



Universitatea „Politehnica” Timișoara  
Facultatea de Electronică, Telecomunicații și  
Tehnologii Informaționale Timișoara



# **LUCRARE DE DISERTAȚIE**

## **Studiul și îmbunătățirea comportamentului uman într-o casă inteligentă**

Coordonator:

**Conf. Univ. Dr. Ing. Muguraș Mocofan**

Student:

**Ing. Carol Alin Radi**

**Septembrie 2022**

# CUPRINS

<b>Capitolul 1. Analiza algoritmilor și sistemelor ce echipează o Casă Inteligentă .....</b>	<b>4</b>
<b>1.1 Inteligența Ambientală în Case Inteligente .....</b>	<b>4</b>
<b>1.1.1 Detectare .....</b>	<b>6</b>
<b>1.1.2 Raționament .....</b>	<b>7</b>
<b>1.2. Algoritmi de predicție în Casele Inteligente .....</b>	<b>8</b>
<b>1.2.1. Arhitectura casei inteligente .....</b>	<b>8</b>
<b>1.2.2 Modelul de mobilitate .....</b>	<b>9</b>
<b>1.2.3 Algoritmi de predicție .....</b>	<b>11</b>
<b>1.3. Securitatea unei Case Inteligente utilizând recunoașterea facială .....</b>	<b>17</b>
<b>1.3.1 Generalități .....</b>	<b>17</b>
<b>1.3.2 Mod de aplicare a recunoașterii faciale într-o casă inteligentă .....</b>	<b>18</b>
<b>1.3.3. Avantajele sistemului de securitate într-o casă inteligentă .....</b>	<b>20</b>
<b>Capitolul 2. Python cu librării de Inteligență Artificială și Viziune Computerizată .....</b>	<b>21</b>
<b>2.1. Internet of Things (IoT) cu Python .....</b>	<b>24</b>
<b>2.1.1. Cele mai bune soluții pentru IoT în Python .....</b>	<b>25</b>
<b>2.1.2. IoT Backend cu Python .....</b>	<b>27</b>
<b>2.2. Librării pentru AI în Python .....</b>	<b>29</b>
<b>2.2.1 Cele mai bune biblioteci Python pentru Machine Learning și AI .....</b>	<b>31</b>
<b>2.3. Librăria OpenCV din limbajul Python .....</b>	<b>36</b>
<b>2.3.1. O scurtă introducere despre OpenCV .....</b>	<b>36</b>
<b>2.3.2. Aplicații OpenCV .....</b>	<b>37</b>
<b>2.3.3. OpenCV în automatizarea caselor IoT .....</b>	<b>39</b>
<b>Capitolul 3. Sensorii Arduino. Tipuri și aplicații .....</b>	<b>42</b>
<b>3.1. Ce sunt senzorii Arduino? .....</b>	<b>42</b>
<b>3.1.1 Aplicații ale senzorilor Arduino .....</b>	<b>43</b>
<b>3.2. Exemple de senzori Arduino .....</b>	<b>44</b>
<b>Capitolul 4 - Descriere soluție de implementare .....</b>	<b>57</b>
<b>4.1. Algoritm de predicție și generarea comportamentului uman .....</b>	<b>57</b>
<b>4.2. Recunoașterea emoțiilor umane folosind Inteligența Artificială .....</b>	<b>64</b>

<b>4.3. Recunoașterea facială în casele inteligente .....</b>	<b>68</b>
<b>4.3.1 Conceptul de bază al algoritmului de cascadă HAAR.....</b>	<b>68</b>
<b>4.3.2 Detecția HAAR-Cascade în OpenCV .....</b>	<b>69</b>
<b>4.3.3. Recunoașterea feței folosind OpenCV.....</b>	<b>71</b>
<b>CONCLUZII .....</b>	<b>77</b>
<b>BIBLIOGRAFIE.....</b>	<b>78</b>

## Capitolul 1. Analiza algoritmilor și sistemelor ce echipază o Casă Inteligentă

Odată cu trecerea timpului, internetul și sistemele comunicaționale au cunoscut o reală evoluție. Datorită acestui fapt, conceptul de *Internet of Things (IoT)* a devenit din ce în ce mai discutat. Ceea ce, pare un lucru benefic atât pentru noi oamenii, adică cei care ajungem să ne folosim de el, cât și pentru specialiști, respectiv cei care se ocupă în mod direct de conceperea lui. Dar, această creștere rapidă nu a adus doar beneficii, ci a venit la pachet și cu câteva provocări. Iar înainte să vedem care sunt acestea, trebuie să aflăm ce presupune termenul *Internet of Things*.

Așadar, IoT este un sistem complex, compact, care presupune utilizarea unui mijloc de legătură, mai exact Internetul, cu scopul de a realiza o interconexiune între diferite dispozitive, protocoale, canale de comunicație, formând astfel o rețea. Pe lângă acest aspect, *Internet of Things* permite oamenilor și obiectelor să fie conectate în orice loc și la orice oră, folosind căile și serviciile generale, cunoscute de toată lumea. Cu alte cuvinte, dispozitivele inteligente au simțuri senzoriale cu mediul înconjurător, înțeleg schimbările ce se produc, se analizează reciproc și interacționează unul cu celălalt, doar pe baza standardelor și internetului.

### 1.1 Inteligența Ambientală în Case Inteligente

În ziua de astăzi, IoT este utilizat, pe scară largă, în diverse domenii, precum transport și asistență medicală, astfel că, în viitor, se preconizează că aproape orice dispozitiv va avea o interfață legată la internet. Acest domeniu acoperă o multitudine de zone și asigură rezolvarea problemelor legate de hardware, securitate, fiabilitate, interoperabilitate și chiar de partajarea datelor, mai ales că, de-a lungul anilor, au tot apărut inițiative privind arhitecturile pentru rezolvarea acestora. Plus de asta, există și alte domenii de studiu, de exemplu: *cloud computing*, *context awareness* sau *managementul datelor și al serviciilor*, care sunt asociate sistemului IoT.

Senzorii, care pot fi integrați cu ușurință de oricine și la orice „instrument”, joacă și ei un rol extrem de important în acest proces. Adică, ajută la dezvoltarea, într-un ritm alert, a ambianței inteligente (AMI). La ce se referă mai exact acest concept? AMI se definește ca fiind capacitatea sistemului de a percepe mediul înconjurător și de a răspunde de prezența oamenilor la anumite evenimente sau în condiții determinate de la bun început, în funcție de comportamentul lor din diferite situații.

De-a lungul vremii, cercetări în acest domeniu s-au tot făcut. De exemplu, în anul 2001, Comisia Europeană a găsit o metodă de analizare a ambianței inteligente. Un factor predominant în apariția și dezvoltarea acestui domeniu îl constituie evoluția tehnologiei, pe care am menționat-o încă de la începutul lucrării de față. La începuturi, calculatoarele aveau un preț exorbitant și erau greu de utilizat, devenind astfel o resursă rară și prețioasă.

Drept consecință, ajunseseră să fie folosite, pe rând, de mai multe persoane. Pe măsură ce industria a progresat și costurile au scăzut, un utilizator ar fi putut accesa mai mult de un singur produs electronic asemănător. Prin urmare, astăzi, accesul la mai multe computere nu înseamnă neapărat deținerea unui PC sau a unui laptop. Din momentul în care a început etapa de miniaturizare a microprocesoarelor, puterea de calcul a fost încorporată în electrocasnice (mașina de spălat, frigiderul, cuptorul cu microunde etc). Această disponibilitate extinsă a resurselor a stat la baza genezei domeniului Ambient Intelligence (AMI).

În zilele noastre, experiența utilizatorilor cu propriile calculatoare a creat un context favorabil pentru întreg domeniu, în care așteptările de la aceste sisteme continuă să crească, iar teama de a le utiliza se află într-o evidentă scădere. Concomitent cu această diferență privind modul în care societatea percepe tehnologia, există și o schimbare la nivelul gestionării serviciilor. Pentru a da și un exemplu concludent acestui caz, aducem în discuție descentralizarea asistenței medicale.

Ideea de inteligență ambientală nu este una nouă, însă ceea ce putem adăuga pentru a oferi acel statut de noutate se referă la faptul că, în momentul de față, AIM se caracterizează ca fiind o disciplină proprie, cu un set unic de contribuții. Cred că mulți dintre noi am văzut filme SF, în care ușile se deschid singure atunci când o persoană se apropie de ele, fără ca numele lor să fie precizat.

Aceste tehnologii păreau demne de Epoca de Piatră, dar, treptat, odată cu evoluția tehnologică, unele caracteristici au devenit pură realitate, luând astfel naștere industria de inteligență ambientală.

Din punct de vedere tehnic, majoritatea dintre noi locuim astăzi în case care, în anii 60, erau considerate a fi „inteligente” și la un preț rezonabil. Pentru noi, termostatele și senzorii de mișcare care controlează lumina au ajuns să fie un lucru pe cât se poate de obișnuit. Chiar și capacitatea de a lega senzori de mișcare la o alarmă de securitate pentru a detecta intrușii, nu mai reprezintă un interes major, mai ales că interacționăm în mod regulat cu astfel de facilități.

Deci, putem spune că ceea ce generațiilor anterioare li se păreau idei aflate la limita realității, pentru noi, generația nouă, sunt normale.

Așadar, concluzionând toate cele prezentate mai sus, putem afirma că scopul unui mediu ambiental inteligent este acela de a îmbunătăți relația dintre mediul înconjurător și tehnologia, reprezentată de senzori și dispozitive interconectate. Cum sistemele ambientale inteligente au o conexiune strânsă cu multe arii de informatică, vom împărți această tehnologie în următoarele două etape: detectare, raționament. Și acum, le vom lua la rând și vom afla ce presupune fiecare.

### **1.1.1 Detectare**

Deoarece ambientul inteligent este conceput pentru medii fizice reale, utilizarea eficientă de senzori este vitală. Fără componenta fizică, adică cea care să permită unui agent să se activeze senzorial și să acționeze asupra unui mediu, ne-am confrunta doar cu un grup de algoritmi teoretici, care nu ar avea nicio utilizare practică. Cu alte cuvinte, senzorii sunt cheia ce leagă puterea de calcul disponibilă de aplicațiile fizice. Prin urmare, vom afla în paragrafele următoare de ce sunt ei atât de importanți în acest domeniu.

În primul rând, algoritmi de inteligență ambientală se bazează pe date senzoriale din lumea reală. Astfel că, algoritmul percepe mediul și folosește aceste informații pentru a raționa ceea ce a primit din exterior, evidențiind acțiunile disponibile în vederea eventualelor schimbări. Percepția de mai sus se realizează folosind o varietate de senzori, care, fiind foarte mici, pot fi integrați în aproape orice aplicație AMI. Scopul proiectării unor asemenea senzorii a fost acela de a măsura pozițiile, de a detecta umiditatea, lumina, sunetul, temperatura, presiunea sau direcția, dar și de a monitoriza sănătatea protagoniștilor.

