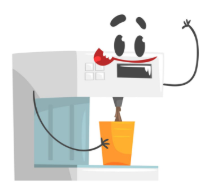
Universitatea Tehnică “Gheorghe Asachi”

Facultatea de Inginerie Electrică, Energetică și Informatică Aplicată

Aparat de cafea inteligent



Autori : Grigore Alin

Grecu Bianca

Rusti Alexandra

Titirez Daniel

Îndrumător: Marius Brânzilă

Cuprins

1. Introducere
   1. Definiție RNA
   2. Asemănarea cu creierul
   3. Neuronul artificial
   4. Machine learning
2. Partea Hardware
   1. Raspberry Pi
   2. Cameră video
   3. Senzori de mișcare
   4. Expressor
3. Partea Software
4. Concluzii
5. Anexe
6. Bibliografie

**Abstract - În aceasta lucrare este prezentat un sistem de recunoaștere facială ca o aplicație computerizată capabilă să detecteze și să verifice identitatea unei persoane dintr-o imagine digitală sau un cadru de la o cameră video. Lucrarea noastră poate face acest lucru prin compararea caracteristicilor faciale ale persoanei din imagine cu o bază de date deja existentă.**

**Keywords: Raspberry Pi, cameră video, senzor de mișcare, rețele neuronale, machine learning.**

1. **Introducere**

În această lucrare este prezentat un sistem de recunoaștere facială bazat pe rețele neuronale. Sistemul este împărțit structural pe 2 părți:

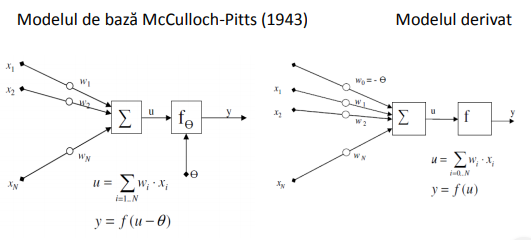
* Partea Hardware
* Partea Software

Rețelele neuronale folosite pentru recunoașterea optică fac parte din aria computer vision (CV) numele sugerează crearea unor abstractizări ale rețelelor neuronale, în cadrul unei mașini computaționale, pentru a putea recunoaște și raporta diverse tipare vizuale. [2]

* 1. Definiție Rețele Neuronale Artificiale (RNA):

O rețea neuronală artificială este un calculator distribuit, masiv paralel, care achiziționează noi cunoștințe pe baza experienței anterioare și le face disponibile pentru utilizarea ulterioară (S.Haykin, 1994). [6]

* 1. Asemănarea cu creierul :
* Cunoștințele sunt înmagazinate în conexiunile inter-neuronale (ponderi sinaptice)
* Cunoștințele sunt achiziționate de rețeaua neurală printr-un proces de învațare
* Tipul unitaților funcționale (elemente de procesare numite neuroni)
* Arhitectura (amplasare unitați funcționale)
* Algoritm de funcționare (transformare semnal intrare în semnal ieșire)
* Algoritm de învatare (cum achizitionează rețeaua noi cunoștințe pe bază de exemple)
  1. Neuronul artificial

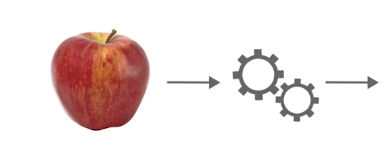


* 1. Machine learning

Studiul algoritmilor care “învață” din exemple și experiențe, nu se bazează pe reguli predefinite, “hard-codate”. Denumirea aduce un pic a “Sci-Fi”, dar conceptul de bază este ușor de înțeles. În momentele ce urmează vă prezentăm un exemplu pentru a înțelege mai bine despre ce este vorba.

O problemă la care dacă se caută rezolvare tradițională, am irosi mult prea multe resure este, de exemplu, cum facem un program care să distingă un măr de o portocală?

Să ne imaginăm că programul realizat tradițional are ca intrare o imagine ce reprezintă un fruct iar ca ieșire tipul fructului.



“MAR”

Pentru a face diferența dintre fructe, am putea, de exemplu, să implementăm o funcție care numară și compară pixelii portocalii și cei verzi din poză, raportul dintre cele două ne-ar putea da un indiciu în luarea deciziei.

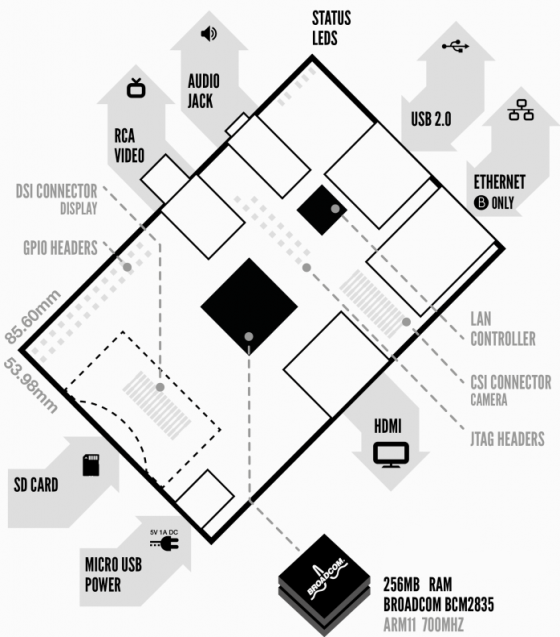
Totul functionează bine pentru imagini simple precum cele prezentate, dar ce ne facem cand regulile impuse de noi nu mai funcționează (poza care este alb-negru sau nu avem nici un fruct in poză). Ar trebui să scrie o mulțime de reguli doar pentru a diferenția cele 2 fructe. Iar dacă am considera un alt exemplu, ar trebui să începem totul de la zero. ☹

Clar avem nevoie de ceva mai bun!

Avem nevoie de un algoritm care își dă seama de reguli de unul singur, fără ca noi să intervenim. Iar pentru aceasta avem nevoie de un “clasiffier” – îl considerăm ca fiind o funcție care atribuie etichete intrărilor.

Tehnica folosită pentru a implementa acest “classifier”, se numește “supervised learning”.

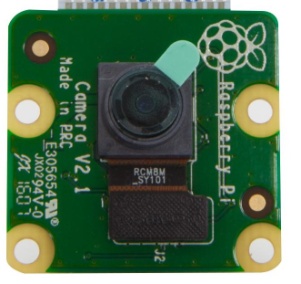
1. **Partea Hardware** 
   1. Raspberry Pi



RaspberryPi, zis **RasPi**, este un computer care rulează Linux, dar are dimensiunile unei cărți de credit.

