de programare Python care să utilizeze o adresă de Gmail, creată special pentru această aplicație, pentru a trimite un mesaj proprietarului în cazul în care un intrus a pătruns în locația ce se dorește a fi protejată. Pentru a fi mai ușor de prezentat am împărțit codul în șase secvențe distincte, fiecare cu atribuțiile sale.

1. Prima secvență este simpla, constă în doar două lini de cod și are rolul de a configura detaliile de autentificare ale protocolului.

```
smtp_server = 'smtp.gmail.com'
smtp_port = 587
```

Codul 5-5 Cod pentru stabilirea serverului și portului

2. A doua secvență are rolul de a obține cel mai recent fișier video din directorul în care sistemul salvează filmarea făcută la declanșarea senzorului infraroșu cu întrerupere.

```
video_files = glob.glob(os.path.join(video_directory, '*.mp4'))
latest_video = max(video_files, key=os.path.getctime)
```

Codul 5-6 Cod pentru obținerea fișierului video

3. A treia secvență creează un obiect care să conțină mesajul pe care vrem să-l transmitem.

```
msg = MIMEMultipart()
msg['From'] = sender_email
msg['To'] = receiver_email
msg['Subject'] = '👾 INVADER 👾'
```

Codul 5-7 Cod pentru crearea mesajului

4. A patra secvență deschide fișierul video și îl atașează mesajului.

```
with open(latest_video, 'rb') as f:
    attachment = MIMEBase('application', 'octet-stream')
    attachment.set_payload(f.read())
    encoders.encode_base64(attachment)
    attachment.add_header('Content-Disposition', f'attachment; filename={latest_video}')
    msg.attach(attachment)
```

Codul 5-8 Cod pentru atașarea fișierului video la mesaj

5. A cincea secvență are scopul de a inițializa conexiunea SMTP și de a trimite mesajul.

```
try:
    server = smtplib.SMTP(smtp_server, smtp_port)
    server.starttls()
    server.login(sender_email, sender_password)
    server.sendmail(sender_email, receiver_email, msg.as_string())
    print('E-mail sent successfully!')
except Exception as e:
    print(f'Error occurred while sending e-mail: {str(e)}')
finally:
    server.quit()
```

Codul 5-9 Cod pentru inițializarea conexiunii SMTP

6. A șasea și ultima secvență de cod conține datele expeditorului, adresa de e-mail a persoanei care trebuie să primească mesajul și calea către directorul în care se află fișierul video.

```
sender_email = 'lucianraspberry07@gmail.com'
sender_password = '****************'
receiver_email = 'neagulucian1889@gmail.com'
video_directory = '/home/pi/Desktop/proiect_licenta/video'
```

Codul 5-10 Cod pentru adresele e-mail

Aceste bucăți de cod lucrează împreună pentru a îndeplini scopul fișierului “mail.py”, ele generând mesajul ce va conține avertizarea si captura video o intrusului. Mesaj ce va fi trimis la adresa de e-mail a proprietarului cu ajutorul protocolului SMTP și prin intermediul serverelor Google.

5.4 PROGRAMUL PRINCIPAL

Fișierul “main_cod.py” reprezintă coloana vertebrală a sistemului, aici sunt folosite datele obținute de primele două coduri prezentate în acest capitol, camera, senzorul și codul prezentat în subcapitolul 5.3 pentru transmiterea mesajului către proprietar.

În acest program se utilizează limbajul Python pe Raspberry Pi pentru a efectua recunoașterea facială, integrată în acest sistem, cu ajutorul camerei și al bibliotecii OpenCV. Pentru a realiza complet procesul de recunoaștere facială se folosește algoritmul Haar Cascade, inclus în biblioteca OpenCV, în baza unor caracteristici specifice acest algoritm identifică în imagini zonele care pot fi considerate fețe. După ce acestea au fost detectate se folosește un algoritm de codificare a fețelor existent în biblioteca face_recognition din OpenCV, care calculează un vector de codificare pentru fiecare față detectată, acest lucru le face mai ușor de comparat.

Compararea se face între aceste codificări obținute din imaginile înregistrate de cameră și datele din fișierul “encodings.pickle”, unde se află codificările vectoriale obținute din imaginile prelucrate în procesul de antrenament. În funcție de cât de similare sunt datele comparate sistemul returnează o valoare, true (adevărat) dacă imaginile sunt similare sau false (fals) dacă imaginile nu se potrivesc. În continuare voi prezenta modul de funcționare al programului, evidențiind secvențe de cod relevante pentru funcționalitățile pe care le explic.

Primul lucru pe care l-am făcut a fost să import modulele necesare, acestea fiind modulele pentru manipularea imaginilor, recunoaștere facială, manipularea camerei, controlul GPIO și altele. După care am definit pinul 17 pentru a detecta starea senzorului și o funcție de 'callback' ce este apelată la schimbarea stării senzorului. În funcția 'callback' am implementat o condiție care comandă camerei să filmeze dacă raza este întreruptă.

```
else:
        print("Beam broken")
        with PiCamera() as camera:
            camera.rotation = 180
            camera.start_preview()
            timestamp = strftime("%Y-%m-%d-%H-%M-%S")
            video_file = f'/home/pi/Desktop/proiect_licenta/video/video_{timestamp}.h264'
            camera.start_recording(video_file)
            sleep(5)
            camera.stop_recording()
            camera.stop_preview()
        process_video(video_file)
```

Codul 5-11 Cod pentru manipularea camerei

Următoarea funcție are rolul de a converti videoclipul filmat în format mp4 pentru a fi mai ușor de manipulat și pentru a putea fi accesat de pe orice dispozitiv.

```
def convert_to_mp4(video_file):
    output_file = video_file.replace('.h264', '.mp4')
    command = ['MP4Box', '-add', video_file, output_file]
    subprocess.run(command)
    print(f"Video saved as {output_file}")
```

Codul 5-12 Cod pentru convertirea videoclipului

Această secvență este urmată de o funcție care șterge primul videoclip înregistrat, deoarece este inutil sa ocupăm memoria cu două clipuri identice, dar în formate diferite.

În funcția 'process_video' se realizează procesul de comparare a datelor pentru a se efectua identificarea facială asupra imaginilor captate, în următoarea secvență se vor stabilii valorile necesare pentru procesarea videoclipului și se vor încărca codificările și detectorul de fețe.

```
def process_video(video_file):
    currentname = "unknown"
    encodingsP = "encodings.pickle"
    print("[INFO] loading encodings + face detector...")
    data = pickle.loads(open(encodingsP, "rb").read())
```

Codul 5-13 Cod pentru convertirea videoclipului

Codul de mai sus este urmat de o buclă while care va identifica și afișa fețele recunoscute pe fiecare cadru din videoclip. În prima parte a buclei se vor citi cadre succesive și se va aplica algoritmul de recunoaștere facială.

```
while not video_ended:
        ret, frame = vs.read()
        if not ret:
            video_ended = True
            break
        frame = imutils.resize(frame, width=500)
        boxes = face_recognition.face_locations(frame)
        encodings = face_recognition.face_encodings(frame, boxes)
        names = []
```

Codul 5-14 Cod pentru citirea cadrelor și aplicarea algoritmului de recunoaștere

În a doua parte a buclei apare o buclă for care este responsabilă cu asocierea numelor fețelor pentru fiecare cadru detectat în prima parte a buclei while.

```
for encoding in encodings:
            matches = face_recognition.compare_faces(data["encodings"], encoding)
            name = "Unknown"
            if True in matches:
                matchedIdxs = [i for (i, b) in enumerate(matches) if b]
                counts = {}
                for i in matchedIdxs:
                    name = data["names"][i]
                    counts[name] = counts.get(name, 0) + 1
                name = max(counts, key=counts.get)
                if currentname != name:
                    currentname = name
                    print(currentname)
            names.append(name)
```

Codul 5-15 Cod pentru asocierea numelor

Ultima parte a buclei se ocupă cu afișarea imaginilor într-o fereastră, astfel încât dacă persoana este recunoscută în jurul feței acestea va apărea un pătrat de culoare galbenă cu numele persoanei respective, iar daca persoana nu este recunoscută va apărea mesajul “Unknown” în locul numelui.