În al doilea rând, înțelegerea datelor provenite de la senzori reprezintă o sarcină complexă. Datele senzoriale vin cu caracteristici unice, care își asumă rolul de a provoca tehnicile convenționale să analizeze informațiile primite. Ele generează volumuri mari de date, imposibil de interpretat manual. Adesea trebuie să fie rezolvate din mers sau chiar direct din flux.

Descoperim astfel că pot avea un caracter spațial sau o componentă temporală.

În al treilea rând, atunci când se analizează datele senzorilor sistemelor AMI, se folosește un model centralizat sau distribuit. Senzorii din modelul centralizat transmit date către un server central, care colectează, analizează și conectează informațiile primite.

Spre deosebire de acesta, în modelul distribuit, fiecare sensor are capabilități de procesare și efectuare a calculelor locale, proces realizat înainte de a comunica rezultatele parțiale a celorlalte

noduri din rețeaua de senzori. În ambele cazuri, datele de senzori sunt colectate din diferite surse și combinate ulterior, cu scopul de a produce informații mai precise și mai complete decât cele individuale.

Prin urmare, procesul de detectare împreună cu cel de acționare (o altă etapă importantă în aceste tehnologii) oferă legături între algoritmi inteligenți și lumea reală în care aceștia operează. Pentru a face ca algoritmi de față să fie receptivi, benefici și adaptabili oricărui utilizator, trebuie să aibă loc o serie de tipuri de raționament. În secțiunea de mai jos vom vedea la ce se referă această etapă, ce presupune, dar și ce includ mai exact respectivele exemple.

### **1.1.2 Raționament**

Cum am menționat anterior, avem mai multe tipuri de raționament. Acestea, la rândul lor, sunt formate din câteva procese. Cele mai întâlnite și des folosite de utilizatori sunt: modelarea, predicția, recunoașterea activităților și luarea deciziilor. Iar fiecare proces enumerat răspunde de anumite caracteristici. Mai jos vom explica cum funcționează acestea și care este rolul fiecăreia.

O primă caracteristică definitorie este dată de capacitatea de a modela comportamentul utilizatorului. Aceasta separă algoritmi de calcul general de cei care răspund utilizatorilor. Dacă se poate construi un astfel de șablon, poate fi folosit cu scopul de a personaliza manifestările software-ului de ambient inteligent față de consumator. În cazul în care rezultă într-o bază solidă și clară, poate sta, cu ușurință, la baza unui model de securitate, care ajută la detectarea atât a anomaliilor, cât și a modificărilor în tiparul de comportament.

O a doua caracteristică ne este oferită de studiile realizate pe tot parcursul timpului. În ultimii ani, au apărut tehnologii care vin în sprijinul inteligenței ambientale. Foarte puține lucruri se pot realiza într-un sistem AMI fără o referință la reperele spațio-temporale ale evenimentelor semnificative. Pentru ca un sistem să ia decizii, trebuie să aibă cunoștințe despre locul pe care îl au sau l-au avut utilizatorii într-o perioadă de timp.

Aceste perspective împreună cu alte informații vor oferi indicii importante asupra tipului de activități, dar și a răspunsurilor adecvate. Raționamentul spațial și temporal este util în a înțelege elementele cheie ale unei situații în plin proces de dezvoltare. Și pentru a fi siguri că noțiunile de mai sus sunt clare, vom oferi un exemplu destul de simplu. Să presupunem că monitorizăm toate activitățile cu risc crescut de pericol din casă, respectiv tot ce este legat de electricitate sau gaz.

Ori de câte ori cineva pornește aragazul, pleacă din încăpere și îl lasă nesupravegheat mult timp, sistemul este obligat să ia măsuri. Printre acestea, putem menționa notificarea utilizatorului sau sistarea aragazului.

În concluzie, acest subcapitolul a avut rolul de a ne introduce în lumea caselor inteligente, de a vedea, în mare, cu ce se ocupă și din ce sunt alcătuite, dar și de a afla care sunt principalele etape ce stau la baza unor astfel de tehnologii. În paginile următoare, vom detalia acest concept și vom descoperi ce limbaje de programare sunt utilizate în crearea mediului ambientat inteligent.

## 1.2. Algoritmi de predicție în Casele Inteligente

Cum am văzut în subcapitolul precedent, casa inteligentă se definește ca fiind un spațiu de locuit sau de desfășurarea a activităților profesionale, care interacționează într-un mod natural și se adaptează ocupantului. Adaptarea se referă la faptul că acesta învață să se recunoască și să se schimbe în funcție de identitatea și activitatea desfășurată de ocupant, cu o intervenție minimă din partea acestuia. Prin urmare, un agent de casă inteligentă trebuie să fie capabil să prezică tiparele de mobilitate și utilizarea dispozitivelor de către locuitori [7]. Cu ajutorul acestor previziuni, casa poate direcționa cu precizie mesajele și informațiile multimedia și poate automatiza activități care altfel ar fi efectuate manual de către locuitori. Casa inteligentă se comportă ca un agent rațional, percepând starea casei prin intermediul senzorilor și acționând asupra mediului înconjurător prin intermediul unor efectoare. Scopul locuințelor inteligente este de a maximiza confortul și siguranța, de a optimiza utilizarea energiei și de a elimina activitățile repetitive obositoare [1].

### 1.2.1. Arhitectura casei inteligente



Figura 1.1: Arhitectura unei case inteligente



Arhitectura unei astfel de case poate fi împărțită în 4 niveluri:

**Nivelul 1** poartă denumirea de „nivel fizic”. Acesta conține partea hardware care, la rândul său, include diferite dispozitive generale, dispozitive pentru internet și senzori.

**Nivelul 2** este cel de comunicare. El include software-ul destinat formării și comunicării între utilizatori, casa inteligentă și sursele externe.

**Nivelul 3** se ocupă de informații. Acest nivel colectează, stochează și generează cunoștințele folosite pentru luarea deciziilor.

Iar ultimul nivel, **nivelul 4**, are în atribuție deciziile. Selectează secțiunea pe care trebuie să o ia, în funcție de informațiile provenite de la celelalte niveluri [7].

În funcție de cele patru niveluri menționate mai sus, putem discuta atât despre un proces de percepție, cât și despre un proces de acțiune. Să aflăm și ce presupune fiecare. Percepția este un proces care funcționează de jos în sus. Adică, senzorii monitorizează casă și, în cazul în care este necesar, transmite informații mai departe prin nivelul de comunicare.

Baza de date înregistrează astfel informația în nivelul de informații și actualizează conceptele și predicțiile în funcție de datele primite, alertând apoi ultimul nivel (cel de decizie) de prezența unor noi date [7].

Execuția, pe de cealaltă parte, este un proces care rulează din partea de sus a nivelului în jos. Astfel, nivelul de decizie selectează o acțiune și transmite decizia luată către nivelul de informație. După actualizarea acestuia, nivelul de comunicare transmite acțiunea primului nivel, iar la rândul său, alocă secțiuni destinate execuției [4].

### 1.2.2 Modelul de mobilitate

O casă inteligentă îndeplinește diferite sarcini, așa că trebuie să aibă anumite dotări. În primul rând, prezența conectivității foarte bună este o necesitate. Drept urmare, va trebui să existe o conexiune wireless încorporată în toată casa. În al doilea rând, casa va fi împărțită pe zone sau sectoare, iar managementul locației va implica urmărirea persoanelor din zonele arondate.

Mai mult decât atât, locația actuală a persoanelor poate fi cel mai ușor de aflat. Ea se calculează în funcție de locația anterioară [7].

Rețeaua casei inteligente poate fi reprezentată de un graf conex de grade mărginit,  $G = (\mathcal{Z}, \epsilon)$ , unde nodul setat  $\mathcal{Z}$  reprezintă zonele, iar setul de muchii  $\epsilon$  reprezintă vecinătatea (pereți, holuri etc.) dintre perechile de zone. În Figurate mai jos avem și reprezentarea clară a informațiilor precizate [7].

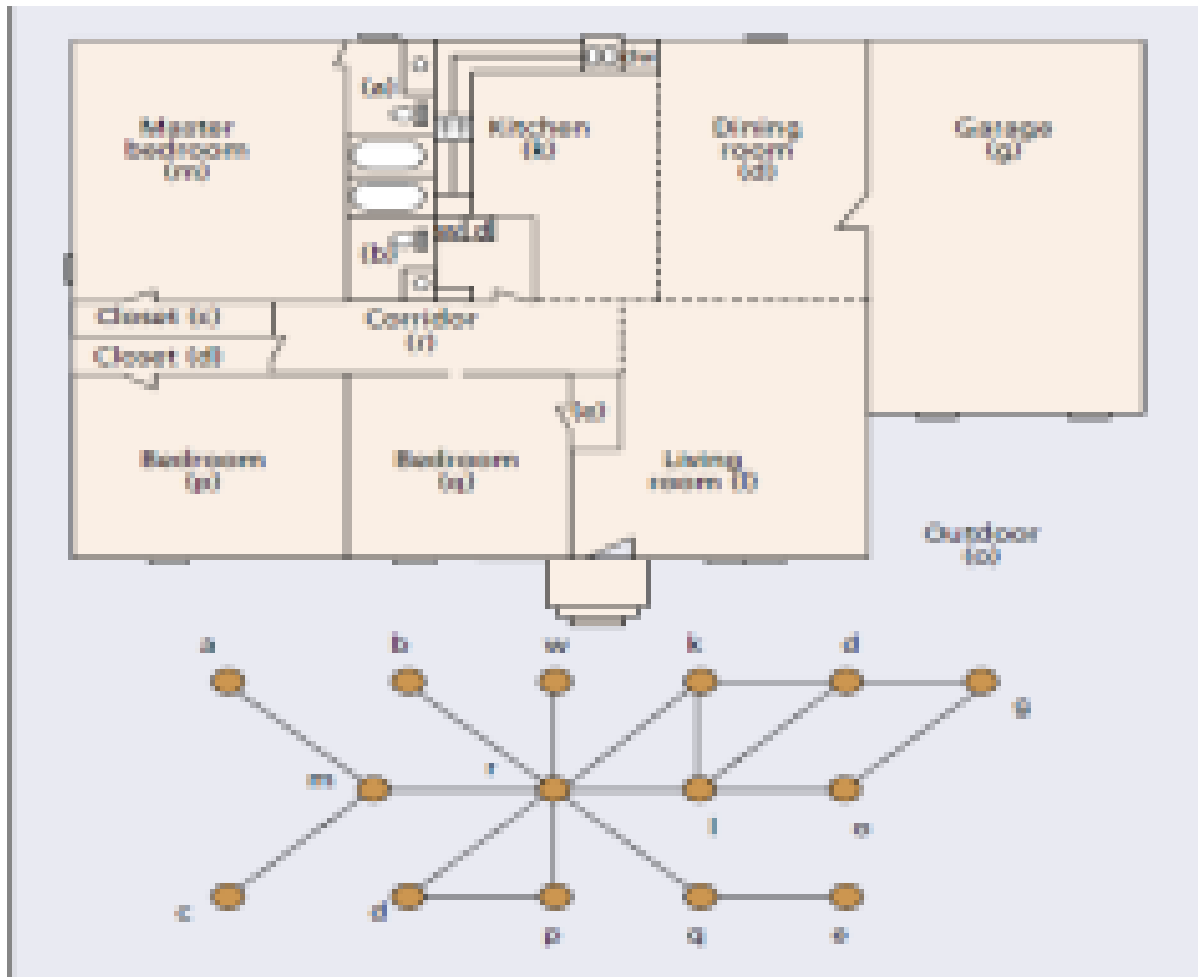


Figura 1.2: Reprezentarea unei case inteligente printr-un graf.