Specificații hardware:

* SoC Broadcom BCM2835 (CPU, GPU, DSP, and SDRAM)
* CPU: 700 MHz ARM1176JZF-S core (ARM11 family)
* GPU: Broadcom VideoCore IV, OpenGL ES 2.0, 1080p30 h.264/MPEG-4 AVC high-profile decoder
* Memory (SDRAM): 256 Megabytes (MiB)
* Video outputs: Composite RCA, HDMI
* Audio outputs: 3.5 mm jack, HDMI
* Onboard storage: SD, MMC, SDIO card slot
* 10/100 Ethernet RJ45 onboard networkStorage via SD/ MMC/ SDIO card slot. [3]
  1. Camera video Raspberry Pi

Specificații hardware:

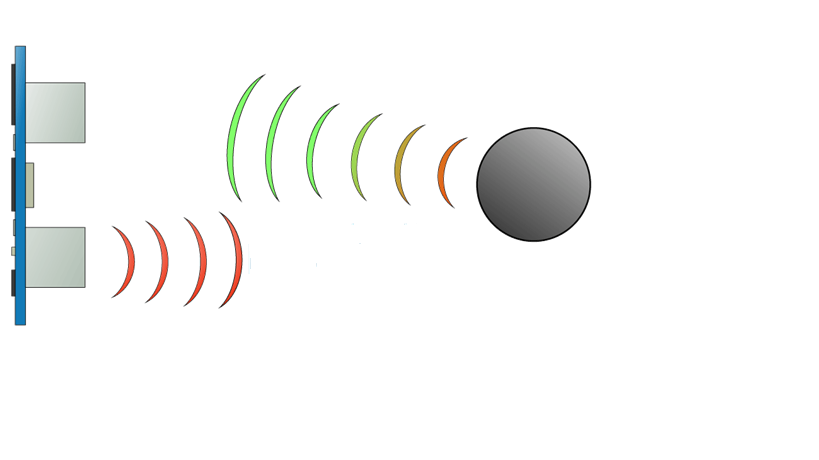
* Senzor 8 MegaPixeli (fata de 5 in versiunea 1)
* Photo : 3280 x 2464 px (fata de 2592 px x 1944 px in versiunea 1)
* Formate : 1080p / 720p
* 25mm x 23mm x 9mm. [4]
  1. Senzor de miscare HC-SR04

Specificații hardware:



* Putere: 5V DC
* Pasiv curent: <2mA
* Unghi eficace: <15 °
* Distanta variind: 2cm – 500 cm / 1 „- 16ft
* Rezolutie : 0,3 cm
* Dimensiune : 45 x 20 x 15mm
* Greutate: 8,5 g. [5]

Senzorul emite ultrasunete la frecvența de 40 000 Hz care se deplasează prin aer și dacă există un obiect sau un obstacol în calea sa, acesta va reveni la modul. Având în vedere timpul de călătorie și viteza sunetului, putem calcula distanța.

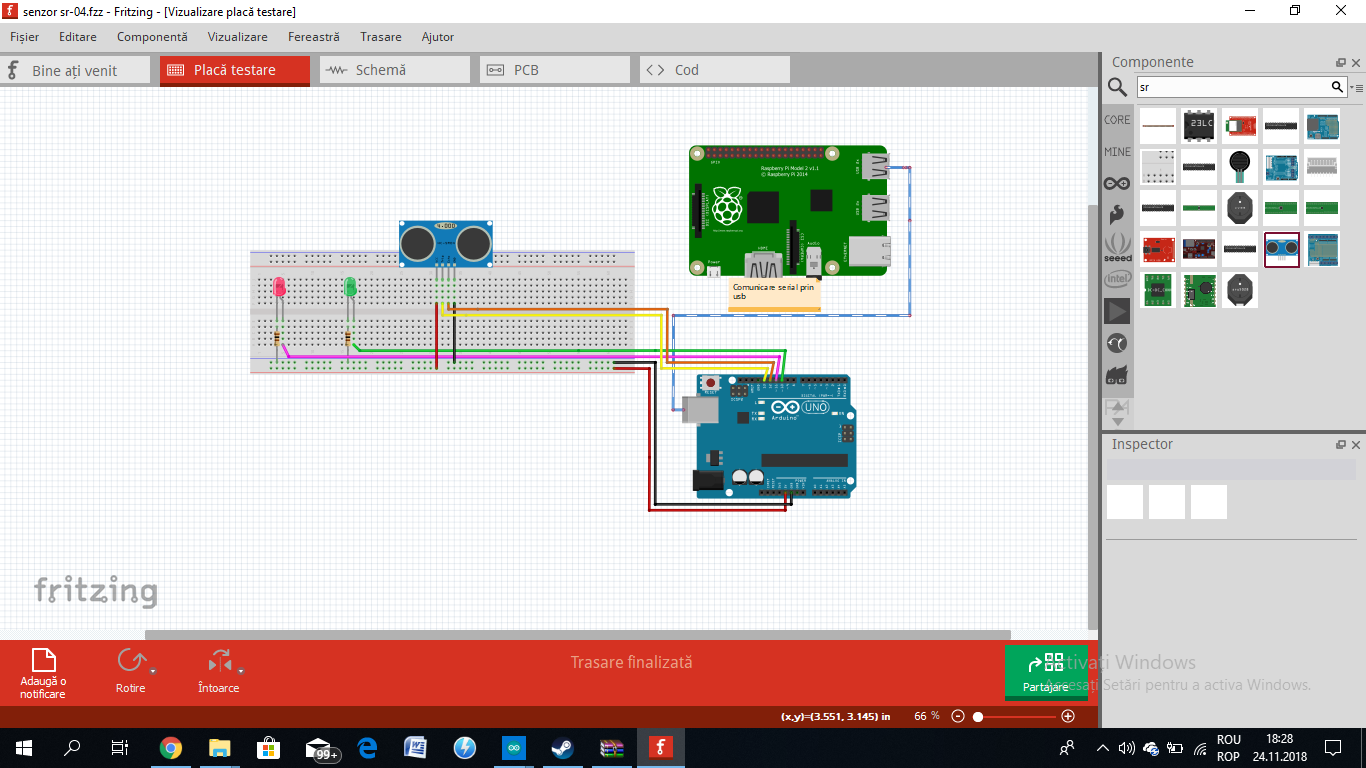


Modulul HC-SR04 cu ultrasunete are 4 pini: GND, VCC, Trig și Echo. Conectorii de la GND și VCC ai modulului trebuie să fie conectați la pinii de la GND și cei de 5 volți de pe placa Arduino, respectiv pinii de tip Trig și Echo la orice pin digital I / O de pe placa Arduino[10].

Pentru a arăta mai concret funcționalitatea sistemului, am folosit două led-uri. Led-ul roșu se aprinde atunci când sistemul este pus în funcțiune și indică faptul că senzorul nu detectează nici un obiect la o distanță mai mică de un metru. În momentul în care senzorul detectează un obiect la o distanță mai mică de un metru se face tranziția: se aprinde led-ul verde si se stinge led-ul roșu,ceea ce semnifică faptul că s-a îndeplinit condiția.