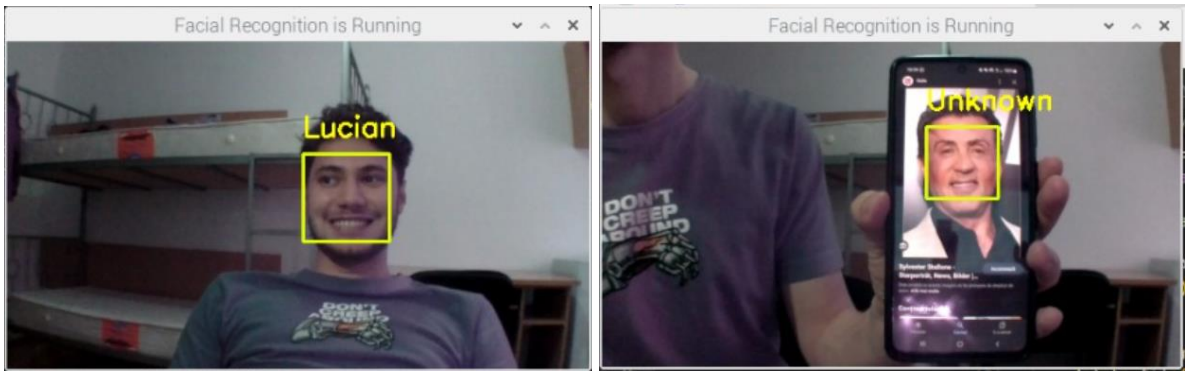


Figura 5-4 Exemplu de recunoaștere/nerecunoaștere a unei persoane

După ce toate aceste procese s-au terminat se rulează o structură de control if a cărei argument presupune ca numele curent, numele afișat deasupra pătratului, să fie diferit de numele înregistrat pentru codificările făcute în timpul procesării codului de antrenament, numele proprietarului. Dacă condiția este îndeplinită se execută funcția ’send_video_mail’ care apelează fișierul “mail.py” pentru a trimite înștiințarea de pătrundere neautorizată proprietarului.

La finalul executării programului se va afișa în consolă mesajul “E-mail sent successfully!” în cazul în care nu este întâmpinată nicio problemă pe parcursul parcurgerii codului. Din acel moment până la primirea notificării e-mail vor trece doar câteva minute în funcție de latența serverelor Google. În figura 5-5 se prezintă o astfel de notificare pe e-mail.

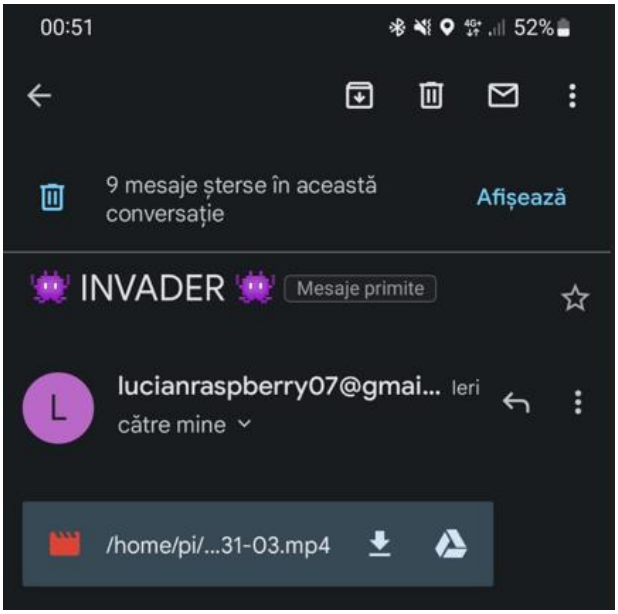


Figura 5-5 Exemplu de notificare de avertizare primită pe e-mail

5.5 MĂSURAREA EFICIENȚEI SISTEMULUI

Măsurarea eficienței sistemului este un proces ce are ca scop evaluarea performanței acestuia pentru a determina dacă resursele sunt utilizate eficient și dacă sistemul își îndeplinește funcțiile și obiectivele în parametri optimi.

Eficiența unui sistem poate fi determinată în mai multe moduri, având în vedere natura sistemului respectiv, dar în general eficiența poate fi raportată la valoarea resurselor consumate și a gradului de satisfacție pe care aplicația îl aduce. În acest capitol am încercat să măsor eficiența raportându-mă la utilizarea resurselor, eficiența în timp a programului și la calitatea procesului de recunoaștere facială în funcție de diverși factori.

În funcție de scopul unui sistem este important să se identifice indicatori relevanți, care să fie măsurați corect, pentru a se realiza o reprezentare clară a performanței acestuia. Rezultatele obținute pot ajuta dezvoltatorii unei aplicații sau a unui sistem să determine posibilități de îmbunătățire pentru versiunile următoare.

Următoarele trei subcapitole reprezintă teste, măsurători și comparații făcute pentru a estima eficiența sistemului meu.

5.5.1 Eficiența în timp a programului

Eficiența în timp a unui program constă în măsurarea timpului necesar de execuție al programului, de la parcurgerea primei lini de cod, până la îndeplinirea scopului acestuia. Se poate realiza o astfel de măsurătoare asupra programului meu prin utilizarea unui cod în Python.

```
import time

# Timpul de start
start_time = time.time()

# ... Codul programului de executat ...

# Timpul de oprire
end_time = time.time()

# Calcularea duratei de execuție
execution_time = end_time - start_time

print("Durata de executie a programului: ", execution_time, " secunde.")
```

Codul 5-16 Secvență de cod pentru calcularea timpului de execuție

În codul 5-16 am prezentat o secvență de cod ce poate fi încorporată în programul principal al sistemului meu pentru a măsura timpul de execuție al acestuia. Modul de funcționare al acestei secvențe este simplu și constă în folosirea modulului 'time' din Python. Variabilele start_time și end_time înregistrează timpul de la începutul executării programului, respectiv de la sfârșitul executării programului, iar apoi se calculează diferența acestora și este reținută în variabila execution_time.