Determinarea locației locuitorului este o schemă bazată exclusiv pe mișcare. O actualizare este generată ori de câte ori este detectată o trecere a limitei unei zone sau este observată o activitate a utilizatorului de internet diferită față de ultima activitate.

Istoricul mișcărilor unui utilizator este reprezentat de un șir de caractere " $v_1v_2v_3...$ " cu simboluri din alfabetul  $\mathcal{Z}$ , unde  $\mathcal{Z}$  este setul de zone din casă, iar  $v_i$  reprezintă id-ul zonei raportat de către actualizarea  $i$ -th.

Să luăm în considerare istoricul mișcărilor lui Ann, în casa inteligentă, pe parcursul unei zile întregi, este generat sub forma mamcmrkdtkd- googdk., unde simbolurile care denotă istoricul mișcărilor pot fi interpretate cu ajutorul schemei din Figura 1.3. [7] .

Tabel 1.3. Presupunerea tacită este că mișcarea unui locuitor este doar o reflectare a tiparelor vieții sale, iar acestea pot fi învățate.

Zone	Activity
m	Wake up in master bedroom
a	Go to attached bathroom
m	Back in bedroom
c	Change in closet
m	Back in bedroom
r	Out of bedroom
k	Go to kitchen to make breakfast
d	Go to dining room to eat
k	Back to the sink at kitchen
d	Walk to the garage through dining room
g	Start the car at gatage
o	Drive away through outdoor area
o	Drive back through outdoor area
g	Back to garage
d	Enter through dining room
k	To kitchen for a snak and drink

Acest lucru definește faza de învățare care, la rândul său, ajută la luarea deciziilor atunci când se detectează reapariția tiparelor [4].

### 1.2.3 Algoritmi de predicție

Algoritmul de predicție urmărește să construiască un predictor sau un estimator universal pentru a determina următoarea acțiune a utilizatorului. Schema creează un dicționar de id-uri de zonă tratate ca simboluri de caractere și utilizează dicționarul pentru a aduna statistici bazate pe contexte sau fraze din istoricul mișcărilor.

Problema prezicerii următorului simbol (reprezentând o acțiune a utilizatorului) într-o secvență de intrare poate fi definită formal după cum urmează: Fie  $\Sigma$  setul de simboluri de intrare posibile și fie  $A = a_1 \dots a_n$  cu  $a_j \in \Sigma$  o secvență de simboluri de intrare din care primele  $i$  simboluri,

adică  $a_1 \dots a_i$ . Un algoritm de predicție decide la început dacă este capabil să facă o predicție și, în caz afirmativ, returnează probabilitatea pentru fiecare simbol  $x \in \Sigma$ , ca  $x$  să fie următorul element din secvența de intrare. Aceste valori definesc o distribuție de probabilitate condiționată  $P$  pe  $\Sigma$ , unde  $P(x|a_1 \dots a_i)$  este probabilitatea pentru subsetul singleton din  $\Sigma$  care îl conține pe  $x$ [7].

Există două moduri de calculare a probabilităților: la cerere sau în direct. Algoritmii care utilizează prima metodă mențin o structură de date pentru a calcula probabilitățile. Ei actualizează structura de date după fiecare simbol din secvența de intrare, în timp ce algoritmii live actualizează ei înșiși distribuțiile de probabilitate [3].

#### A. Algoritmul LZ78

Compresia de date LZ78 este un algoritm de analiză incrementală bazat pe modelul Markov. Acest algoritm a fost interpretat ca o schemă de modelare universală care calculează secvențial probabilități empirice în fiecare context al datelor; probabilitățile generate reflectă contextele observate de la începutul secvenței analizate până la simbolul curent.

Lungimea codului LZ a oricărei secvențe individuale atinge entropia empirică markoviană pentru orice ordine Markov finită. Acest algoritm analizează un șir de intrare " $x_1, x_2 \dots x_i$ " în  $c(i)$  subșiruri " $w_1, w_2, \dots, w_{c(i)}$ " astfel încât pentru toate  $j > 0$ , prefixul subșirului  $w_j$  (adică toate caracterele din  $w_j$ , mai puțin ultimul) este egal cu un anumit  $w_i$  pentru  $1 < i < j$ .

Din cauza acestei proprietăți de prefix, subșirurile analizate pot fi păstrate în mod eficient într-o trie. LZ78 este utilizat doar ca un sistem care descompune o anumită secvență (șir) de stări în fraze[7].

```

initialize dictionary := null
initialize phrase w := null
loop
  wait for next symbol v
  if ((w.v) in dictionary):
    w := w.v
  else
    add (w.v) to dictionary
    w := null
    increment frequency for every
    possible prefix of phrase
  endif
forever

```

Se consideră secvența de simboluri de intrare  $nx = "aaabababbbbaabccddcbaaaa"$ . O analiză LZ78 a acestui șir de simboluri de intrare, conform algoritmului LZ78 menționat mai sus, ar produce următorul set de fraze: "a,aa,aa,b,ab,bb,bba,bba,abc,c,d,dc,dc,ba,aaa". Dicționarul generat este prezentat mai jos. Așa cum s-a descris mai sus, acest algoritm păstrează statistici pentru toate contextele observate în cadrul frazelor, iar aceste statistici sunt stocate într-o trie (Figura 1.4.)

Tabel 1.3: Dicționar simboluri și faze

a	5
aa	2
b	4
ab	2
bb	2
bba	1
abc	1
c	1
d	2
dc	1
ba	1
aaa	1

Algoritmul LZ78 suferă de o problemă de convergență lentă. Acest lucru se datorează faptului că se pierd toate informațiile care traversează limitele frazei. În multe situații, vor exista modele semnificative care traversează limitele frazei, iar aceste modele vor afecta următorul simbol din secvență.

Vom da și un exemplu pentru a fi noțiunile mai clare. În șirul de mai sus (aaabababbbbaabccddcbaaaa), al 6-lea simbol (bba) și al 7-lea (abc) nu formează fraza baab[7].

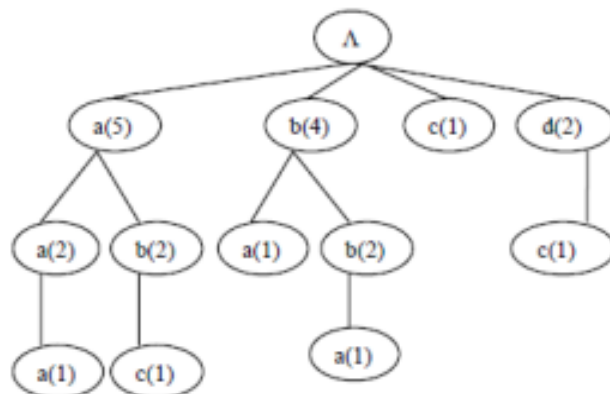


Figura 1.4: Trie formată prin analiza LZ78 a șirului de caractere "aaabababbbbaabccddcbaaaa" (model Markov de ordinul 2)

### *B. Algoritmul activ LeZi*

Activ LeZi este un algoritm la cerere care se bazează pe modele Markov și care stochează în principal frecvența modelelor de intrare într-o trie în conformitate cu algoritmul de compresie LZ78. Cantitatea de informații care se pierde prin limitele frazei crește rapid atunci când există o creștere a numărului de stări observate în secvența de intrare. Această problemă poate fi depășită prin menținerea unei ferestre de lungime variabilă a simbolurilor văzute anterior [7].

Lungimea celei mai lungi fraze observate într-o analiză LZ78 clasică este aleasă ca fiind egală cu lungimea ferestrei în fiecare etapă. Motivul pentru care se alege această dimensiune a ferestrei este acela că algoritmul LZ78 construiește, în esență, un model Markov de ordinul  $k-1$ , unde  $k$  este egal cu lungimea celei mai lungi fraze LZ78 văzute până în prezent [7].

În cadrul acestei ferestre, putem acum aduna statistici pentru toate contextele posibile. Astfel, se construiește o aproximare mai bună a modelului Markov de ordinul- $k$ , deoarece a capturat informații despre contextele din secvența de intrare care traversează limitele frazei în parsarea clasică LZ78. Prin urmare, obținem o rată de convergență mai bună către predictibilitatea optimă, precum și o mai mare acuratețe de predicție[5].

#### **Caracteristicile lui Active LeZi sunt următoarele:**

Un model Markov de ordin crescător atinge o predictibilitate optimă a FS, datorită caracterului optim al LZ78. Pe măsură ce lungimea celei mai lungi fraze LZ78 crește, Active LeZi stochează din ce în ce mai multe informații; pe măsură ce secvența de intrare (experiența) crește, algoritmul este mai performant.

Aceasta este o caracteristică de dorit pentru orice algoritm de învățare. Tria din Figura1.5. prezintă analiza Active LeZi a aceluiași șir de caractere "aaabababbbbaababccddcbaaaa" conform algoritmului Active LeZi[4].