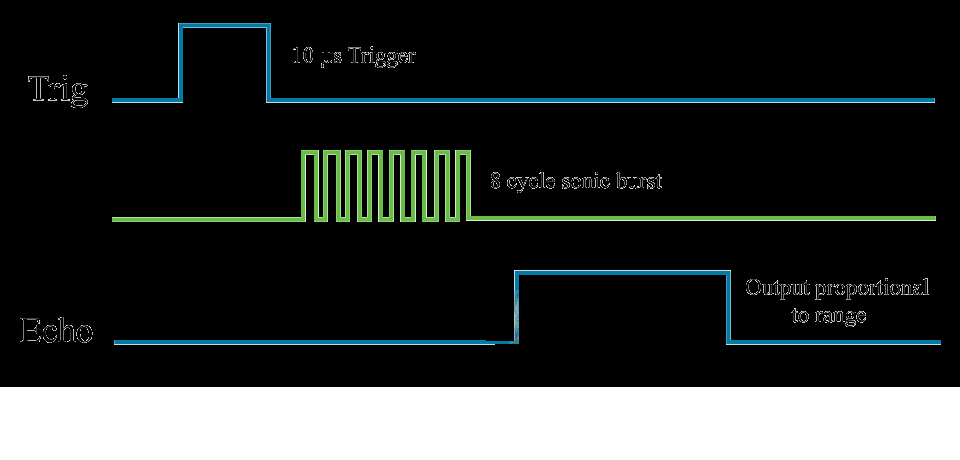
Schema montajului a fost realizată în programul Fritzing. Din schema bloc se observă conexiunea dintre senzor și Arduino,dar și comunicarea serial ce se realizează prin USB, prin care are loc transmisia datelor la Raspberry Pi 2.

Senzorul are pinul de GND și cel deVCC conectati la 5V de la Arduino respectiv la masa acestuia, iar pinii de date: trigger si echo sunt conectati la pinul 13 respectiv 12.Cele doua led-uri sunt conectate la pinii 10 si 11, având o rezistență pe pinul de la masă pentru protecția acestora.



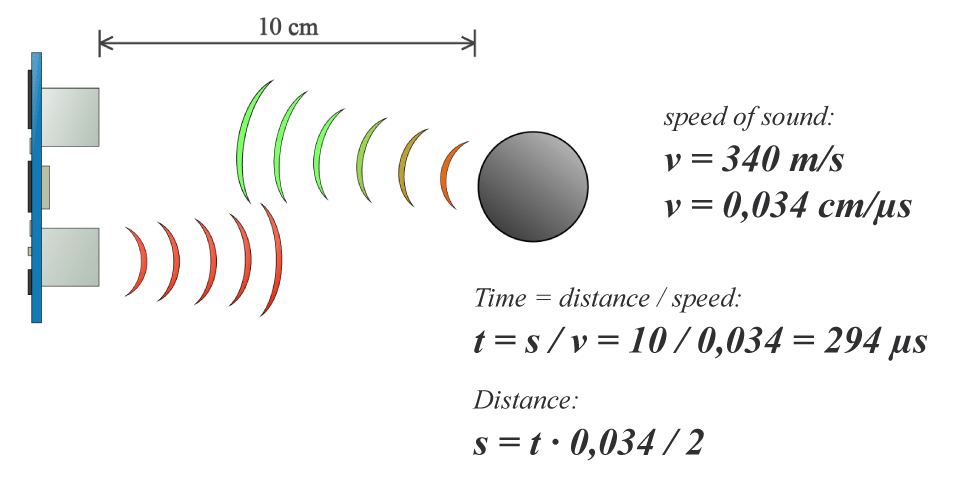
Schema bloc a montajului utilizat

Pentru a genera ultrasunetele trebuie să setăm pinul Trig la un nivel înalt timp de 10 μs. Aceasta va trimite un sunet de 8 cicluri care va călători cu viteza sunetului și va fi recepționată în pinul Echo. Pinul Echo va scoate la ieșire timpul, în microsecunde, de la transmisie pana la recepția undelor sonore[11].



Formele de undă specifice

De exemplu, dacă obiectul este la 10 cm distanță față de senzor și viteza sunetului este de 340 m / s sau 0,034 cm / μs, undele sonore vor trebui să călătorească aproximativ 294 secunde. Ceea ce se va obține de la pinul Echo va fi dublul numărului, deoarece valul sonor trebuie să călătorească înainte și înapoi. Deci, pentru a obține distanța în cm, trebuie să înmulțim valoarea timpului de călătorie primit de la Echo cu 0.034 și să o împărțim la 2.



* 1. Expressor

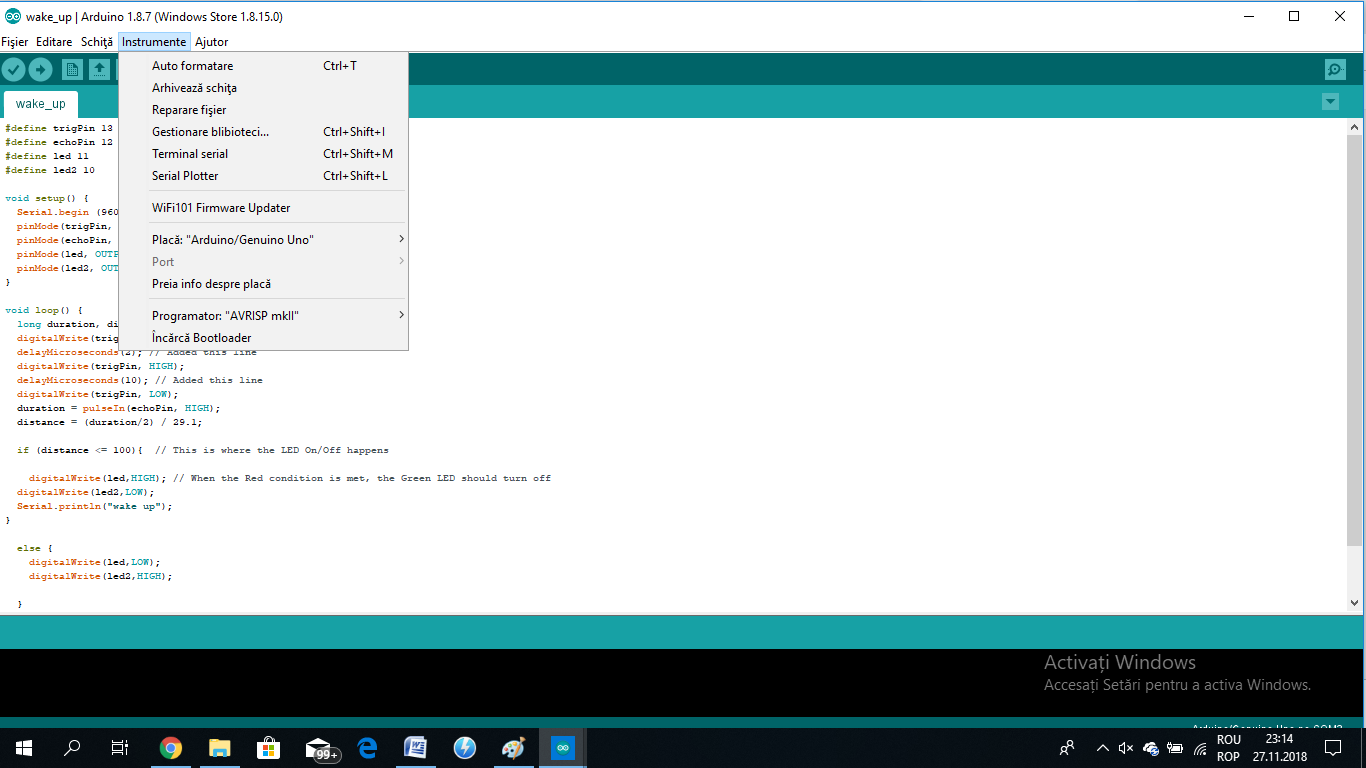
Simulăm prezența expresorului printr-un LED verde.