Tabelul 5-1 Timpul de execuție în cazul recunoașterii

Nr. crt.	Timp de execuție (secunde)	Durată medie (secunde)
Execuție 1	108.429	116.272
Execuție 2	108.030	
Execuție 3	98.713	
Execuție 4	112.919	
Execuție 5	107.158	
Execuție 6	183.275	
Execuție 7	117.665	
Execuție 8	133.734	
Execuție 9	132.118	
Execuție 10	111.922	
Execuție 11	115.735	
Execuție 12	109.125	
Execuție 13	115.185	
Execuție 14	112.324	
Execuție 15	107.959	
Execuție 16	108.778	
Execuție 17	108.610	
Execuție 18	120.610	
Execuție 19	106.237	
Execuție 20	106.922	

Cu ajutorul acestor linii de cod implementate în programul meu, am realizat o serie de experimente pentru a determina care este durata medie de execuție a acestuia. Datele empirice obținute în urma acestor experimente sunt trecute în două tabele, tabelul 5-1 și tabelul 5-2.

În tabelul 5-1 am prezentat timpul de rulare și media acestuia la 20 de execuții cu rezultatul în secunde. Scopul experimentului a fost să determine care este durata medie de execuție a programului în cazul în care persoana înregistrată de camera de supraveghere este recunoscută.

Tabelul 5-2 Timpul de execuție în cazul nerecunoașterii

Nr. crt.	Timp de execuție (secunde)	Durată medie (secunde)
Execuție 1	141.619	143.183
Execuție 2	142.875	
Execuție 3	111.024	
Execuție 4	109.031	
Execuție 5	101.489	
Execuție 6	139.166	
Execuție 7	175.667	
Execuție 8	202.757	
Execuție 9	201.239	
Execuție 10	144.921	
Execuție 11	141.778	
Execuție 12	139.177	
Execuție 13	150.580	
Execuție 14	159.548	
Execuție 15	113.830	
Execuție 16	136.658	
Execuție 17	141.621	
Execuție 18	139.200	
Execuție 19	140.325	
Execuție 20	131.168	

În tabelul 5-2 am înregistrat datele unui experiment similar cu cel realizat pentru datele prezentate în tabelul 5-1, singura diferență fiind că în tabelul 5-2 s-a determinat durata medie de execuție a programului în cazul în care persoana din videoclipul captat de camera de supraveghere nu este recunoscută, iar în figura 5-6 este reprezentat un grafic al valorilor din tabelul 5-2.

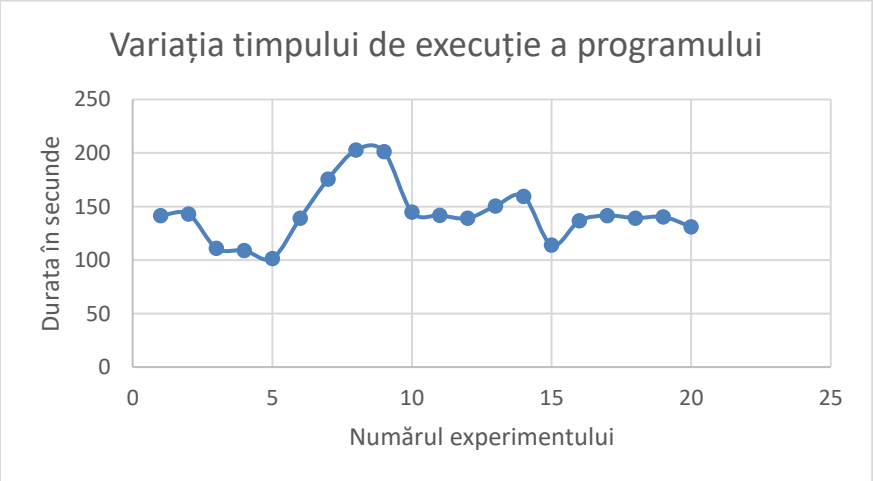


***Figura 5-6** Variația timpului de execuție al programului (persoana nerecunoscută)*

Această acțiune durează mai mult timp, deoarece în cazul în care persoana nu este recunoscută se rulează și codul pentru trimiterea e-mail-ului de avertizare. Datele obținute în urma măsurătorilor făcute, mai ales cele din tabelul 5-2, sunt importante pentru estimarea timpului în care avertizarea ajunge la proprietar.

Momentul în care destinatarul primește e-mail-ul de avertizare de la sistemul de securitate nu poate fi calculat cu exactitate, deoarece chiar dacă am realizat o medie a timpului de procesare pe care programul îl necesită de la primirea semnalului de la senzor până la trimiterea e-mail-ului, între momentul în care e-mail-ul s-a trimis și momentul în care destinatarul îl primește pot intervenii diverși factori care să întârzie livrarea. Printre aspectele care pot influența intervalul de timp necesar de la câteva secunde, la câteva minute se numără:

- Latența rețelei;
- Verificarea și filtrarea antispam;
- Configurarea și performanța serverelor;
- Probleme tehnice;

Toate aceste aspecte enumerate anterior influențează timpul de transmisie în moduri diferite. Latența rețelei, înțeleasă ca timpul necesar de care au nevoie pachetele pentru a ajunge de la expeditor la serverul Gmail și apoi la destinatar poate varia în funcție de distanța fizică sau de congestia rețelei. Verificarea antispam poate adăuga și ea o mică întârziere, dar în general este aproape instantanee.

Pe de altă parte, configurarea și performanța serverului de e-mail pot aduce întârzieri în funcție de performanța efectivă a serverului, în majoritatea cazurilor serverele sunt optimizate pentru a fi rapide, iar problemele tehnice, care pot afecta cel mai mult, sunt destul de rare și sunt rezolvate în general rapid în funcție de gravitatea acestora.

În ultimul rând, cel mai influent factor este conexiunea la internet. Aceasta trebuie să fie rapidă și stabilă, cel puțin în cazul sistemului care expediază mesajul de avertizare, deoarece acesta este plasat într-un punct fix, iar în cazul destinatarului calitatea accesului la rețeaua de internet poate varia din cauza caracterului mobil al acestuia, dar este de preferat să fie cât mai consecventă, pentru o transmisie reușită.

5.5.2 Utilizarea resurselor

Practica de urmărire și evaluare a diferitelor resurse de sistem consumate de o anumită aplicație sau proces informatic este cunoscută sub numele de monitorizare a resurselor. Aceasta presupune urmărirea și examinarea unor indicatori precum CPU, RAM și alte caracteristici relevante. Scopul este înțelegerea modului în care este utilizat sistemul sau aplicația, cât de performantă sau de eficientă este, precum și pentru a detecta orice posibilă neregulă, care poate provoca probleme viitoare.

Pentru a monitoriza o resursă în mod eficient, datele trebuie să fie colectate în timp, fie periodic, fie în timp real. Prezentarea lor trebuie să fie făcută atât într-un mod semnificativ, cât și utilizabil. Cu ajutorul datelor dobândite se pot face decizii pertinente pentru optimizări și îmbunătățiri ale aplicației, deoarece acestea ne oferă informații despre tendințe și tipare ale sistemului. Se pot utiliza diverse instrumente pentru monitorizarea resurselor, fie ele integrate sau externe, care oferă capacitatea de a vizualiza parametri resurselor. În acest proiect, pentru această sarcină, am ales să folosesc comanda htop.

Comanda htop este o unealtă folosită în linia de comandă, terminal, a sistemelor Unix și Linux pentru monitorizarea resurselor. Aceasta oferă o vizualizare interactivă și detaliată a proceselor care rulează pe sistem. Interfața htop conține atât o listă de procese organizată într-un

mod intuitiv, cât și grafice în timp real pentru vizualizarea utilizării CPU și a memoriei. Fiecare proces din listă este reprezentat printr-o linie care oferă informații despre PID (identificatorul procesului), utilizare CPU și memorie, stare, timp de execuție și alte atribute.