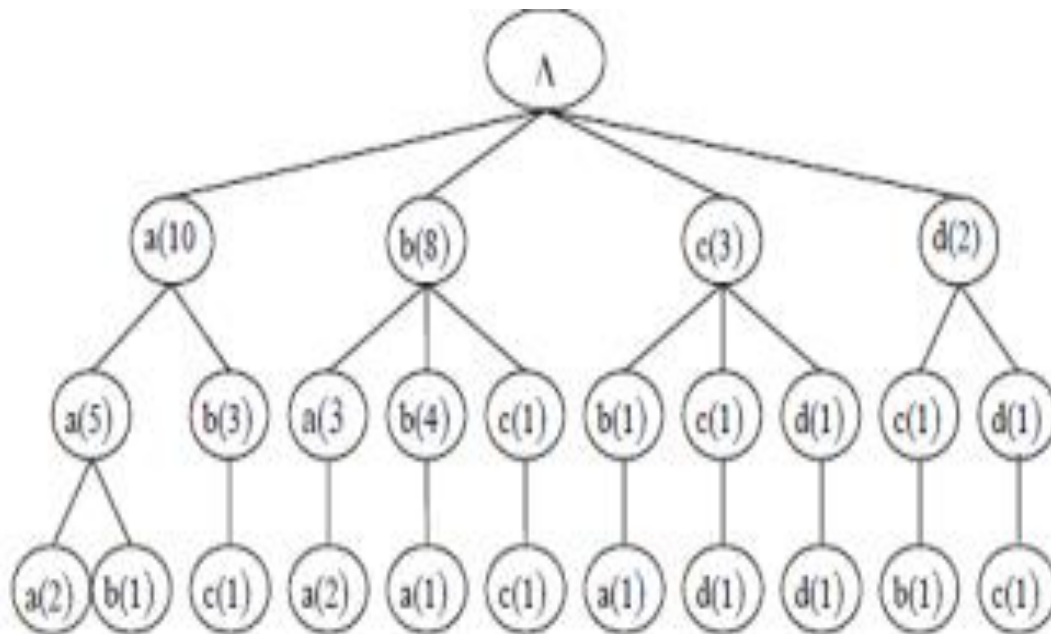


Figura 1.5: Tria formată prin analiza Active LeZi a șirului "aaabababbbbaabccddcbaaaa"  
 Vom da și un exemplu de algoritm, cu scopul de a vedea cum se scrie în cod.

```

initialize dictionary:= null
  initialize phrase w:= null
  initialize window:= null
  initialize Max_LZ_length = 0
loop
  wait for next symbol v
  if ((w.v) in dictionary):
    w:= w.v
  else
    add (w.v) to dictionary
    update    Max_LZ_length    if
    necessary
    w:= null
  endif
  add v to window
  if (length(window) > Max_LZ_length)
    delete window[0]
  endif
  Update frequencies of all possible
    contexts within window that includes v
forever

```

Algoritmul activ LeZi construiește un model Markov de ordinul  $k$ . Strategia de predicție prin potrivire parțială (PPM) de excludere pentru a aduna informații de la modelele de ordinul 1 până la  $k$  pentru a atribui simbolului următor valoarea sa de probabilitate. Luați în considerare exemplul secvenței anterioare "aaabababbbbaabccddcbaaaa".

Fereastra menținută de Active LeZi reprezintă setul de contexte utilizate pentru a calcula probabilitatea simbolului următor. În exemplu, se utilizează ultima frază "aaa" (care este, de asemenea, fereastra ALZ curentă). În cadrul acestei fraze, contextele care pot fi utilizate sunt toate sufixele din cadrul frazei, cu excepția ferestrei în sine (adică "aa", "a" și contextul nul) [7].

Să presupunem că se calculează probabilitatea ca următorul simbol să fie un "a". Se observă că (referință Figura 1.5), un "a" apare de două din cele cinci ori când apare contextul "aa", celelalte cazuri producând două rezultate nule și un "b". Prin urmare, probabilitatea de a întâlni un "a" în contextul "aa" este de  $2/5$ , iar acum ne întoarcem la ordinul-1 cu probabilitatea de  $2/5$ . În contextul de ordinul 1, vedem un "a" de cinci ori din zece ori când vedem contextul "a", iar în celelalte cazuri, vedem două rezultate nule. Prin urmare, prezicem "a" la contextul de ordinul-1 cu o probabilitate de  $5/10$  și scăpăm la modelul de ordinul-0 cu o probabilitate de  $2/10$ . La modelul de ordin 0, vedem "a" zece din cele 23 de simboluri văzute până acum și, prin urmare, prezicem "a" cu o probabilitate de  $10/23$  în contextul nul [7].

Prin urmare, probabilitatea amestecată de a vedea un "a" ca următorul simbol este:  $2/5 + 2/5 \times (5/10 + 2/10 \times (10/23))$ . Probabilitatea ca următorul simbol să fie un "c". În acest caz, contextele de ordinul 2 și de ordinul 1 nu dau un "c". Prin urmare, trecem la modelul de ordin 0 și prezicem un "c" cu o probabilitate de  $3/23$ . În acest caz, probabilitatea totală de a vedea un "c" ar fi:  $0/5 + 2/5 \times (0/10 + 2/10 \times (3/23)) = 2/5 \times 2/10 \times 3/23$  [7].

Această metodă de atribuire a probabilităților prezintă următoarele avantaje:

1) Rezolvă problema frecvenței zero. În exemplu de mai sus, dacă ar fi fost ales doar cel mai lung context pentru a lua o decizie privind probabilitatea, acesta ar fi returnat o probabilitate zero pentru simbolul "c", în timp ce modelele de ordin inferior arată că această probabilitate este într-adevăr diferită de zero.

2) Această strategie de combinare atribuie o pondere mai mare modelelor de ordin superior în calcularea probabilității dacă simbolul luat în considerare se găsește în acel context, în timp ce modelele de ordin inferior sunt suprimate din cauza probabilității de scăpare a contextului nul [5].



### **1.3. Securitatea unei Case Inteligente utilizând recunoașterea facială**

În perioada actuală, securitatea locuinței reprezintă o preocupare destul de importantă pentru toată lumea. Metodele tradiționale de securizare a casei noastre sunt foarte ușor de spart și pot duce la furt. Iar dacă ne dorim să ne protejăm casa așa cum este indicat, trebuie să instalăm un sistem de securitate costisitor. Această problemă reprezintă un impediment, dar se poate depăși într-un mod rapid, folosind o soluție bazată pe IoT, prin care putem configura un sistem inteligent de securitate pentru casă. Astfel, în acest subcapitol, ne propunem să discutăm despre un sistem cu ajutorul recunoașterii faciale. De asemenea, el ne oferă și facilitatea de a ne monitoriza casa de la distanță, dar și de a lua măsuri adecvate în cazul în care ceva nu merge bine.

Camera Pi va fi atașată la Raspberry pi însoțită de infraroșu pasiv și de alți senzori. Ea captează o imagine a persoanei din fața ușii, apoi recunoașterea feței în timp real se face cu ajutorul modelului binar local (LBP). Dacă imaginea persoanei se potrivește cu unul dintre membrii casei sau cu o persoană identificată, atunci ușa se va debloca, altfel nu se va debloca. E-mailul care conține imaginea persoanei necunoscute va fi trimis proprietarului casei la adresa sa de Gmail. Sistemul pe care îl menționăm îl informează în timp real pe proprietar despre persoana necunoscută de la ușa locuinței. Aceste informații îl vor ajuta pe utilizator să ia măsurile necesare.

#### **1.3.1 Generalități**

Așa cum am menționat anterior, un sistem de securitate la domiciliu este necesar pentru a ne proteja bunurile și obiectele de valoare de orice fel de jaf sau furt. Spre deosebire de România, situația în alte țări este un pic diferită. Și mult mai gravă. De exemplu, în SUA, în fiecare 13 secunde are loc un furt din locuință, 4 spargerii pe minut, 240 pe oră și aproape 6.000 pe zi.

Sistemul tradițional de securitate pentru locuințe este ușor de spart și destul de învechit. Acest lucru duce la rândul său la jafuri și necesită, de asemenea, instalarea sistemului de securitate costisitor. Sistemul despre care vorbim în acest moment oferă o soluție eficientă din punct de vedere al costurilor și al consumului de energie pentru securitatea locuinței prin utilizarea IoT și a recunoașterii feței. Sistemul de recunoaștere a feței bazat pe LBP ne ajută să detectăm persoana de la ușă cu mai multă acuratețe [după cum se observă în Figura 1.6. și Figura 1.7.]. Internetul lucrurilor, sau IoT, este un sistem de dispozitive de calcul, mașini mecanice și digitale, obiecte, animale sau persoane interconectate, cărora li se oferă identifikatori unici (UID) și capacitatea de

a transfera date printr-o rețea fără a necesita interacțiune între oameni sau între oameni și computere. Transmiterea de date prin intermediul unei rețele poate face parte din sistemul IoT prin încorporarea acestora cu hardware electronic, cum ar fi senzori, software și echipamente de rețea.

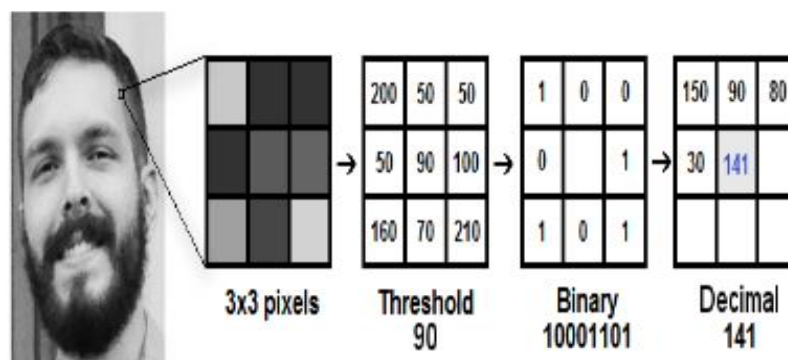


Figura1.6: Recunoaștere facială bazat pe algoritmul LBP

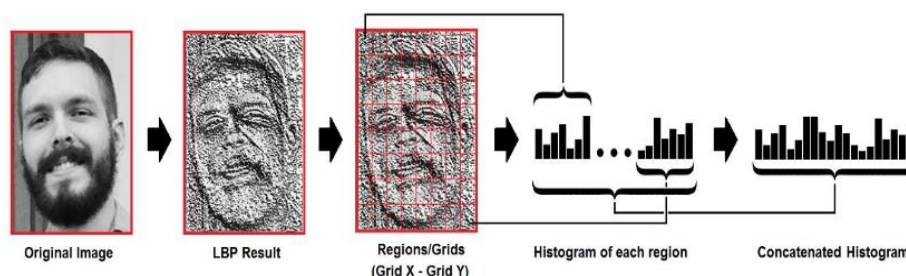


Figura1.7: Recunoaștere facială bazat pe algoritmul LBP

Cele mai recente tehnologii disponibile pentru securitate includ tehnologii de carduri RFID, biometrice, sisteme de deblocare bazate pe OTP și multe altele. Fiecare dintre acestea este eficientă în anumite domenii, dar nu oferă un sistem de securitate complet. Mulți dintre cercetători au propus un sistem de securitate la domiciliu, dar foarte puțini au folosit recunoașterea feței. Recunoașterea facială are un potențial bun pentru a oferi un sistem de securitate puternic pentru locuințe. Mai jos vom prezenta, pe scurt, care sunt avantajele unui astfel de sistem de securitate, dar și cum funcționează într-o casă inteligentă.