1. **Partea Software**
   1. **Software Arduino**

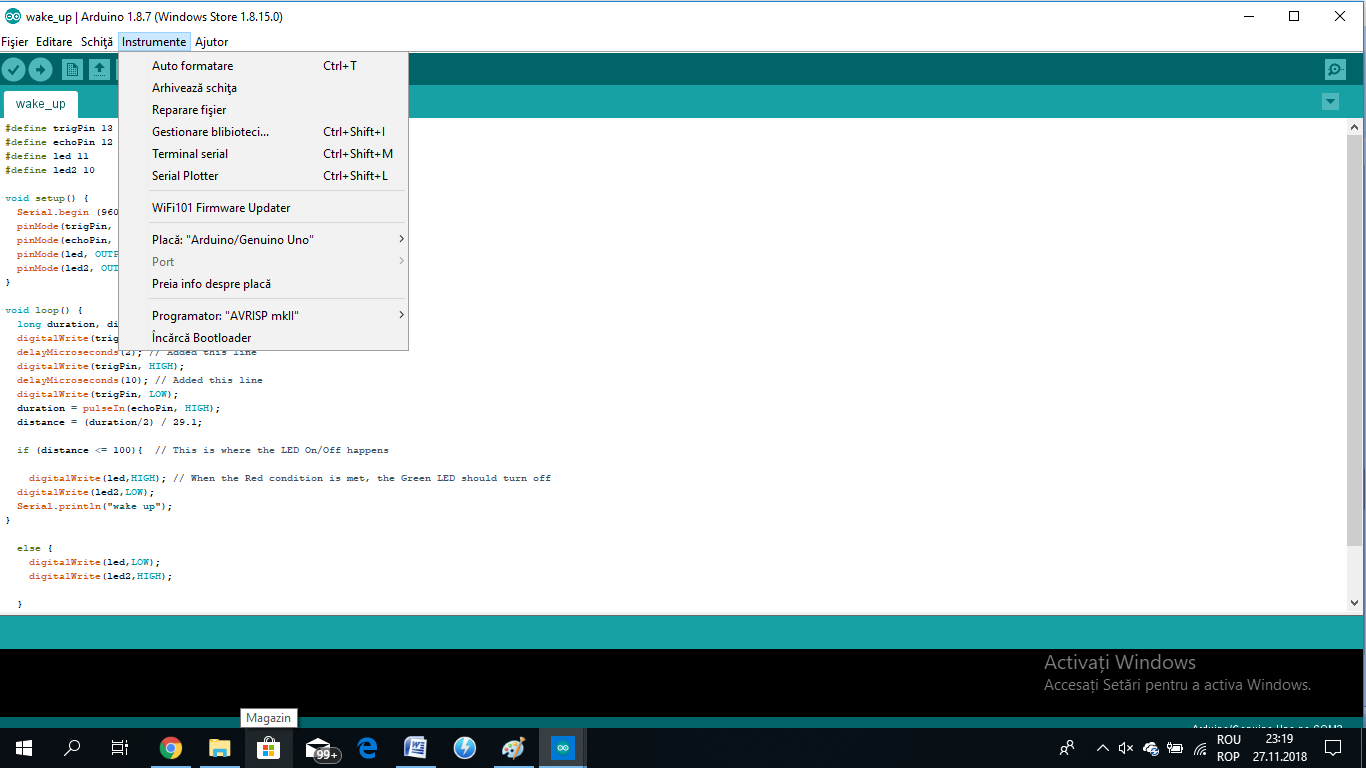
Partea Software ce include placa de dezvoltare Arduino a fost realizată în mediul de dezvoltare integrat(IDE) bazat pe proiectul Processing, care include suport pentru limbaje de programare precum C și C++.

Înainte ca programul să fie încărcat pe placa de dezvoltare,trebuie urmați pașii de configurare a compilatorului:

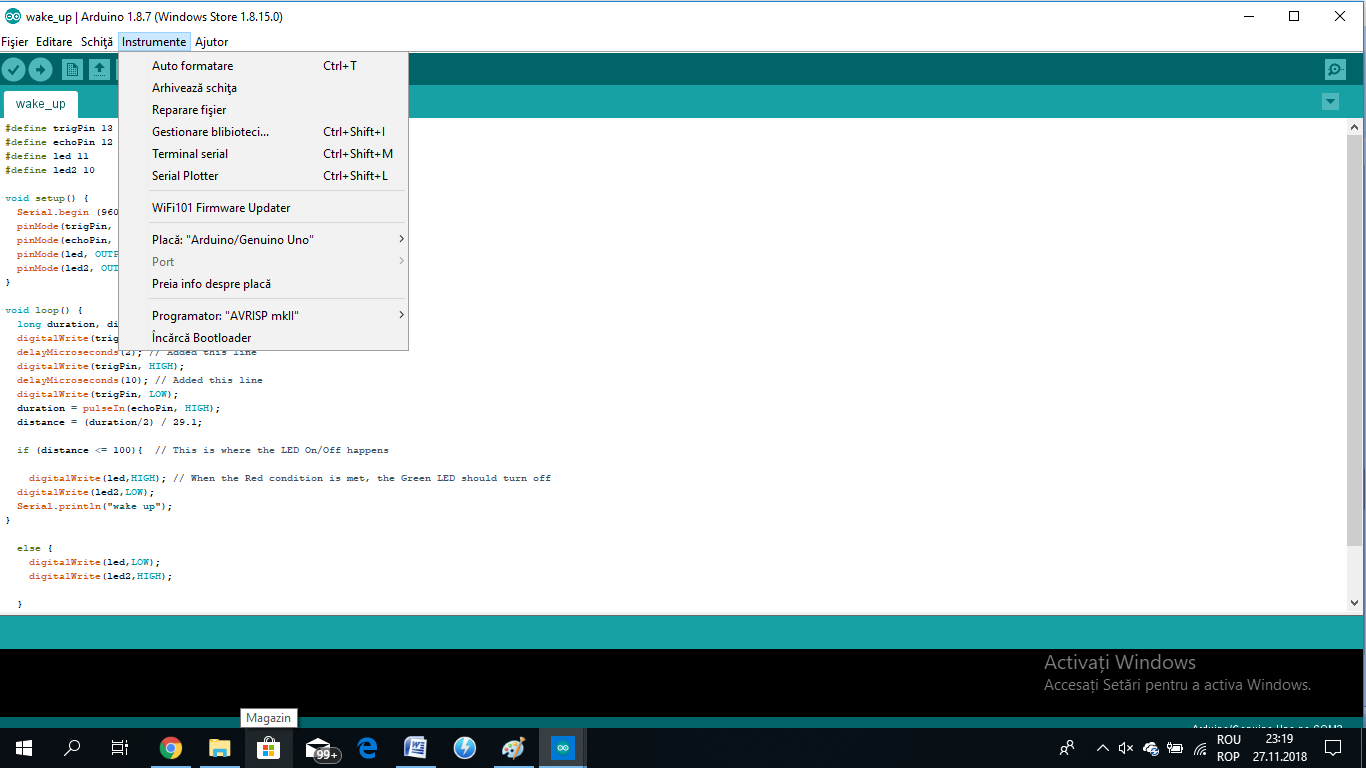
1. Selectare placa de dezvoltate folosita:



1. Selectare port prin intermediul căruia se face incărcarea codului:



1. Selectarea programatorului



**Codul sursa**

Orice program Arduino are două secțiuni. Secțiunea "setup", care este rulată o singură dată, atunci cand placa este alimentată (sau este apăsat butonul "Reset"), și secțiunea "loop" care este rulată în cicli, atât timp cât este alimentată placa. Astfel, în rutina "setup" vom pune de obicei cod de inițializare, iar în rutina "loop" vom scrie partea principală a programului nostru.

Codul pentru Arduino se regasește in Anexa I.

* 1. **Software Raspberry Pi**

Pentru comunicare pe serial prin intermediul USB-ului s-au folosit urmatoarele comenzi in terminal:

**Sudo apt-get update** – avem nevoie pentru a actualiza driverele si librariile necesare.

**Sudo apt-get upgrade** - se realizeaza instalarea pachetului de update

**Sudo apt-get install moreutils** - Printre utilitățile UNIX disponibile, există un pachet care include unele utilități minunate care fac viața noastră mult mai ușoară: pachetul mostutils.

Instrumentele mostutils:

• **chronic**: execută o comandă liniștit, cu excepția cazului în care nu reușește

• **combine**: combinați liniile în două fișiere utilizând operațiile booleene

• **errno**: caută nume și descrieri errno

• **ifdata**: obține informații despre interfața de rețea fără a analiza output-ul ifconfig

• **ifne**: execută un program dacă intrarea standard nu este goală

• **isutf8**: verifică dacă un fișier sau o intrare standard este utf-8

• **lckdo**: execută un program cu o blocare ținută

• **mispipe**: conduce două comenzi, returnând starea de ieșire a primei

• **parallel**: execută mai multe lucrări simultan

• **pee**: intrare standard în tuburi

• **sponge**: absoarbe intrarea standard și scrie intr-un fișier

• **ts**: intrarea standard a marcajului de timp

• **vidir**: editează un director în editorul de text

• **vipe**: introduce un editor de text într-un tub

• **zrun**: dezarhivează automat argumentele pentru a comanda

**Wget** - grabserial - prin această comandă importam codul pentru citirea de pe serial[7].

**Lsusb** - lsusb este un utilitar pentru afișarea informațiilor despre magistralele USB din sistem și despre dispozitivele conectate la acestea.

**Dmesg** - (mesajul de afișare sau mesajul driverului) este o comandă pentru majoritatea sistemelor de operare asemănătoare Unix care tipărește buffer-ul de mesaje al kernel-ului. Ieșirea acestei comenzi conține în mod obișnuit mesajele produse de driverele de dispozitive.

**Python grabserial | ls** – executam codul importat de pe github folosind compilatorul python pentru a citi datele de pe serial .

Exemplificarea funcționalității celor descrise mai sus se regăsește în urmatoarele două clipuri de pe pagina noastră de Github:

<https://github.com/Akim2222/Proiect_SE_SI/blob/master/Arhitectura_senzori/varianta%20finala/video-1543072425.mp4>

<https://github.com/Akim2222/Proiect_SE_SI/blob/master/Arhitectura_senzori/varianta%20finala/video-1543072436.mp4>

Pentru a face expresorul de cafea mai inteligent, prin a-l învața cine este în fața lui, vom folosi, pentru partea SW, un script scris în limbajul de programare Python. Acesta va fi responsabil pentru recunoaștere și identificare facială.

Deci vom avea două “module”: unul specializat pe recunoaștere și identificare, iar altul pe antrenarea rețelei neuronale.

Recunoașterea este diferită de identificarea facială din mai multe puncte de vedere, pe care le vom discuta în curand.

Pentru realizarea acestor lucruri avem nevoie de Pyton3, OpenCV (Open-source Computer Vision) cu toate dependențele aferente acestuia și modulul dlib.

OpenCV

OpenCV este un modul(librarie) open-source pentru algoritmi de “vedere computerizată”. Cu ajutorul acestui modul Python este realizată partea de recunoaștere și identificare facială astfel:

* O mulțime de imagini(pozitive si negative) representative;
* Pe baza mulțimii de imagini se va antrena o rețea neuronală care va fi capabilă să detecteze și să identifice fața unei persoane.

Imaginile pozitive trebuie să conțină, de preferat, doar obiectul de interes, sau în cazul nostru, o față. Iar cele negative opusul.

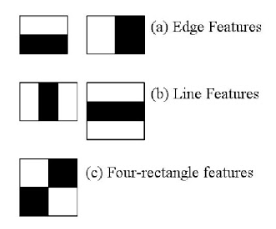
Scriptul care face diferențierea între un raspuns pozitiv și un raspuns negativ se numește clasificator (classifier). Un clasificator este antrenat, de regulă, pe mii sau zeci de mii de imagini pozitive/negative. În cazul nostru nu va fi nevoie de atât de multe exemple deoarece OpenCV ne pune la dispoziție un clasificator pre-antrenat pentru recunoaștere, noi fiind nevoiți doar sa ne antrenăm un identificator.