În figura 5-7 este reprezentată o captură de ecran a datelor înregistrate de comanda htop în timpul procesului de recunoaștere facială. Interpretarea graficului ce reprezintă nucleele procesorului și memoria se face în funcție de bara de progres a fiecărei categorii și de culorile acesteia. Astfel încât fiecare culoare reprezintă un anumit proces, iar frecvența de apariție reprezintă procentajul pe care acel proces îl ocupă din utilizarea resursei respective.

Utilizarea memoriei și a spațiului de swap este reprezentată prin trei culori:

- Verde, procentul de RAM consumat de paginile de memorie;
- Albastru, procentul de RAM consumat de memoria tampon (buffer);
- Portocaliu, procentul de RAM consumat de paginile cache;

În timp ce utilizarea CPU este reprezentată de șapte culori:

- Albastru, procentul din CPU utilizat de procesele cu prioritate scăzută;
- Verde, procentul din CPU utilizat de procesele deținute de utilizatorii normali;
- Roșu, procentul din CPU utilizat de procesele sistemului;
- Portocaliu, procentul din CPU utilizat de IRQ time;
- Magenta, procentul din CPU consumat de Soft IRQ time;
- Gri, procentul din CPU consumat de timpul de așteptare IO;
- Cyan, procentul din CPU consumat de timpul de Steal;

IRQ time reprezintă timpul folosit pentru tratarea întreruperilor de către un sistem de calcul, iar Soft IRQ time reprezintă timpul folosit pentru tratarea întreruperilor software generate de kernel-ul sistemului de operare.

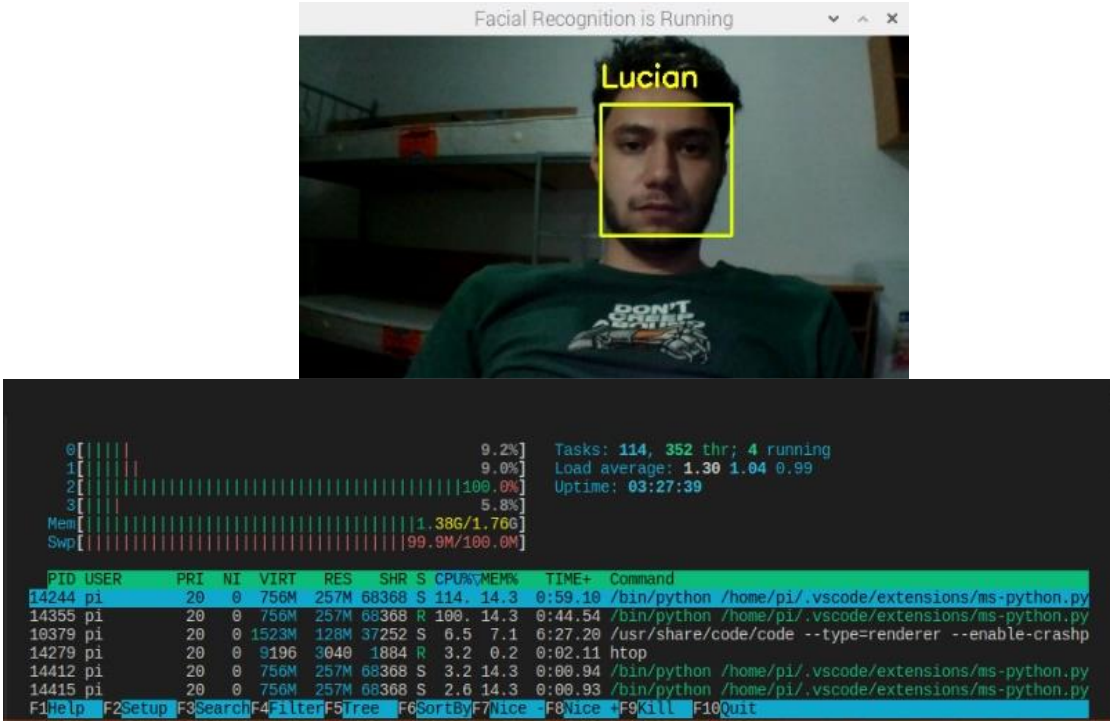


Figura 5-7 Utilizarea resurselor în timpul procesului de recunoaștere facială

După cum se poate observa în figura 5-7 nucleele sunt numerotate de la 0 la 3, iar nucleul numărul 2 este folosit destul de intens în timpul procesului de recunoaștere facială. Segmentele de culoare verde și roșu din bara de progres indicând faptul că nucleul este ocupat cu executarea de procese ale aplicației și ale sistemului. În cea ce ține de memoria RAM și de memoria de swap și acestea sunt solicitate intens în timpul procesului de recunoaștere facială, memoria de swap fiind utilizată la capacitate maximă, iar memoria RAM în procent aproximativ de 78,4%.

Tot în figura 5-7 sunt afișate și numărul de task-uri (sarcini) ale programului, acesta reprezintă totalitatea proceselor deschise, important de menționat este faptul că nu toate procesele deschise consumă constant resurse. Pentru cele 114 task-uri avem 352 de thread-uri și 4 procese care utilizează resursele CPU în mod activ, un thread reprezentând o secvență de instrucțiuni care poate fi executată independent în cadrul unui proces.

5.5.3 Calitatea procesului de recunoaștere facială

Performanța unui sistem ce are incorporat un algoritm de recunoaștere facială poate fi evaluată pe baza mai multor criterii, inclusiv acuratețea, viteza, robustețea și scalabilitatea.

Capacitatea sistemului de a recunoaște și de a potrivi corect fețele este denumită acuratețe. Astfel încât calitatea unui proces de recunoaștere facială poate fi definită în funcție de numărul de fețe pe care le identifică corect și numărul de fețe pe care le atribuie în mod eronat. De asemenea, mai ales în cazul aplicațiilor în timp real, rapiditatea de realizare a acestui proces este extrem de importantă, iar din punct de vedere al robusteții un sistem puternic de recunoaștere facială trebuie să funcționeze eficient într-o varietate de situații, inclusiv în variații de iluminare, poziție, expresii faciale și ocluzii, cum ar fi parul facial sau purtatul de ochelari. Diferiți algoritmi, implementări și seturi de date pot produce niveluri variate de precizie a recunoașterii faciale.

În figura 5-8 sunt prezentate patru imagini diferite și felul în care algoritmul de recunoaștere facială a prestat în fiecare situație.