### 1.3.2 Mod de aplicare a recunoașterii faciale într-o casă inteligentă

Odată cu apariția recunoașterii faciale, este mai ușor decât înainte să verificăm identitatea unei persoane. Recunoașterea facială poate fi aplicată în multe domenii, unul dintre acestea fiind sistemele inteligente de securitate pentru locuințe. Cu fețele cunoscute capturate într-o bază de

date, camerele de securitate ne vor identifica prietenii și membrii familiei și nu vor da alarma. În caz contrar, vom primi o alertă instantanee, astfel încât să știm că o persoană neidentificată a pătruns pe proprietatea noastră. Dacă până acum suna un pic futurist, acum, în zilele noastre, este ceva cât se poate de normal. Recunoașterea facială este o tehnologie biometrică, care merge dincolo de simpla detectare a fețelor umane într-o imagine sau într-o înregistrare video. Ea merge puțin mai departe pentru a determina a cui este fața.

Un sistem de recunoaștere facială funcționează luând o imagine a unei fețe și precizând dacă fața se potrivește cu o altă față stocată într-o bază de date (sau dacă o față dintr-o imagine se potrivește cu o față din alta). Prin urmare, tehnologia este concepută pentru a compara și a prezice cu exactitate potențialele potriviri ale feței capturate, indiferent de expresia facială, vârstă sau părul facial. În mod tradițional, un furnizor de securitate trimitea un tehnician pentru a instala un sistem cu fir în casa noastră și să ne înscrie într-un serviciu de monitorizare profesionist. Povestea s-a schimbat. Odată cu apariția tehnologiei inteligente, putem chiar să configurăm singur sistemul inteligent. În plus, servim drept monitor profesionist, primind actualizări și notificări în timp real pe un dispozitiv inteligent.

Recunoașterea facială se definește ca fiind camera inteligentă pe care o folosim în casă, ce poate asocia o față cu o identitate. Prin urmare, pentru ca recunoașterea facială să funcționeze, va trebui să îi spunem sistemului ce față aparține cărui nume. Asta înseamnă că, pentru a aplica recunoașterea facială în sistemele de securitate pentru casă, va trebui să creăm profiluri pentru prieteni, rude și alte persoane pe care dorim ca sistemul să le identifice.

Astfel, vom putea ști când aceștia se află la ușa noastră sau când intră în casă. Interesant este că, cu ajutorul recunoașterii faciale, putem seta și criterii de alarmă selectivă. De exemplu, putem seta alerte care să ne anunțe atunci când cineva a cărui față nu este recunoscută de cameră încearcă să intre în casă.

Tehnologia inteligentă a continuat să se dezvolte cu pași repezi. Companiile încep acum să ofere încuietori inteligente pentru uși cu recunoaștere facială. Cu o încuietoare de ușă inteligentă cu recunoaștere facială, ne putem descurca ușile doar cu un zâmbet. Putem, totuși, să blocăm și să deblocăm acea ușă inteligentă folosind alte modalități, inclusiv un cod PIN sau, uneori, chiar o cheie fizică. De asemenea, putem configura încuietoarea inteligentă a casei noastre în așa fel încât, dacă o persoană aflată pe lista neagră încearcă să deschidă încuietoarea inteligentă a ușii, sistemul ne va trimite o alertă de urgență.

### **1.3.3. Avantajele sistemului de securitate într-o casă inteligentă**

#### **Rate de precizie ridicate**

Majoritatea software-urilor de recunoaștere facială utilizate în încuietorile inteligente pentru locuințe pot determina cu precizie dacă persoana care încearcă să intre în locuință se potrivește cu baza de date a persoanelor care au fost autorizate să intre. Acestea fiind spuse, unele programe software sunt mai precise decât altele, în special atunci când vine vorba de detectarea fețelor din diferite unghiuri și de recunoașterea fețelor de diferite naționalități.

Pentru cei care sunt preocupați de acuratețe, trebuie remarcat faptul că unele programe software vin cu opțiunea de a seta praguri de încredere personalizate, ceea ce poate reduce foarte mult posibilitatea ca sistemul să returneze un fals pozitiv. Există, de asemenea, opțiunea de a utiliza autentificarea cu doi factori, cum ar fi recunoașterea facială combinată cu un cod PIN sau un breloc de chei.

#### **Automatizare**

Odată ce am configurat sistemul de securitate inteligent pentru casă și am acordat acces persoanelor pe care dorim să le permitem să intre în casă, sistemul le va permite automat accesul atunci când detectează o potrivire. Nu este nevoie să răspundem la sonerie sau să răspundem la o cerere de intrare.

#### **Integrare inteligentă**

Instrumentele de recunoaștere facială pot fi integrate cu ușurință în sistemele existente folosind un SDK sau API. Dacă decizia este una dificilă, atunci putem menționa și următorul aspect: verificarea biometrică se bazează pe trăsături comportamentale și fiziologice unice, printre care se numără amprente digitale, dinamica apăsărilor de taste, trăsăturile faciale, mersul și amprenta vocală. La pol opus, recunoașterea facială pare a fi favorită datorită acurateței, ușurinței de utilizare și a faptului că nu necesită contact.

În concluzie, tehnologia de recunoaștere facială a fost implementată pentru a rămâne pe viitor, urmând să se îmbunătățească cu timpul. O astfel de tehnologie inteligentă a dat curs dezvoltării erei digitale și a sporit opțiunile privind îmbunătățirea locuinței ideale.

## Capitolul 2. Python cu librării de Inteligență Artificială și Viziune Computerizată

Dacă în primul capitol am vorbit generalități despre ce înseamnă o casă inteligentă, cum funcționează sistemul de securitate, precum și ce algoritm a fost utilizat, în cel curent vom vorbi despre limbajul de programare Python (folosit pentru implementarea unei astfel de case). Cu alte cuvinte, vom vedea care sunt librăriile reprezentative pentru AI, librăriile „OpenCV”, dar și cum se va „descurca” limbajul în sistemul IoT. Dar înainte să intrăm în detalii, este necesar să aruncăm o privire asupra ceea ce înseamnă limbajul de programare Python. Așadar, începem analiza noastră din capitolul doi cu o întrebare: Ce este Python?

Python este un limbaj de programare popular. Acesta a fost creat de către dezvoltatorul software Guido van Rossum și lansat în 1991. În momentul în care a dezbătut subiectul originii limbajului, Guido a făcut următoarea afirmație: „Cu mai bine de șase ani în urmă, în decembrie 1989, căutam un proiect de programare (un fel de "hobby"), care să mă țină ocupat în săptămâna din preajma Crăciunului. Biroul meu (un laborator de cercetare guvernamental din Amsterdam) urma să fie închis, dar aveam un computer acasă și nu prea aveam altceva pe cap. M-am decis să scriu un interpret pentru noul limbaj de scripting la care mă gândisem în ultima vreme: un descendent al ABC care să fie pe placul hackerilor Unix/C. Am ales Python ca titlu de lucru pentru proiect, fiind într-o dispoziție ușor ireverențioasă (și un mare fan al Monty Python's Flying Circus)”.

Dacă analizăm declarația domnului van Rossum, putem trage următoarea concluzie: Python este un limbaj de scripting. Și totuși, sunt suficiente informațiile astea pentru a continua cu această idee? Cercetările ulterioare au demonstrat că limbajul de programare Python este considerat a fi de acest tip, dar, în același timp, este și un limbaj dinamic, coerent și bine pus la punct, neexistând niciun motiv pentru a nu-l folosi pentru o gamă largă de aplicații. Totodată, problematica acestei categorisiri vine și din partea companiilor de marketing. Pe rețelele de socializare și în mediul online, se promovează noțiunea că Python este un limbaj de scripting ușor de învățat. Drept urmare, toate limbajele asemănătoare sunt comprehensibile fiecărui utilizator. Iar programatorii din zilele noastre au demonstrat prin munca lor continuă că situația nu este tocmai atât de roz. De aceea, trebuie să face noi propria analiză pentru a vedea cum se prezintă situația sau situațiile în cauză.

După ce am clarificat noțiunile introductive despre apariția limbajului și tipologia lui, a venit momentul să aflăm și motivul pentru care programatorii noștri sunt adepții acestuia limbaj. Afirmția de mai sus a fost făcută în urma unor sondaje de opinie în mediul online, realizate de cercetători, care au demonstrat că Python se clasează în primele zece limbaje de programare iubite de IT-iști (Figura2.1). Deci, continuăm analiza noastră și punem următoare întrebare: De ce Python?

- funcționează pe diferite platforme (Windows, Mac, Linux, Raspberry Pi etc.);
- are o sintaxă simplă, similară cu cea a limbii engleze;
- are o sintaxă care permite dezvoltatorilor să scrie programe cu mai puține linii decât alte limbaje de programare;
- este portabil. Funcționează aproape oriunde (servere de înaltă performanță și stații de lucru, până la Windows CE);
- rulează pe un sistem de interpretare, ceea ce înseamnă că codul poate fi executat imediat ce este scris. Acest lucru înseamnă că realizarea prototipurilor poate fi foarte rapidă;
- poate fi tratat în mod procedural, orientat pe obiecte sau funcțional.



Figura 2.1: Sondaj: 10 limbaje de programare iubite de IT-iști (sursa: geektech.me)

Pe lângă aceste calități, limbajul despre care vorbim mai are și unele reprezentative pentru el. De exemplu, are o sintaxă clară, fără elemente suplimentare, și tipuri de date de înalt nivel. Sau, folosește spații albe pentru a delimita blocurile. Prin urmare, toate aceste trăsături pe care le are Python l-au transformat, în momentul actual, într-un limbaj de actualitate, folosit destul de des de programatorii din România (și nu numai) și căutat de companiile de software.

Mergem mai departe cu studiul nostru și aducem în discuție „capitolul”: Ce poate face Python? Manualele de IT sunt arhipline de informații mai mult sau mai puțin relevante pentru subiectul nostru, așa că vom enumera câteva dintre cele mai importante obiective ale acestui limbaj de programare. Și avem așa:

- poate fi utilizat pe un server pentru a crea aplicații web;
- poate fi utilizat alături de software pentru a crea fluxuri de lucru;
- se poate conecta la sisteme de baze de date. De asemenea, poate citi și modifica fișiere;
- poate fi utilizat pentru a gestiona date mari și pentru a efectua calcule matematice complexe;
- poate fi utilizat pentru prototipuri rapide sau pentru dezvoltarea de software gata de producție.

După cum se poate observa, este un limbaj care îndeplinește sarcini destul de complexe și des folosite în era noastră tehnologică, aflată în plin proces de dezvoltare. Pentru că ar mai fi multe de zis despre acest limbaj, ne vom limita la ceea ce ne interesează pe noi la acest studiu și vom vorbi în următoarele rânduri despre ultimul element definitoriu al limbajului, și anume: În ce se folosește?

Așa cum am văzut în cele prezentate mai sus, Python, chiar dacă are o sintaxă simplă și este ușor de învățat, în comparație cu alte limbaje de programare, este un limbaj complex. Deci, se poate spune că este folosit pentru:

- prototiparea rapidă;
- scripting web;
- programare ad-hoc, de unică folosință;
- conducerea aplicațiilor științifice;

- limbaj de extensie;
- procesare XML;
- aplicații de baze de date;
- aplicații GUI.