De interes pentru noi, din OpenCV, sunt clasificatorii în cascada de tip HAAR și LBP. Clasificatorii nu au nevoie de informația conținută în imagine sub formă de culoare, de aceea toate imaginile care vor fi prelucrate de acestea vor fi alb-negru (grayscale).

În cele ce urmeaza vom încerca, pe scurt să explicăm diferențele, avantajele și dezavantajele celor două tipuri de clasificatori.

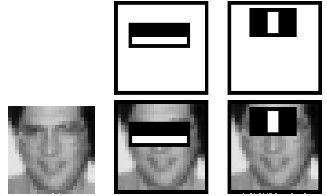
***Clasificatorul în cascada de tip HAAR***

Clasificatorul de tip Haar are la bază un algoritm(creat de Paul Viola și Michael Jones în 2001)îin care, practic, cauta și compara zone din imagini. Din imagine se vor extrage caracteristici cu ajutorul caracteristicilor de tip haar reprezentate mai jos.



Caracteristicile de tip haar

În imaginile primite de la cameră, se definesc zeci sau chiar sute de mii de zone(proporțional cu rezoluția) cu ajutorul caracteristicilor de tip haar, dar clasificatorul în cascada de tip haar disponibil verifică fiecare caracteristică pe zone și dacă informația obținută nu este relevantă, algoritmul ignoră complet zona respectiva și trece mai departe la o noua zonă din imagine. Acest proces de “filtrare” a zonelor se numește Adaboost



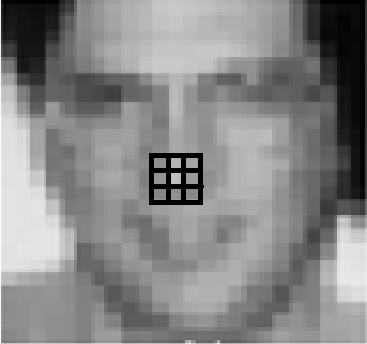
Caracteristicile de tip haar suprapuse pe imaginea de interes

În imaginea de mai sus observam cum toate caracteristicile de tip haar sunt aplicate imaginii de interes. Pentru fiecare caracteristică se gasește un prag decisiv, dupa care se decide în final dacă imaginea este pozitivă sau negativă.

***Clasificatorul de tip LBP***

Clasificatorul LBP (Local Binary Pattern) sau modelul local binar este un descriptor vizual/textural care procesează imaginea în modul următor:

* Împarte imaginea(matricea bidimensională) în subcategorii(submatrici) de 3x3 pixeli



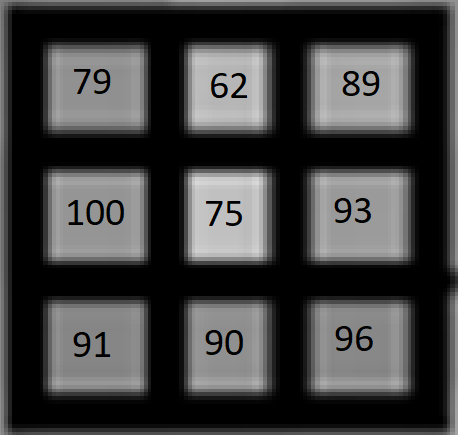
Imagine impartita in sectiuni de 3x3

* Selectează zona de 3x3



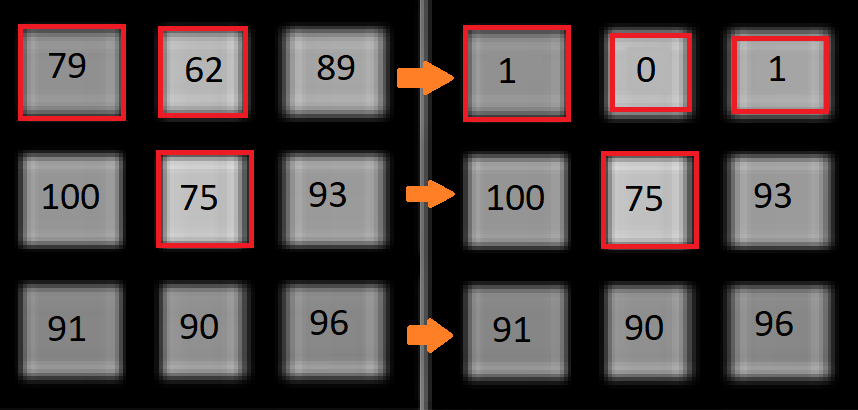
Submatricea de 3x3 selectata de algoritm

* Compară valorile(0-255) celor 8 pixeli exteriori cu valoarea pixelului din mijlocul matricei în sensul acelor de ceasornic.



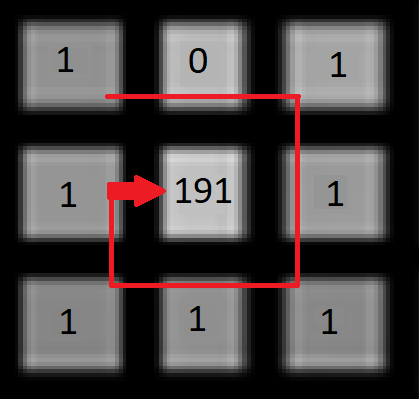
. Compararea valorilor pixelilor

* Dacă valoarea unui pixel raportată la valoarea pixelului central este mai mică, atunci valoarea acestuia se scuprascrie cu o noua valoare, 0 și 1 dacă aceasta este mai mare.



Suprascrierea valorilor biților cu echivalentul în binar rezultat din comparative

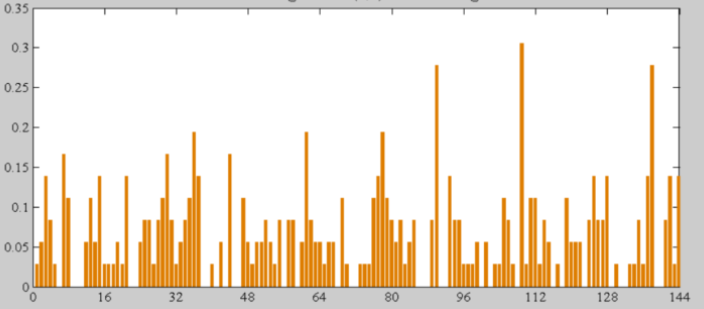
* Valorile obținute în final vor fi considerate un nou șir binary, iar valoarea acestuia în decimal va lua locul valorii pixelului din mijlocul matricii.