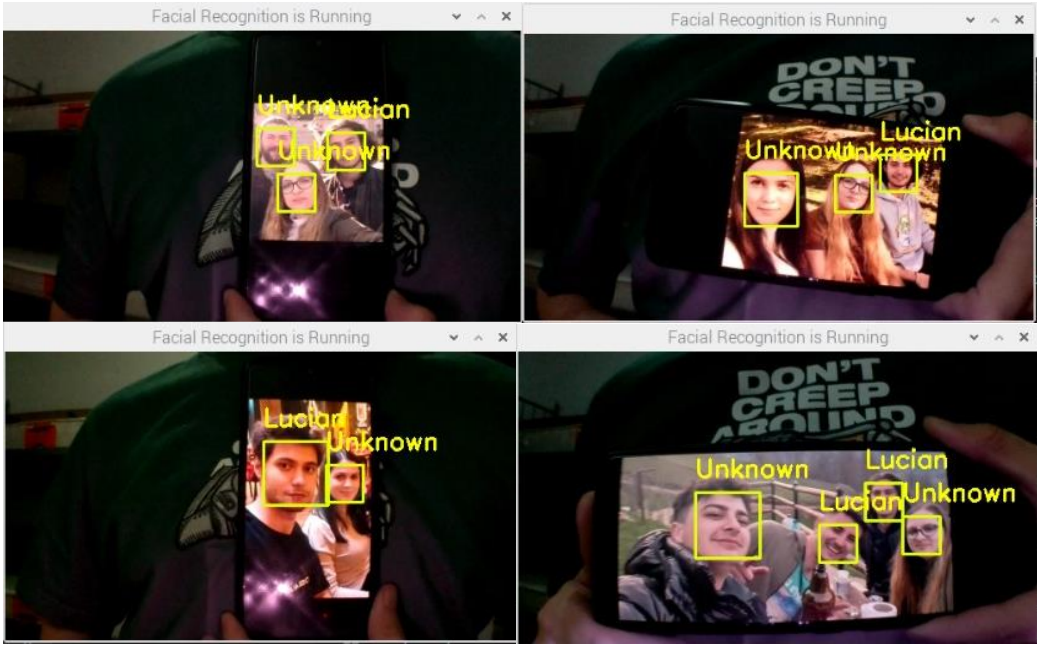


Figura 5-8 Calitatea procesului de recunoaștere facială

După cum se poate observa în imaginile din figura 5-8 am testat sistemul cu diferite imagini în care mă aflu alături de colegi și prieteni de la facultate. În procesarea acestor imagini algoritmul a reușit să mă identifice de fiecare dată, chiar dacă fotografiile în care apar sunt puțin diferite de cele folosite în antrenarea algoritmului. Sistemul de recunoaștere facială m-a identificat atât în imaginile în care aveam păr lung sau scurt cât și în cele în care purtam barbă sau nu, fără ca mediile diferite în care au fost realizate fotografiile să afecteze calitatea procesului.

În ultima fotografie, cea din dreapta jos, se poate observă că algoritmul a confundat unul dintre prietenii mei cu mine. Cea ce arată că sistemul pe care l-am implementat nu este perfect și poate efectua confuzii în cazul în care o altă persoană are trăsături faciale similare cu ale mele sau unghiul și poziționarea feței scanate creează erori de calcul la preluarea și compararea datelor.

De-a lungul testelor pe care le-am efectuat în timpul procesului de dezvoltare și de testare a sistemului, am observat comportamentul acestuia la variații ale iluminării, modificări ale expresiei faciale și a poziției capului. Din observațiile mele, am tras concluzia că algoritmul a avut performanțe bune, reușind întotdeauna să mă recunoască indiferent de modificările menționate anterior.

În figura 5-9 am reprezentat procesul de recunoaștere facială executat asupra unei imagini de joasă calitate și cu iluminare redusă.

Figura 5-9 Exemplu de imagine cu rezoluție și iluminare slabă

Chiar și în aceste condiții extreme algoritmul reușește să examineze fața din imagine și să extragă pe care le codifică în vectori și le compară, realizând cu succes procesul de recunoaștere facială .

Singurul dezavantaj major al programului apare în cazul în care orientarea imaginilor se schimbă. Această situație reprezintă, de fapt, unul dintre dezavantajele algoritmului Haar Cascade, pe care l-am prezentat în subcapitolul 3.2 (paragraful 3.2.1). Datorită faptului că modelul Haar Cascade este antrenat cu ajutorul unui set de date care conține imagini cu fețe în poziții standard, verticale, detectarea fețelor orientate cu susul în jos este dificilă.

În figura 5-10 este prezentată o imagine capturată după ce am schimbat orientarea camerei la 180 de grade.

Figura 5-10 Schimbarea orientării camerei

În această situație algoritmul este dezorientat, nu poate să izoleze punctele de interes din imagine și nu reușește nici măcar să-mi detecteze fața, realizarea unei identificări fiind imposibilă pentru programul meu în cazul de față.

6 IMPLEMENTĂRI VIITOARE

Dezvoltarea continuă a proiectului, îmbunătățirea funcționalităților și a performanțelor prin includerea de noi caracteristici lasă o ușă deschisă pentru ca proiectul să își atingă potențialul maxim. Implementările viitoare pot aduce numeroase oportunități proiectului prin ridicarea valorii acestuia, făcându-l mai puternic, mai adaptabil și mai relevant pentru contextul în care este utilizat.

Prin perfecționarea componentelor deja existente sau prin adăugarea unora noi, dezvoltatorul poate menține interesul utilizatorilor asupra proiectului. Câteva astfel de îmbunătățiri pot include modificarea interfeței, pentru a o face mai modernă, mai prietenoasă cu utilizatorul sau modificarea întregului cod pentru a îi crește eficiența și fiabilitatea. De asemenea există posibilitatea integrării cu tehnologii sau platforme suplimentare. De exemplu se pot integra dispozitive IoT sau se poate realiza conectarea la servicii de cloud computing.

Scopul final al îmbunătățirilor aduse proiectului nu este reprezentat doar de atingerea unui nivel superior de performanță și eficiență, dacă dezvoltatorul vrea să facă din sistemul creat un produs sau serviciu, acesta trebuie să îndeplinească nevoile și să depășească așteptările utilizatorilor.

În acest moment sistemul meu este funcțional, iar aplicabilitatea lui este destul de bună, dar există loc pentru numeroase îmbunătățiri, mai ales dacă se dorește implementarea lui la nivel comercial. În acest capitol voi enumera și prezenta câteva posibile viitoare implementări ale acestui proiect.

- Cea mai evidentă îmbunătățire care se poate realiza în cadrul acestui proiect este perfecționarea algoritmului de recunoaștere facială. Acesta poate fi perfecționat atât în procesul de detectare și identificare, cât și prin adăugarea capacității de a recunoaște mai multe persoane în același timp.

- Se pot schimba componentele hardware cu unele mai performante. De exemplu senzorul infraroșu cu detectarea întreruperii razei de lumină poate fi înlocuit cu un senzor PIR (senzor infraroșu pasiv), iar camera poate fi schimbată cu una cu rezoluție mai bună și cu capacitatea de aș-i modifica poziția. Astfel încât camera să se orienteze în direcția generală în care senzorul PIR a detectat mișcare. În figura 6-1 am reprezentat un senzor PIR și o cameră cu posibilitatea modificării poziției.

Figura 6-1 Schimbarea orientării camerei [25][26]

- Realizarea unei aplicații mobile care să comunice cu sistemul și care să poată să-l activeze sau sa-l dezactiveze. De asemenea tot în cadrul acelei aplicații să se primească notificările de avertizare în caz de pătrundere neautorizată și să se stocheze și să se catalogheze, în funcție de oră și data, videoclipurile înregistrate. Aplicația se poate realiza în Android Studio.

- O altă implementare utilă pentru îmbunătățirea sistemului poate fi reprezentată de crearea unei metode de conectare la Raspberry Pi fără a fi în aceeași rețea de internet cu acesta. Printr-o asemenea metodă utilizatorul poate verifica personal sistemul de securitate chiar dacă se află la distanță mare de acesta. O variantă bună pentru a obține această funcționalitate este utilizarea unui serviciu de tunelare SSH, cum ar fi ngrok sau Serveo, pentru a crea un tunel securizat prin intermediul căruia să se poată accesa sistemul de la distanță. Aceste servicii funcționează în baza unui URL prin care se realizează o conexiune securizată.