În concluzie, pe baza tuturor informațiilor menționate anterior, Python este un limbaj de programare aparte, folosit într-o sumedenie de aplicații și, nu în ultimul rând, în implementarea caselor inteligente, o soluție de viitor pentru toți oamenii. Despre cum se construiește o astfel de casă în limbajul de programare Python, vom discuta în următoarele trei subcapitole.

## **2.1. Internet of Things (IoT) cu Python**

Înainte de a intra în miezul acestui subiect, este necesar să clarificăm câteva noțiuni legate de acest termen: „Internet of Things”. Ce este el, cu ce se ocupă mai exact și care este legătura lui cu limbajul Python?

Termenul de "IoT" datează din 1999, fiind inventat de Kevin Ashton. De atunci, importanța și amploarea sa a explodat, de fapt, unul dintre principalii indicatori este dimensiunea pieței sale de 151 de miliarde de dolari în 2018, cu o creștere constantă de la an la an. Pe baza previziunilor marketerului pentru 2022, piața IoT ar putea atinge pragul de 561 de miliarde de dolari.

Pe vremuri, „Internet of Things” era explicat cu exemple ca acesta: "Putem folosi telefonul pentru a aprinde și stinge un bec din cameră". În zilele noastre, aproape nimeni nu ar fi surprins de un contor de electricitate inteligent care transmite citiri ale consumului de electricitate, încarcă aceste informații în cloud și generează facturi lunare trimise direct pe e-mail.

În toate industriile, IoT este din ce în ce mai mult utilizat pentru a eficientiza procesele și a le face mai eficiente. De exemplu, liniile de producție din industria prelucrătoare și agricultura sunt exemple excelente de industrii diferite care profită de numeroasele beneficii ale IoT. În cazul specific al agriculturii, IoT ajută la coordonarea secerătorilor cu camioanele care au elevatoare pentru a manipula eficient cerealele.



Deseori, prototipurile sau sistemele Internet of Things (IoT) din viața reală trebuie dezvoltate rapid și eficient. Atunci când se întâmplă acest lucru, două sarcini prind imediat viață: programarea dispozitivelor IoT și organizarea unui backend care interacționează cu aceste dispozitive. În ambele sarcini, putem folosi Python ca limbaj de dezvoltare. Sau, putem utiliza o versiune complet funcțională și practică a MicroPython pentru a lucra pe dispozitive cu resurse de calcul reduse și, în consecință, la un cost foarte scăzut. Să aruncăm o privire asupra modului în care puteți utiliza Python pentru a programa dispozitivele IoT și pentru a crea un backend pentru ca acestea să funcționeze.

Pentru mulți dezvoltatori, Python este limbajul preferat pe piață. Este ușor de învățat, are o sintaxă curată și este sprijinit de o comunitate mare online. În IoT, Python este o alegere excelentă pentru partea de dezvoltare backend, precum și pentru dezvoltarea software a dispozitivelor. În plus, Python este disponibil pentru a rula pe dispozitive Linux și putem utiliza MicroPython pentru microcontrolere.

Unele dintre numeroasele avantaje ale lucrului cu Python pentru dispozitivele IoT sunt viteza cu care vom dezvolta codul și numărul mare de biblioteci pentru toate tipurile de platforme. Python este un aliat excelent pentru a dezvolta prototipuri de dispozitive. Chiar dacă rescriem o parte din cod în timpul producției în C, C++ sau Java pentru a îmbunătăți performanța, în general, sistemul va funcționa perfect în Python.

### **2.1.1. Cele mai bune soluții pentru IoT în Python**

#### *Python pe Raspberry Pi*

Primul lucru pe care ar trebui să îl facem în legătură cu rularea Python pe un dispozitiv IoT este să apelăm la Raspberry Pi. Python va fi preinstalat în sistemul de operare și nu ne va rămâne decât să ne scriem propriul script.

În acest caz, putem controla porturile I/O de pe bara de expansiune a Raspberry Pi. Din fericire, placa suportă comunicații wireless (Bluetooth și WiFi), Ethernet și putem, de asemenea, să conectăm un monitor la ieșirile HDMI. Controlerele, disponibile într-o gamă largă de puteri de calcul și bugete, pot fi alese pentru sistemul nostru IoT - de la rapidul Raspberry Pi 4 Model B de 8GB până la cel mai mic Raspberry Pi Zero, toate suportând Python. Dacă este necesar, putem instala versiunea anterioară de Python 2.7 pentru compatibilitate în trecut.

Pentru a controla porturile I/O, vom scrie cu ușurință câteva linii în Python folosind biblioteca GPIO Zero, așa cum vedem în exemplul din Figurade mai jos (Figura2.2.):

```
1 from gpiozero import Button
2 from time import sleep
3
4 button = Button(2)
5
6 while True:
7     if button.is_pressed:
8         print("Button Press")
9     else:
10        print("Button Release")
11        sleep(1)
```

Figura 2.2: Cod în Python, folosind biblioteca GPIO zero

Acest exemplu arată cât de ușor este să primim și să procesăm semnale prin apăsarea butonului de pe cel de-al doilea pin în momentul apăsării și în momentul eliberării. Avantajele utilizării acestei abordări sunt disponibilitatea unei game largi de instrumente de dezvoltare, biblioteci și comunicații pentru cele mai complexe dispozitive bazate pe Raspberry Pi, inclusiv procesarea video de la camere.

### Python pe PyBoard

Următoarea soluție excelentă pentru Python în dispozitivele IoT este PyBoard cu un microcontroler STM32F405RG. PyBoard este o placă de dezvoltare electronică compactă și puternică, care rulează MicroPython. Se conectează la PC prin USB, oferindu-ne o unitate flash USB pentru a ne salva scripturile Python și un prompt Python serial (un REPL) pentru programare instantanee. Aceasta funcționează cu Windows, Mac și Linux. PyBoard rulează MicroPython, o implementare ușoară a interpretorului CPython standard. MicroPython este o rescriere completă a limbajului de programare Python (versiunea 3.4) astfel încât să se potrivească și să ruleze pe un microcontroler.

Acesta include multe optimizări pentru eficiență și folosește foarte puțină memorie RAM. MicroPython rulează bare-metal pe PyBoard, oferindu-ne, în esență, un sistem de operare Python. Modulul pyb încorporat conține funcții și clase pentru a controla perifericele disponibile pe placă, cum ar fi UART, I2C, SPI, ADC și DAC. Dimensiunile plăcii sunt impresionante, ocupând aproximativ două sferturi, 33 mm x 43 mm, și cântărind doar 6 grame.

## 2.1.2. IoT Backend cu Python

### Protocolul MQTT cu Python

Cea mai populară metodă de conectare pentru dispozitivele IoT este MQTT, un protocol care este implementat eficient cu Python. Protocolul MQTT este un protocol de conectivitate machine-to-machine (M2M)/Internet of Things (Internetul lucrurilor) care este conceput ca un transport de mesagerie publish/subscribe extrem de ușor. Este util pentru conexiuni cu locații la distanță, unde este necesară o amprentă de cod mică și/sau unde lățimea de bandă a rețelei este foarte redusă. Biblioteca client Eclipse Paho MQTT Python implementează versiunile 5.0, 3.1.1 și 3.1 ale protocolului MQTT. Codul bibliotecii Paho oferă o clasă client care permite aplicațiilor să se conecteze la un broker MQTT pentru a publica mesaje, a se abona la subiecte și a primi mesaje publicate. De asemenea, oferă câteva funcții de ajutor pentru a simplifica publicarea de mesaje unice către serverele MQTT. Interesant este faptul că biblioteca acceptă Python 2.7.9+ sau 3.5+. Integrarea imaginii cu versiunile mai vechi 2.7 ale Python este mai ușoară.

### Backend IoT pe Flask în Python

Pentru un instrument rapid și fără complicații pentru a scrie backendul pentru sistemele IoT și pentru a configura ușurință informațiile de intrare/ieșire de pe server, microcadru Flask este aici pentru a ne salva și este plin de funcționalități.

Pentru a începe, decidem care sunt solicitările pe care trebuie să le servim de la dispozitivele dvs. IoT, configurăm microframeworkul Flask și scriem câteva linii de cod. Metoda GET va returna acum informații la solicitarea din partea clientului. În multe cazuri, cel mai bine este să ne orientăm către protocolul RESTful atunci când lucrăm cu dispozitivele IoT. Acest lucru simplifică schimbul între componentele sistemului și ne permite să extindem sistemul de schimb de informații în viitor.

Dezavantajul utilizării acestei abordări este potențiala lipsă de inițiere a transferului de date de la server la dispozitiv. Adică, IoT trebuie să tragă independent și periodic de la server. Dar există soluții și pentru a aborda acest risc. Putem utiliza socket-uri web sau o bibliotecă Python pentru Pushsafer. Cu PushSafer, putem trimite și primi cu ușurință și în siguranță notificări push în timp real pe dispozitivele iOS, Android și Windows (mobile și desktop), precum și pe browsere precum Chrome, Firefox, Opera etc.

### Microsoft Azure IoT backend în Python

Microsoft a lansat o nouă extensie IoT open-source care extinde capacitățile Azure CLI 2.0. Azure CLI 2.0 include comenzi pentru a interacționa cu Azure Resource Manager și cu punctele finale de gestionare. De exemplu, putem utiliza Azure CLI 2.0 pentru a crea un Azure VM sau un hub IoT. Extensia CLI permite unui serviciu Azure să completeze Azure CLI, oferind utilizatorilor acces la capacități suplimentare specifice serviciului.

Extensia IoT oferă dezvoltatorilor acces prin linia de comandă la capacitățile IoT Hub, IoT Edge și IoT Hub Device Provisioning Service. Azure CLI 2.0 permite gestionarea imediată a resurselor Azure IoT Hub, a instanțelor serviciului de furnizare a dispozitivelor și a hub-urilor asociate. Noua extensie IoT îmbogățește Azure CLI 2.0 cu funcții precum gestionarea dispozitivelor și toate capacitățile IoT Edge:

- azure CLI 2.0 capacități IOT - planul de control;
- gestionarea instanțelor hub-urilor IoT, a grupurilor de consumatori și a lucrărilor;
- gestionarea instanțelor serviciului de furnizare a dispozitivelor, politicile de acces, Linked-hub și certificatele;
- caracteristici noi pentru extensii - planul de date;
- gestionarea identităților dispozitivelor și modulelor de margine, precum și definițiile gemene respective;
- interogarea IoT Hub pentru informații precum gemeni de dispozitive și module, sarcini și setarea mesageriei;
- invocarea metodelor pentru dispozitive și module;
- generarea de token-uri SAS și preluarea șirurilor de conexiuni;
- mesagerie de la cloud la dispozitiv și de la dispozitiv la cloud;
- încărcarea fișierelor de dispozitive;
- simularea dispozitivelor pentru testare.