Valorile obtinute in urma comparatiei

(1011 1111)2=(191)10

* Se repetă pașii de mai sus pentru fiecare porțiune de 3x3 din imagine până când toată suprafața imaginii va fi acoperită.
* La final, toate valorile submatricilor de 3x3 se vor converti într-o histograma și se concatenează pentru a forma un vector ce descrie o trasatură(feature)

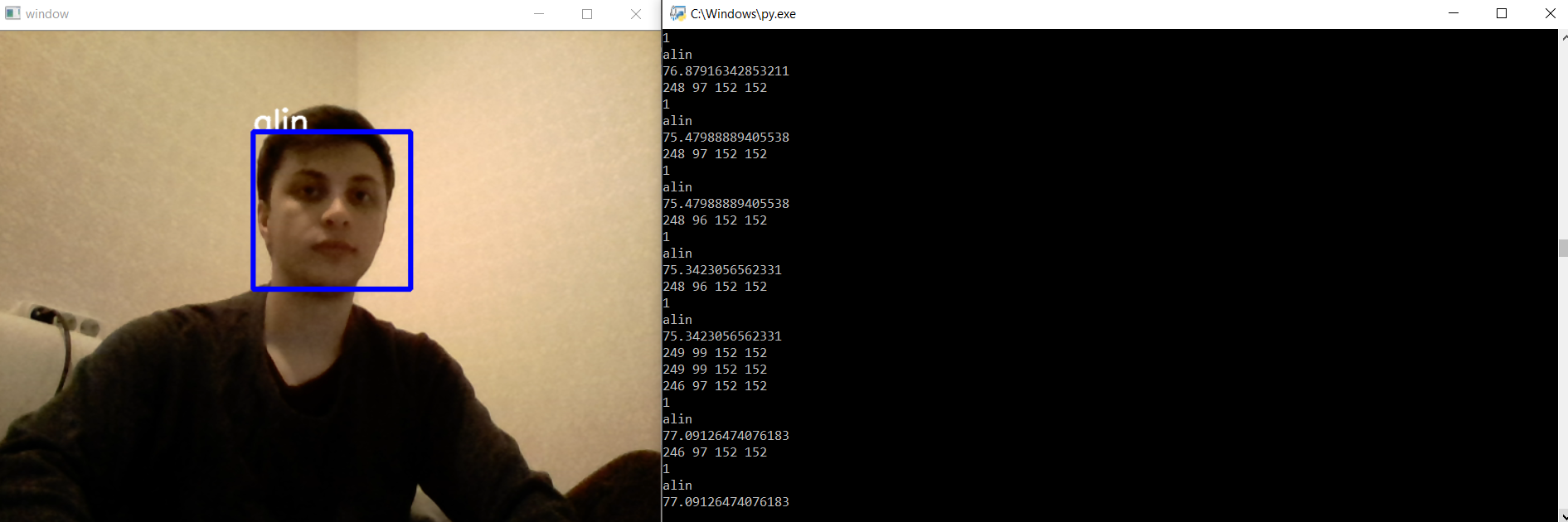


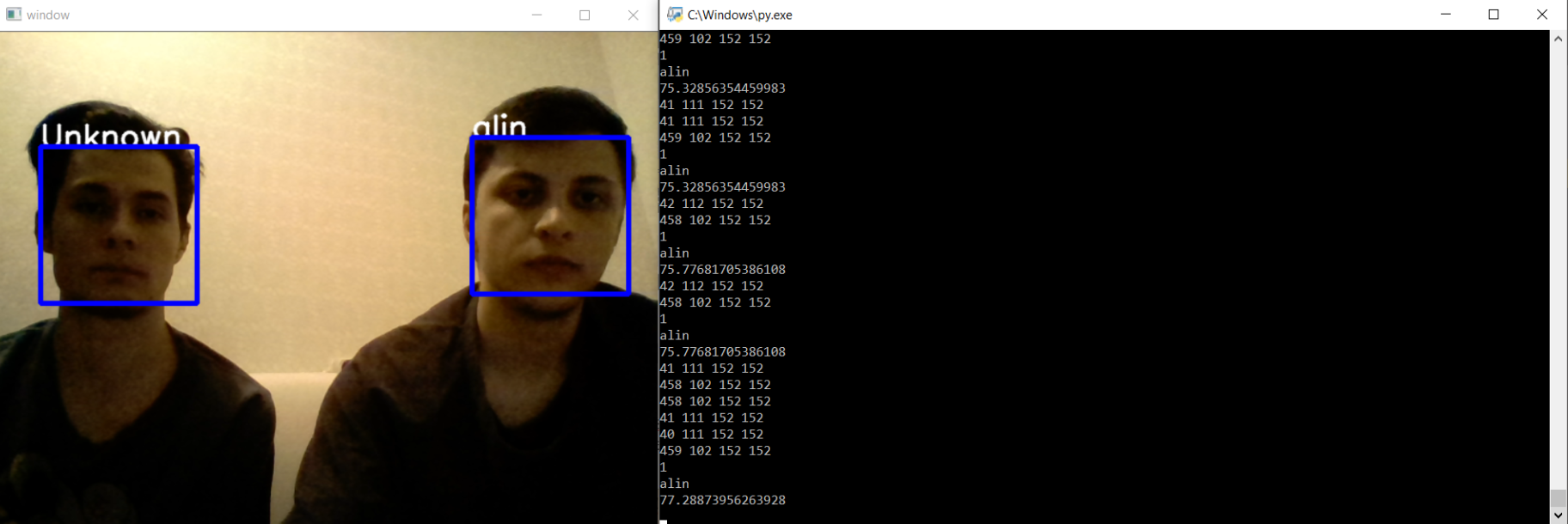
Histograma valorilor blocurilor de 3x3 pixeli din imagine

**Haar vs. LBP**

|  |  |  |
| --- | --- | --- |
|  | Avantaje | Dezavantaje |
| Haar | 1. Acuratete de detective mare  2.Rata scazuta de detectare a pozitivelor false | 1.Complexitate computationala crescuta  2.Timp necesar pentru antrenare crescut  3.limitari mari in ceea ce priveste iluminarea |
| LBP | 1. Complexitate scazuta  2.Rapiditate foarte mare  3.Robustete in detectarea fetelor indifferent de puterea iluminarii | 1 Acuratete scazuta  2. Rata de detectare a pozitivelor false crescuta |

În final am decis să folosim ceea ce este mai bun din fiecare, deci o combinație de haar (Clasificatorul de tip cascadă Haar din modulul OpenCV) și LBP(antrenat de noi, pe fețele noastre) pentru a obține performanțe și viteze acceptabile.





1. Concluzii

Pe parcursul proiectului am întampinat probleme de compatibilitate și de performanță a dispozitivelor. Raspberry Pi nu are suficientă rată de performanță și astfel nu a putut procesa imaginea pentru o recunoaștere optimă,de aceea am utilizat laptop-ul cu camera integrată, iar din punct de vedere hardware a rămas implementată doar funcționalitatea ca microcontroller sa fie activat prin serial atunci când persoana se află la o distanta rezonabilă. În rest, microcontroller-ul se află în starea de Sleep.