- Implementarea unui sistem de alarmă sonoră poate aduce beneficii semnificative în cea ce privește siguranța si protecția unui spațiu. Acesta poate contribuii atât la descurajarea intrusului, cât și la alertarea autorităților, care se pot autosesiza dacă aud alarma sau pot fi alertate de o altă persoană care a auzit alarma.

- Integrarea unui sistem suplimentar de control al accesului. Sistemului actual de securitate cu recunoaștere facială poate fi îmbunătățit cu un sistem auxiliar de control al accesului, care să permită accesul, sau nu, într-o locație specifică în baza identificării faciale.

7 CONCLUZII

Prin intermediul acestui proiect am vrut să implementez un sistem de securitate bazat pe algoritmi de recunoaștere facială, care să poată alerta proprietarul acestuia în cazul unei pătrunderi neautorizate în zona protejată. Am urmat mai mulți pași în realizarea acestui sistem, pași pe care i-am prezentat anterior în paginile acestei documentații.

Pentru a realiza acest proiect a fost nevoie să mă documentez asupra unor aspecte absolut necesare precum sunt algoritmi de recunoaștere facială, protocolul SMTP, librăria OpenCV și să îmi dezvolt abilități noi în limbajul de programare Python. De asemenea a trebuit să aleg piesele potrivite pentru implementarea hardware a sistemului. În final am ales să folosesc pentru detectarea mișcării un senzor infraroșu cu detectarea întreruperii razei de lumină și un computer mono-circuit Raspberry Pi 4 Model B pentru implementarea software-ului.

În realizarea proiectului etapa ce mai ușoară a fost reprezentată de asamblarea componentelor hardware, deoarece nu a fost nevoie să proiectez un PCB sau să efectuez lipiri. Toate componentele acestui proiect s-au conectat simplu și eficient utilizând interfețele și porturile pe care Raspberry Pi le posedă.

Implementarea codului în Python a fost un proces îndelungat în care am realizat numeroase teste pentru funcționalitățile software și pentru componentele hardware ale acestui proiect de diplomă. Această etapă a proiectului a fost definită de expresia încercare și eșec, deoarece am încercat multiple combinații de secvențe de cod până am ajuns la rezultatul dorit.

O etapă din dezvoltarea sistemului, care mi-a consumat mai mult timp și atenție decât m-aș fi așteptat a fost reprezentată de integrarea bibliotecii OpenCV pe sistemul de operare Raspberry Pi OS. În cadrul procesului de configurare ale librăriilor conținute de OpenCV am întâmpinat erori ce au necesitat atenție specială pentru soluționarea acestora.

După ce am reușit să implementez sistemul așa cum l-am vizualizat, am efectuat diverse teste pentru a descoperi posibile vulnerabilități și deficiențe ale acestuia. Teste ale căror rezultate au fost mulțumitoare, deoarece în cadrul acestora nu am găsit defecțiuni majore sau prelucrări eronate a datelor. Singurele inconveniențe sau puncte slabe pe care le-am observat au fost limitările generale pe care le au algoritmii de recunoaștere facială.

În concluzie, prin intermediul acestui proiect, am învățat noțiuni de bază ale protocolului SMTP, detalii semnificative ale specificațiilor componentelor hardware, principiile algoritmilor de recunoaștere facială și cum să-i implementez utilizând limbajul de programare Python.

6 BIBLIOGRAFIE

[1] D. Peleshko, Y. Ivanov, B. Sharov, I. Izonin and Y. Borzov, "Design and implementation of visitors queue density analysis and registration method for retail video surveillance purposes," 2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine, 2016, pp. 159-162, doi: 10.1109/DSMP.2016.7583531.

[2] I. Aciobanitei, P. D. Urian and M. Pura, "A Cryptography API: Next Generation Key Storage Provider for Cryptography in the Cloud," 2018 10th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Iasi, Romania, 2018, pp. 1-4, doi: 10.1109/ECAI.2018.8679042.

[3] C. Marshall, T. Parker and T. White, "Infrared sensor technology," Proceedings of 17th International Conference of the Engineering in Medicine and Biology Society, Montreal, QC, Canada, 1995, pp. 1715-1716 vol.2, doi: 10.1109/IEMBS.1995.579906.

[4] S. Khanji, R. Jabir, L. Ahmad, O. Alfandi and H. Said, "Evaluation of Linux SMTP server security aspects — A case study," 2016 7th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 2016, pp. 252-257, doi: 10.1109/IACS.2016.7476120.

[5] M. Khan, S. Chakraborty, R. Astya and S. Khepra, "Face Detection and Recognition Using OpenCV," 2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 2019, pp. 116-119, doi: 10.1109/ICCCIS48478.2019.8974493.

[6] G. Singh, I. Gupta, J. Singh and N. Kaur, "Face Recognition using Open Source Computer Vision Library (OpenCV) with Python," 2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2022, pp. 1-6, doi: 10.1109/ICRITO56286.2022.9964836.

[7] P. A. Harsha Vardhini, S. P. R. D. Reddy and V. P. Parapatla, "Facial Recognition using OpenCV and Python on Raspberry Pi," 2022 International Mobile and Embedded Technology Conference (MECON), Noida, India, 2022, pp. 480-485, doi: 10.1109/MECON53876.2022.9751867.

[8] N. Boyko, O. Basystiuk and N. Shakhovska, "Performance Evaluation and Comparison of Software for Face Recognition, Based on Dlib and Opencv Library," 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine, 2018, pp. 478-482, doi: 10.1109/DSMP.2018.8478556.

[9] T. Surasak, I. Takahiro, C. -h. Cheng, C. -e. Wang and P. -y. Sheng, "Histogram of oriented gradients for human detection in video," 2018 5th International Conference on Business and Industrial Research (ICBIR), Bangkok, Thailand, 2018, pp. 172-176, doi: 10.1109/ICBIR.2018.8391187.

[10] R. R. Kalangi, S. Maloji, P. S. Sundar and S. H. Ahammad, "Deployment of Haar Cascade Algorithm to Detect Real-Time Faces," 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2022, pp. 1676-1680, doi: 10.1109/ICSSIT53264.2022.9716530.

[11] R. Rosnelly, M. S. Simanjuntak, A. Clinton Sitepu, M. Azhari, S. Kosasi and Husen, "Face Recognition Using Eigenface Algorithm on Laptop Camera," 2020 8th International Conference on Cyber and IT Service Management (CITSM), Pangkal, Indonesia, 2020, pp. 1-4, doi: 10.1109/CITSM50537.2020.9268907.

[12] S. Q. Nur Septi, I. N. Yulita and H. Napitupulu, "Face Recognition Using Fisherface and Support Vector Machine Method," 2021 International Conference on Artificial Intelligence and Big Data Analytics, Bandung, Indonesia, 2021, pp. 50-55, doi: 10.1109/ICAIBDA53487.2021.9689738.

[13] Suherwin, Z. Zainuddin and A. A. Ilham, "The Performance of Face Recognition Using the Combination of Viola-Jones, Local Binary Pattern Histogram and Euclidean Distance," 2020 4th International Conference on Informatics and Computational Sciences (ICICoS), Semarang, Indonesia, 2020, pp. 1-4, doi: 10.1109/ICICoS51170.2020.9299073.