Drept concluzie, putem face, următoarea afirmație: nu Python este cel care face ca lucrul cu dispozitivele IoT și cu backend-ul să fie atât de convenabil. Există mai mulți factori care intră în joc, care lucrează împreună pentru a îmbunătăți ușurința de utilizare a Python pentru IoT:

- prag scăzut pentru a intra în construcțiile de bază ale limbajului de programare;
- număr uriaș de biblioteci pentru o varietate de sarcini;
- un element de bază pentru a rula Python pe partea clientului;
- posibilitatea de a scrie un backend în diverse cadre Python;
- integrare ușoară a backend-ului cu partea de client a IoT, deoarece ambele sunt scrise în Python;
- cerințe de sistem reduse, în special pentru Micropython;
- suport pentru cloud;
- număr mare de specialiști instruiți.

Python nu va putea să facă totul pentru noi, dar ne va ajuta în mod semnificativ să creăm primul prototip, iar în următoarele versiuni ale sistemului IoT, va îndeplini, de asemenea, funcții atribuite împreună cu alte limbaje de programare și biblioteci. Proiectele privind internetul obiectelor vor face din lumea reală un loc mai bun, iar viața oamenilor va fi într-un progres ascendent continuu.

## **2.2. Librării pentru AI în Python**

Învățarea automată/ „Machine Learning” (ML) și inteligența artificială (AI) se răspândesc în diverse industrii, iar majoritatea întreprinderilor au început să investească activ în aceste tehnologii. Odată cu extinderea volumului, precum și a complexității datelor, ML și AI sunt recomandate pe scară largă pentru analiza și prelucrarea acestora. AI oferă informații mai precise și predicții mai precise pentru a îmbunătăți eficiența afacerii, a crește productivitatea și a reduce costurile de producție.

Proiectele AI și ML diferă de proiectele software convenționale. Acesta variază în funcție de stiva tehnologică, de competențele pentru proiectele bazate pe ML și de cererea de cercetare aprofundată. Pentru a construi conturul ML și AI, trebuie să alegeți un limbaj de programare, care ar trebui să fie flexibil, stabil și să includă biblioteci și cadre predefinite. Python este unul dintre aceste limbaje, în care puteți vedea multe proiecte Python de învățare automată și inteligență artificială care se dezvoltă astăzi. Vom enumera mai jos și vom detalia (pe fiecare în parte) care sunt primele opt cele mai bune biblioteci Python care ar putea fi utilizate pentru învățarea automată. Dar înainte să vorbim despre ele, trebuie să aflăm de ce limbajul de programare menționat mai sus este utilizat în ML și AI.

Cu alte cuvinte, Python sprijină dezvoltatorii pe tot parcursul dezvoltării de software pentru a fi atât productivi, cât și încrezători în produsul pe care îl construiesc. El oferă o mulțime de beneficii pentru construirea de proiecte AI și ML. În Figura 2.3. se pot observa care sunt aceste beneficii. Prin urmare, aceste caracteristici adaugă valoare la popularitatea generală a limbajului de programare.

Colecțiile extinse de biblioteci Python pentru învățare automată simplifică sarcina de dezvoltare și reduc timpul de dezvoltare. Sintaxa sa simplă, precum și lizibilitatea sprijină testarea rapidă a proceselor complexe și face ca limbajul să fie ușor de înțeles pentru neprogramatori. PHP este considerat un concurent al lui Python în ceea ce privește paginile web și dezvoltarea de aplicații. Dar în ceea ce privește AI și ML, vom avea nevoie de un PHP specific, cu legătură la bibliotecile ML.

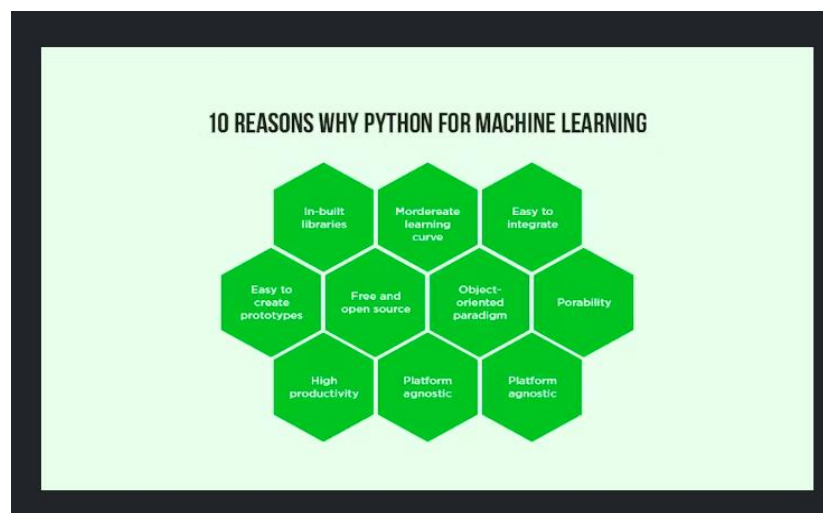


Figura 2.3: Beneficii ale limbajului Python pentru ML (sursă: hackernoon.com)

### 2.2.1 Cele mai bune biblioteci Python pentru Machine Learning și AI

Implementarea algoritmilor ML și AI necesită un mediu bine structurat și bine testat pentru a le permite dezvoltatorilor să vină cu soluții de codare de cea mai bună calitate. Pentru a reduce timpul de dezvoltare, există nenumărate biblioteci Python pentru învățare automată. Biblioteca sau cadrul Python este un program prescris care este gata să fie utilizat pentru sarcini de codare comune. Mai jos vom enunța cele mai bune biblioteci Python de învățare automată.

#### Tensor Flow Python

TensorFlow este o bibliotecă de învățare automată python end-to-end pentru efectuarea de calcule numerice de vârf. TensorFlow poate gestiona rețele neuronale profunde pentru recunoașterea imaginilor, clasificarea cifrelor scrise de mână, rețele neuronale recurente, NLP (Natural Language Processing), înglobarea cuvintelor și PDE (Partial Differential Equation). TensorFlow Python asigură un suport excelent pentru arhitectură, pentru a permite implementarea ușoară a calculelor pe o gamă largă de platforme, inclusiv pe desktopuri, servere și mobile.

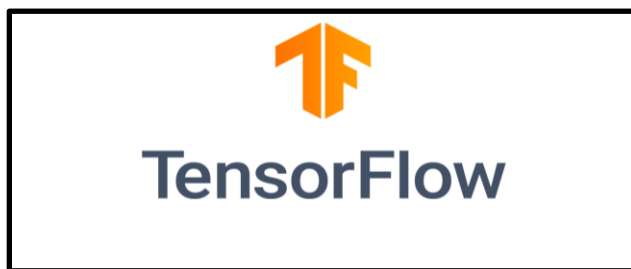


Figura 2.4: Tensor Flow - Logo

Abstractizarea este beneficiul major al TensorFlow Python față de proiectele de învățare automată și de inteligență artificială. Această caracteristică le permite dezvoltatorilor să se concentreze pe logica cuprinzătoare a aplicației în loc să se ocupe de detaliile banale ale implementării algoritmilor. Cu această bibliotecă, dezvoltatorii python pot acum să profite fără efort de AI și ML pentru a crea aplicații unice și receptive, care răspund la intrările utilizatorului, cum ar fi expresia facială sau vocală.

#### Keras Python

Keras este o bibliotecă Python open-source de top, scrisă pentru construirea de rețele neuronale și proiecte de învățare automată. Poate rula pe Deeplearning4j, MXNet, Microsoft Cognitive Toolkit (CNTK), Theano sau TensorFlow. Oferă aproape toate modulele de sine

stătătoare, inclusiv optimizatoare, straturi neuronale, funcții de activare, scheme de inițializare, funcții de cost și scheme de regularizare. Permite adăugarea de noi module cu ușurință, la fel ca și adăugarea de noi funcții și clase. Deoarece modelul este deja definit în cod, nu este nevoie de fișiere de conFiguraurare a modelului separate.



Figura 2.5: Keras Python - Logo

Keras simplifică proiectarea și dezvoltarea unei rețele neuronale pentru începătorii în domeniul învățării automate. Keras Python se ocupă, de asemenea, de rețele neuronale de convoluție. Acesta include algoritmi pentru normalizare, optimizator și straturi de activare. În loc de a fi o bibliotecă de învățare automată Python end-to-end, Keras funcționează ca o interfață extensibilă și ușor de utilizat, care îmbunătățește modularitatea și expresivitatea totală.

### Theano Python

De la sosirea sa în 2007, Theano a captat dezvoltatorii Python și cercetătorii de ML și AI. La bază, este o bibliotecă de calcul științific bine cunoscută care vă permite să definiți, să optimizați, precum și să evaluați expresii matematice, care se ocupă de matrici multidimensionale.

Fundamentul mai multor aplicații ML și AI este calculul repetitiv al unei expresii matematice complicate. Theano vă permite să efectuați calculele cu utilizare intensivă a datelor de până la o sută de ori mai rapid decât atunci când se execută doar pe procesorul dumneavoastră.

În plus, este bine optimizat pentru GPU, care oferă o diferențiere simbolică eficientă și include capacități extinse de testare a codului.



Figura 2.6: Theano Python - Logo



Când vine vorba de performanță, Theano este o bibliotecă Python de învățare automată excelentă, deoarece include capacitatea de a face față calculelor în rețele neuronale mari. Scopul său este de a crește timpul de dezvoltare și timpul de execuție al aplicațiilor ML, în special în cazul algoritmilor de învățare profundă. Un singur dezavantaj al Theano în fața TensorFlow este că sintaxa sa este destul de dificilă pentru începători.

### Scikit-learn Python

Scikit-learn este o altă bibliotecă proeminentă de învățare automată Python open-source cu o gamă largă de algoritmi de clusterizare, regresie și clasificare. DBSCAN, gradient boosting, păduri aleatoare, mașini vectoriale și k-means sunt câteva exemple. Poate interoperă cu bibliotecile numerice și științifice Python, cum ar fi NumPy și SciPy.



Figura 2.7: Scikit-learn Python – Logo

Este o bibliotecă de inteligență artificială utilizabilă în comerț. Această bibliotecă Python suportă atât ML supravegheat, cât și nesupravegheat. Iată o listă a beneficiilor principale ale Scikit-learn Python care îl face să fie una dintre cele mai preferabile biblioteci Python pentru învățarea automată:

- reducerea dimensionalității;
- tăiere și inducție a arborelui de decizie;
- învățarea limitelor de decizie;
- analiza și selectarea caracteristicilor;
- detectarea și respingerea valorilor aberante;
- modelare avansată a probabilităților;
- clasificare și grupare nesupravegheată.

### PyTorch Python

PyTorch este o bibliotecă de învățare automată pregătită pentru producție, cu exemple, aplicații și cazuri de utilizare excelente, susținute de o comunitate puternică. Această bibliotecă absoarbe o accelerare puternică a GPU și vă permite să o aplicați din aplicații precum NLP.