1. Anexe
   1. **Cod Arduino**

#define trigPin 13

#define echoPin 12

#define led 11

#define led2 10

void setup() {

Serial.begin (9600); // rata de transfer a datelor

pinMode(trigPin, OUTPUT);//seteaza pinul trigPin ca output;

pinMode(echoPin, INPUT); //seteaza pinul echoPin ca output;

pinMode(led, OUTPUT); //seteaza pinul led ca output;

pinMode(led2, OUTPUT); //seteaza pinul led2 ca output;

}

void loop() {

long duration, distance;

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = (duration/2) / 29.1;

if (distance <= 100){ // reprezinta conditia care decide daca trimite datele pe serial catre Raspberry Pi

//daca este adevarata conditia, atunci datele vor fi trimise pe serial iar led-ul rosu va fi stins si aprins led-ul verde,fapt ce confirma indeplinirea conditiei.

digitalWrite(led,HIGH);

digitalWrite(led2,LOW);

Serial.println("wake up");

}

//daca nu este adevarata conditia, atunci led-ul rosu ramane aprins iar pe serial nu vor fi trimise date.

else {

digitalWrite(led,LOW);

digitalWrite(led2,HIGH);

}

delay(1000); //la fiecare secunda se repeta acest ciclu

}

* 1. **Cod recunoastere**

|  |
| --- |
| import numpy as np |
|  | import cv2 |
|  | import pickle |
|  | # incarcam clasificatorii de tip haar respectiv LBP |
|  | face\_cascade = cv2.CascadeClassifier('cascades/data/haarcascade\_frontalface\_alt2.xml') # clasificatorul haar - recunoastere faciala |
|  | recognizer = cv2.face.LBPHFaceRecognizer\_create() # clasificatorul LBP |
|  | recognizer.read("trainer.yml") # Datele antrenate rezultate din antrenarea retelei - identificare faciala |
|  |  |
|  | # atribuire de tip value:key pentru vectorii imaginilor si cheia atribuita acestora |
|  | labels = {} |
|  | with open("labels.pickle", 'rb') as f: |
|  | original\_labels = pickle.load(f) |
|  | labels = {v:k for k,v in original\_labels.items()} |
|  |  |
|  | cap = cv2.VideoCapture(0) # initializare captura video |
|  |  |
|  | while(True): |
|  | ret, img = cap.read() # captura cadre |
|  | gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY) # convertirea imagini |
|  | faces = face\_cascade.detectMultiScale(gray, scaleFactor=1.5, minNeighbors=5) # folosim KNN K nearest neighbors |
|  |  |
|  | # specificarea unei zone de interes in functie de x,y = punctul de start w = lungime si h = inaltime |
|  | for (x,y,w,h) in faces: |
|  | print (x,y,w,h) |
|  | roi\_gray = gray[y:y+h, x:x+w] |
|  | roi\_color = img[y:y+h, x:x+w] |
|  | # in acest punct, reteaua ofera un scor de incredere prin care se atribuie fetei recunoscute un id\_ pentru a identifica fata gasita in imagine |
|  | id\_, conf = recognizer.predict(roi\_gray) |
|  | if conf >= 45 and conf <=85: |
|  | print(conf) |
|  | print(id\_) |
|  | print(labels[id\_]) |
|  | font = cv2.FONT\_HERSHEY\_SIMPLEX |
|  | name = labels[id\_] |
|  | stroke = 2 |
|  | color = (255, 255, 255) |
|  | # afiseaza id-ul fetei recunoscute, daca scorul de incredere este cuprins intre 45 si 85 |
|  | cv2.putText(img, name, (x,y), font, 1, color, stroke, cv2.LINE\_AA) |
|  | # evidentiaza zona de interes print-o forma rectangulara |
|  | color = (255, 0, 0) |
|  | stroke = 4 |
|  | end\_cord\_x = x + w |
|  | end\_cord\_y = y + h |
|  | cv2.rectangle(img, (x, y), (end\_cord\_x, end\_cord\_y), color, stroke) |
|  |  |
|  |  |
|  |  |
|  | # afisam rezultatul final al progesarii pe imaginea RGB |
|  | cv2.imshow("window", img) |
|  | # conditie care trebuie indeplinita pentru a iesi din bucla infinita. |
|  | if cv2.waitKey(1) & 0xFF == ord('q'): |
|  | break |
|  | cv2.destroyAllWindows() |
|  | cap.release() |

1. Bibliografie
2. [**https://speed.pub.ro/speed3/wp-content/uploads/2017/07/2017\_Proiect\_Diploma\_GEANĂ\_Sorin-Valentin.pdf**](https://speed.pub.ro/speed3/wp-content/uploads/2017/07/2017_Proiect_Diploma_GEANĂ_Sorin-Valentin.pdf)
3. [**https://www.academia.edu/31460585/RETELE\_NEURONALE\_ARTIFICIALE\_PENTRU\_RECUNOASTERE\_OPTICA**](https://www.academia.edu/31460585/RETELE_NEURONALE_ARTIFICIALE_PENTRU_RECUNOASTERE_OPTICA)
4. [**https://ro.wikipedia.org/wiki/Raspberry\_Pi**](https://ro.wikipedia.org/wiki/Raspberry_Pi)
5. [**https://cdn.sparkfun.com/datasheets/Dev/RaspberryPi/RPiCamMod2.pdf**](https://cdn.sparkfun.com/datasheets/Dev/RaspberryPi/RPiCamMod2.pdf)
6. [**https://www.optimusdigital.ro/ro/senzori-senzori-ultrasonici/9-senzor-ultrasonic-hc-sr04-.html**](https://www.optimusdigital.ro/ro/senzori-senzori-ultrasonici/9-senzor-ultrasonic-hc-sr04-.html)
7. [**http://www.bel.utcluj.ro/dce/didactic/sisd/SISD\_curs\_6\_Retele\_Neuronale\_Artificiale.pdf**](http://www.bel.utcluj.ro/dce/didactic/sisd/SISD_curs_6_Retele_Neuronale_Artificiale.pdf)
8. <https://raw.githubusercontent.com/tbird20d/grabserial/master/grabserial>
9. <https://maker.pro/raspberry-pi/tutorial/how-to-connect-and-interface-raspberry-pi-with-arduino>
10. <https://www.raspberrypi.org/blog/raspberry-pi-and-arduino/>
11. <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>
12. <https://arduinobasics.blogspot.com/2012/11/hc-sr04-ultrasonic-sensor.html>

**Github:** [**https://github.com/Akim2222/Proiect\_SE\_SI.git**](https://github.com/Akim2222/Proiect_SE_SI.git)