[14] Bob Mesnik, “The History of Video Surveillance”, Kintronics, 2016, Disponibil online: https://kintronics.com/the-history-of-video-surveillance/ . Accesat pe data de 19 mai 2023.

[15] "Imaginea camerei de supraveghere Tuya 3MP Wi-Fi." Queens Depot, Queens Depot, Disponibil online: https://queensdepot.ro/4-Tuya-camera-ip-3mp-wifi-camera-de-supraveghere-video/img_images-187639.jpeg . Accesat pe data de 25 mai 2023.

[16] "5 proyectos que podemos hacer con Raspberry Pi para nuestra casa." HWLibre, https://www.hwlibre.com/ro/5-proyectos-que-podemos-hacer-con-raspberry-pi-para-nuestra-casa/#Vigila_mascotas . Accesat pe data de 25 mai 2023.

[17] "Raspberry Pi 4 on Sale Now from $35." Raspberry Pi, Raspberry Pi Foundation, Disponibil online: https://www.raspberrypi.com/news/raspberry-pi-4-on-sale-now-from-35/ . Accesat pe data de 25 mai 2023.

[18] "Camera Module." Raspberry Pi Documentation, Raspberry Pi Foundation, Disponibil online: https://www.raspberrypi.com/documentation/accessories/camera.html . Accesat pe data de 26 mai 2023.

[19] "Senzoare infraroșu care detectează întreruperea razei de lumină cu LED-uri de 5 mm." Optimus Digital, Disponibil online:https://static.optimusdigital.ro/16163/senzor-infrarou-care-detecteaza-intreruperea-razei-de-lumina-cu-led-uri-de-5-mm.jpg . Accesat pe data de 27 mai 2023.

[20] Autori: Postel, J. (editor), Titlul documentului: Simple Mail Transfer Protocol, Numărul și numele standardului: RFC 5321, Organizația care emite standardul: IETF (Internet Engineering Task Force), Disponibil online: https://tools.ietf.org/html/rfc5321 . Accesat pe data de 27 mai 2023.

[21] Nesbitt, Peter. "Raspberry Pi CSI-2 Connector Specifications." Peter Vis. https://www.petervis.com/Raspberry_PI/Raspberry_Pi_CSI/Raspberry_Pi_CSI-2_Connector_Specifications.html . Accesat pe data de 29 mai 2023.

[22] Nesbitt, Peter. "Raspberry Pi CSI Interface Connector Pinout." Peter Vis. https://www.petervis.com/Raspberry_PI/Raspberry_Pi_CSI/raspberry-pi-csi-interface-connector-pinout.html . Accesat pe data de 2 iunie 2023.

[23] Sharky93. "Plotting the HOG features". GitHub Pages. http://sharky93.github.io/docs/dev/auto_examples/plot_hog.html . Accesat pe data de [data accesării].

[24] ARM. "ARM Information Center - ARM Cortex-A72 MPCore Processor Technical Reference Manual r0p3." Disponibil la: https://developer.arm.com/documentation/100095/0003/?lang=en . Accesat pe data de 4 iunie 2023.

[25] Modul PIR - Senzor de prezență și mișcare. Disponibil la: https://ardushop.ro/ro/electronica/45-modul-pir-senzor-de-prezenta-miscare.html . Accesat pe data de 7 iunie 2023.

[26] Camera de supraveghere Wi-Fi Sonoff S-Cam HD cu iluminator IR și slot pentru card. Disponibil la: https://www.spy-shop.ro/camera-supraveghere-wi-fi-sonoff-s-cam-hd-iluminator-ir-slot-card.html?gclid=Cj0KCQjwnMWkBhDLARIsAHBOftoFPNRndWsa4jc0gkVdj7_n7y1tRG1_MvF-wWEfWx7rKyLAlXWO33waAqAjEALw_wcB . Accesat pe data de 7 iunie 2023.

REZUMAT

În acest proiect am implementat un sistem de securitate cu recunoaștere facială pentru protecția locuinței sau a unui punct de acces pentru o anumită facilitate. Principalele aspecte tratate în acest proiect de diplomă sunt descrise în următoarele rânduri.

Primul aspect principal pe care l-am tratat a fost documentarea asupra proiectului, unde am prezentat noțiuni importante legate de componentele hardware și software folosite. În cadrul acelui capitol am tratat subiecte precum securitatea SMTP pe Linux, algoritmi utilizați în procesul de recunoaștere facială și utilizarea bibliotecii OpenCV în combinație cu limbajul de programare Python pe Raspberry Pi.

Un alt aspect important a fost configurația hardware, unde am prezentat specificațiile computerului mono-circuit Raspberry Pi 4 Model B, informații necesare legate de senzorul folosit în proiect și descrierea interfeței CSI. De asemenea tot în cadrul acestui capitol am prezentat modul în care am testat componentele hardware folosind coduri Python.

În cadrul capitolului dedicat arhitecturii software am prezentat și explicat secvențele de cod principale utilizate pentru realizarea funcționalității proiectului. Printre acestea se numără codurile pentru antrenarea algoritmului, utilizarea protocolului SMTP pentru a trimite e-mail-uri și programul principal care realizează înregistrarea videoclipului și procesul de recunoaștere facială. Tot în acest capitol am tratat și noțiuni legate de performanța și eficiența sistemului meu, precum timpul necesar de rulare a programului, gradul de utilizarea a resurselor și calitatea procesului de recunoaștere facială.

Toate aceste informații prezentate anterior reprezintă aspectele principale tratate în cadrul proiectului meu de diplomă.

ABSTRACT

In this project I have implemented a facial recognition security system to protect the home or an access point for a specific facility. The main points covered in this diploma project are described in the following.

The first main aspect I dealt with was the documentation of the project, where I presented important notions related to the hardware and software components used. Within that chapter we covered topics such as SMTP security on Linux, algorithms used in the facial recognition process and the use of the OpenCV library in combination with the Python programming language on Raspberry Pi.

Another important aspect was the hardware configuration, where I presented the specifications of the Raspberry Pi 4 Model B single-circuit computer, necessary information related to the sensor used in the project and description of the CSI interface. Also in this chapter I presented how I tested the hardware components using Python programs.

In the chapter dedicated to the software architecture I presented and explained the main code sequences used to realize the project functionality. These include the codes for training the algorithm, using the SMTP to send emails and the main program that performs the video recording and facial recognition process. Also in this chapter I have dealt with notions related to the performance and efficiency of my system, such as the time required to run the program, the resource utilization and the quality of the facial recognition process.

All of this information presented above are the main aspects covered in my diploma project.

ANEXA - CODUL SURSĂ AL SISTEMULUI

headshot_picam.py

```
import cv2
from picamera import PiCamera
from picamera.array import PiRGBArray

name = 'Lucian'

# Inițializare cameră PiCamera
cam = PiCamera()
cam.rotation = 180
cam.resolution = (512, 304)
cam.framerate = 10
rawCapture = PiRGBArray(cam, size=(512, 304))

img_counter = 0

while True:
    # Capturare cadre continuu
    for frame in cam.capture_continuous(rawCapture, format="bgr", use_video_port=True):
        image = frame.array

        # Afișare imagine
        cv2.imshow("Press Space to take a photo", image)

        # Eliberare buffer pentru următorul cadru
        rawCapture.truncate(0)

        # Verificare apăsare tastă
        k = cv2.waitKey(1)
        rawCapture.truncate(0)

        # Dacă este apăsată tasta 'q', se iese din buclă
        if k == ord('q'):
            break
        # Dacă este apăsată tasta Spațiu, se salvează imaginea
        elif k == 32:
            # Tasta Spațiu apăsată
            img_name = "Imagini/" + name + "/imagine_{}.jpg".format(img_counter)
            cv2.imwrite(img_name, image)
            print("{} salvată!".format(img_name))
            img_counter += 1

    if k == ord('q'):
        print("q hit, closing...")
        break

# Închidere ferestre
cv2.destroyAllWindows()
```

mail.py

```
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.base import MIMEBase
from email import encoders
import os
import glob

def send_video_email(sender_email, sender_password, receiver_email, video_directory):
    # Configurarea detaliilor de autentificare SMTP
    smtp_server = 'smtp.gmail.com'
    smtp_port = 587

    # Obținerea cel mai recent fișier video din directorul specificat
    video_files = glob.glob(os.path.join(video_directory, '*.mp4'))
    latest_video = max(video_files, key=os.path.getctime)

    # Crearea obiectului pentru mesajul e-mail
    msg = MIMEMultipart()
    msg['From'] = sender_email
    msg['To'] = receiver_email
    msg['Subject'] = '👾 INVADER 👾'
```

```
    # Deschiderea fișierului video și atașarea acestuia la mesajul e-mail
    with open(latest_video, 'rb') as f:
        attachment = MIMEBase('application', 'octet-stream')
        attachment.set_payload(f.read())
        encoders.encode_base64(attachment)
        attachment.add_header('Content-Disposition', f'attachment; filename={latest_video}')
        msg.attach(attachment)

    # Inițializarea conexiunii SMTP și trimiterea mesajului
    try:

       server = smtplib.SMTP(smtp_server, smtp_port)
        server.starttls()
        server.login(sender_email, sender_password)
        server.sendmail(sender_email, receiver_email, msg.as_string())
        print('E-mail sent successfully!')
    except Exception as e:
        print(f'Error occurred while sending e-mail: {str(e)}')
    finally:
        server.quit()

# Datele de autentificare
sender_email = 'lucianraspberry07@gmail.com'
sender_password = '*****************'
receiver_email = 'neagulucian1889@gmail.com'
video_directory = '/home/pi/Desktop/proiect_licenta/video'

send_video_email(sender_email, sender_password, receiver_email, video_directory)
```

train_model.py

```
from imutils import paths
import face_recognition
import pickle
import cv2
import os

# Definirea directorului pentru setul de date
dataset_folder = "Images"

# Obținerea phats-urilor către imagine
imagePaths = list(paths.list_images(dataset_folder))

# Inițializarea listelor de codificări și nume
knownEncodings = []
knownNames = []

# Parcurgerea în buclă a phat-urilor către imagini
for (i, imagePath) in enumerate(imagePaths):
    # Extragerea numelui persoanei
    name = os.path.basename(os.path.dirname(imagePath))

    # Încărcarea imagini și conversia acesteia în RGB
    image = cv2.imread(imagePath)
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Detectarea locației feței în imagine
    boxes = face_recognition.face_locations(rgb, model="hog")

    # Calcularea codificărilor feței
    encodings = face_recognition.face_encodings(rgb, boxes)


    # Atribuirea codificărilor și a numelor listelor respective
    knownEncodings.extend(encodings)
    knownNames.extend([name] * len(encodings))

# Serializarea codificarilor și a numelor pe disc
print("[INFO] Serializing encodings...")
data = {"encodings": knownEncodings, "names": knownNames}
with open("encodings.pickle", "wb") as f:
    f.write(pickle.dumps(data))
```

main_cod.py

```
from picamera import PiCamera
import RPi.GPIO as GPIO
from time import sleep, strftime
import subprocess
import os
import cv2
import face_recognition
import imutils
import pickle
from imutils.video import FPS
import face_recognition_models


BEAM_PIN = 17

def break_beam_callback(channel):
    """
    Funcție de callback apelată atunci când se detectează o schimbare a
    stării fasciculului. Verifică starea fasciculului și înregistrează un
    videoclip dacă fasciculul este întrerupt.
    """
    if GPIO.input(BEAM_PIN):
        print("Beam unbroken")
    else:
        print("Beam broken")
        with PiCamera() as camera:
            camera.rotation = 180
            camera.start_preview()
            timestamp = strftime("%Y-%m-%d-%H-%M-%S")
            video_file = f'/home/pi/Desktop/proiect_licenta/video/video_{timestamp}.h264'
            camera.start_recording(video_file)
            sleep(5)
            camera.stop_recording()
            camera.stop_preview()
        process_video(video_file)

def convert_to_mp4(video_file):
    """
    Converteste un fișier video .h264 în format .mp4 utilizând MP4Box.
    """
    output_file = video_file.replace('.h264', '.mp4')
    command = ['MP4Box', '-add', video_file, output_file]
    subprocess.run(command)
    print(f"Video saved as {output_file}")

def delete_file(file_path):
    """
    Șterge un fișier din sistem.
    """
    os.remove(file_path)
    print(f"Deleted file: {file_path}")

def process_video(video_file):
    """
    Procesează un fișier video înregistrat anterior, detectează fețele
    și le recunoaște utilizând modele pre-antrenate.
    Apoi afișează videoclipul cu fețele și numele recunoscute.
    """
    currentname = "unknown"
    encodingsP = "encodings.pickle"
    print("[INFO] loading encodings + face detector...")
    data = pickle.loads(open(encodingsP, "rb").read())
    vs = cv2.VideoCapture(video_file)
    fps = FPS().start()
    video_ended = False

    # Capturarea și procesarea fiecărui frame al videoclipului
    while not video_ended:
        ret, frame = vs.read()
        if not ret:
            video_ended = True
            break
        frame = imutils.resize(frame, width=500)
        boxes = face_recognition.face_locations(frame)
        encodings = face_recognition.face_encodings(frame, boxes)
```

```
        names = []

        for encoding in encodings:
            matches = face_recognition.compare_faces(data["encodings"], encoding)
            name = "Unknown"

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

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

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

                # Actualizarea numelui dacă o potrivire este găsită
                if currentname != name:
                    currentname = name
                    print(currentname)

            names.append(name)

        # Desenarea dreptunghiurilor și afișarea numelui pentru fiecare față detectată
        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, .8, (0, 255, 255), 2)

        cv2.imshow("Facial Recognition is Running", frame)
        key = cv2.waitKey(1) & 0xFF

        if key == ord("q"):
            break

        fps.update()

    cv2.destroyAllWindows()
    vs.release()
    fps.stop()

    convert_to_mp4(video_file)
    delete_file(video_file)

    # Trimiterea emailului doar dacă numele detectat nu este "Lucian"
    if currentname != "Lucian":
        send_video_email()

def send_video_email():
    """
    Trimite un email cu videoclipul înregistrat.
    """
    command = ['python3', 'mail.py']
    subprocess.run(command)


GPIO.setmode(GPIO.BCM)
GPIO.setup(BEAM_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.add_event_detect(BEAM_PIN, GPIO.BOTH, callback=break_beam_callback)

message = input("Press enter to quit\n\n")
GPIO.cleanup()
```