Figura 2.8: PyTorch Python - Logo

Deoarece suportă calculele GPU și CPU, vă oferă optimizarea performanței și o instruire distribuită scalabilă atât în cercetare, cât și în producție. Rețelele neuronale profunde și calculul Tensor cu accelerare GPU sunt cele două caracteristici de vârf ale PyTorch.

Acesta include un compilator de învățare mecanică numit Glow, care sporește performanța cadrelor de învățare profundă.

### NumPy Python

NumPy sau Numerical Python este o algebră liniară dezvoltată în Python. Aici ne putem pune și următoarea întrebare: De ce un număr mare de dezvoltatori și experți o preferă celorlalte biblioteci Python pentru învățare automată?

Aproape toate pachetele Python de învățare automată, cum ar fi Matplotlib, SciPy, Scikit-learn etc., se bazează pe această bibliotecă într-o măsură rezonabilă. Ea vine cu funcții pentru a face față operațiilor matematice complexe, cum ar fi algebra liniară, transformarea Fourier, numere aleatoare și funcții care lucrează cu matrici și n-array-uri în Python. Pachetul NumPy Python efectuează, de asemenea, calcule științifice. Este utilizat pe scară largă în manipularea undelor sonore, a imaginilor și a funcțiilor binare.

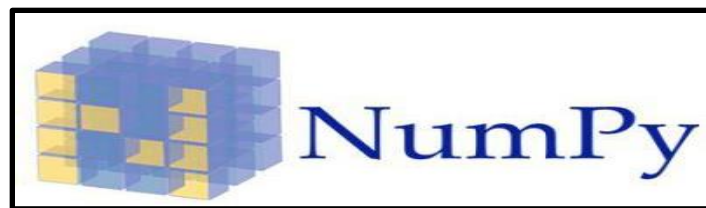


Figura 2.9: NumPy Python - Logo

### Python Pandas

În cadrul proiectelor de învățare automată, o cantitate substanțială de timp este dedicată pregătirii datelor, precum și analizei tendințelor și modelelor de bază. Acesta este momentul în care Python Pandas primește atenția experților în învățare automată. Python Pandas este o bibliotecă open-source care oferă o gamă largă de instrumente pentru manipularea și analiza datelor. Cu această bibliotecă, puteți citi date dintr-o gamă largă de surse, cum ar fi CSV, baze de date SQL, fișiere JSON și Excel.



Figura 2.10: Python Pandas - Logo

Vă permite să gestionați o operațiune complexă cu date cu doar una sau două comenzi. Python Pandas vine cu mai multe metode încorporate pentru combinarea datelor și pentru funcționalitatea de grupare și filtrare a seriilor temporale. În general, Pandas nu se limitează doar la gestionarea sarcinilor legate de date; servește ca cel mai bun punct de plecare pentru a crea instrumente de date mai concentrate și mai puternice.

### Seaborn Python

În cele din urmă, ultima bibliotecă din lista de biblioteci Python pentru învățare automată și inteligență artificială este Seaborn - o bibliotecă de vizualizare de neegalat, bazată pe bazele Matplotlib. Atât povestirea, cât și vizualizarea datelor sunt importante pentru proiectele de învățare automată, deoarece acestea necesită adesea analiza exploratorie a seturilor de date pentru a decide asupra tipului de algoritm de învățare automată care trebuie aplicat. Seaborn oferă o interfață de nivel înalt bazată pe seturi de date pentru a realiza grafice statistice uimitoare.



Figura 2.11: Seaborn Python - Logo

Cu această bibliotecă Python de învățare automată, este simplu să creați anumite tipuri de diagrame, cum ar fi serii de timp, hărți de căldură și diagrame de vioară. Funcționalitățile lui Seaborn depășesc Python Pandas și matplotlib cu caracteristici pentru a efectua estimări statistice în momentul combinării datelor între observații, trasarea și vizualizarea adecvării modelelor statistice pentru a consolida modelele setului de date.

În concluzie, toate aceste biblioteci menționate mai sus sunt extrem de valoroase atunci când decidem să lucrăm la proiecte de machine learning, deoarece economisesc timp și oferă în plus funcții implicite pe care se poate construi. Cu ajutorul acestor biblioteci Python de învățare automată, putem introduce funcții analitice de nivel înalt, chiar și cu cunoștințe minime despre algoritmii de bază pe care îi folosim.

## **2.3. Librăria OpenCV din limbajul Python**

Cele mai recente progrese în domeniul vederii computerizate fac ca viața noastră să fie din ce în ce mai automatizată - algoritmii săi pot administra case, conduce mașini, face diagnostice și asistă oamenii în multe alte moduri. OpenCV este o bibliotecă de viziune computerizată care oferă o selecție bogată de algoritmi utilizați pentru detectarea obiectelor, recunoașterea fețelor, restaurarea imaginilor și multe alte aplicații. Detectarea persoanelor bazată pe viziune computerizată este una dintre caracteristicile bibliotecii, care este deosebit de populară în proiectele de case inteligente. Acest articol vă va vorbi despre OpenCV și despre modul în care îl puteți utiliza pentru a detecta persoane într-un sistem de automatizare a locuinței IoT.

### **2.3.1. O scurtă introducere despre OpenCV**

Viziunea computerizată (CV) este un subansamblu al inteligenței artificiale care se ocupă cu detectarea, procesarea și distingerea obiectelor din imaginile digitale și video. Crearea de aplicații de viziune computerizată vine cu o varietate de soluții și tehnologii, inclusiv biblioteci, cadre și platforme. OpenCV este una dintre aceste soluții. Este un set de biblioteci cu peste 2500 de instrumente care variază de la algoritmi clasici de învățare automată (ML), cum ar fi regresia liniară, mașinile de vectori de suport (SVM) și arborii de decizie, până la învățarea profundă și rețelele neuronale.

O putem considera și o soluție open-source care poate fi utilizată, modificată și distribuită în mod liber sub licența Apache. Biblioteca a fost introdusă inițial de Intel în 1998, iar de atunci corporația a susținut OpenCV și a contribuit la aceasta.

Biblioteca este compatibilă cu o serie de sisteme de operare, inclusiv Windows, Linux, macOS, FreeBSD, Android, iOS, BlackBerry 10, și acceptă software scris în C/C++, Python și Java. Are o puternică capacitate trans platformă și compatibilitate cu alte cadre. De exemplu, putem porta și rula cu ușurință TensorFlow, Caffè, PyTorch și alte modele, fără aproape nicio ajustare.

OpenCV cuprinde o gamă largă de module destinate procesării imaginilor, detectării și urmăririi obiectelor, descrierii caracteristicilor și îndeplinirii multor alte sarcini. Mai jos vom enumera câteva dintre aceste module:

- core. Core functionality
- imgproc. Image Processing
- videoio. Video I/O
- video. Video Analysis
- objdetect. Object Detection
- ml. Machine Learning
- stitching. Images stitching

Pe lângă acestea, biblioteca este echipată cu un modul GPU, care oferă o putere de calcul ridicată pentru a capta videoclipuri, a procesa imagini și a efectua alte operațiuni în timp real. Modulul OpenCV GPU permite dezvoltatorilor să creeze algoritmi avansați pentru aplicații de viziune computerizată de înaltă performanță.

### **2.3.2. Aplicații OpenCV**

OpenCV este o soluție multifuncțională și multifuncțională, care se regăsește într-un spectru larg de aplicații. De exemplu, ajută roboții și mașinile care se conduc singure să navigheze și să evite obstacolele folosind funcții de detectare și urmărire a obiectelor.

Implementarea OpenCV în soluții industriale automatizează sistemele de securitate și procedurile de control al calității. Să luăm în considerare alte câteva aplicații din diverse industrii.

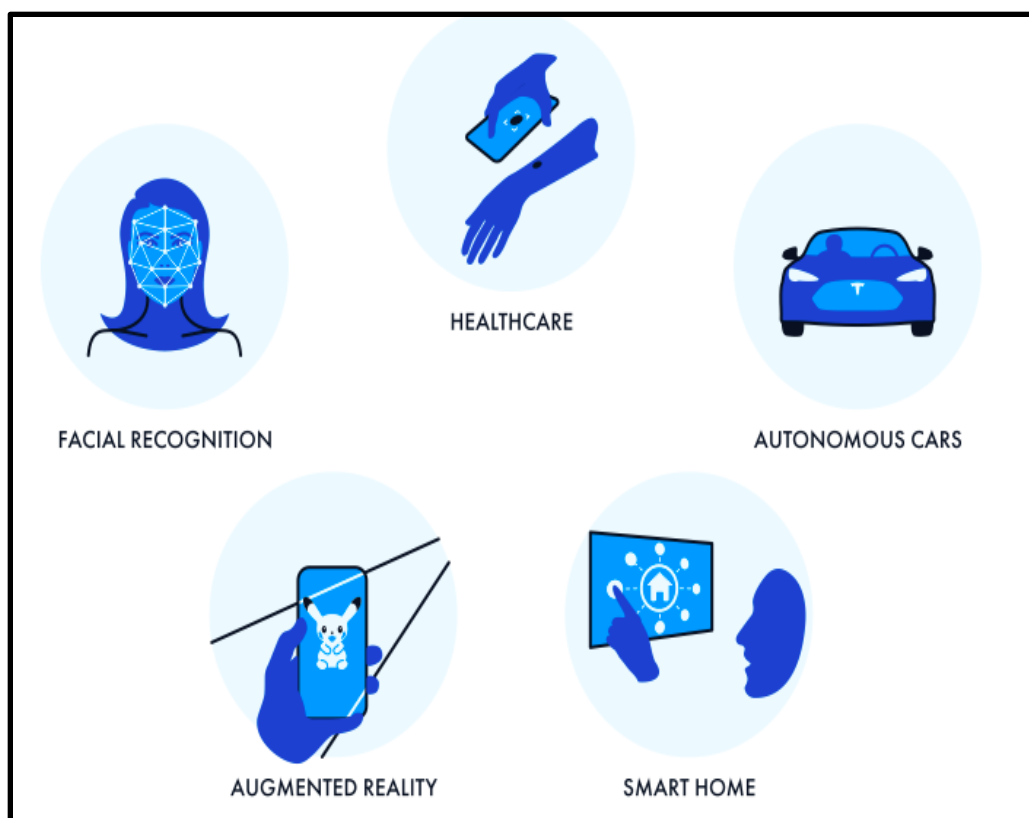


Figura 2.12: Aplicații Open CV

### Sănătate

Algoritmii OpenCV de detecție și clasificare pot identifica o boală sau o potențială amenințare la adresa sănătății prin analizarea imaginilor de organe și țesuturi ale pacienților. Am folosit biblioteca OpenCV pentru a construi o aplicație iOS pentru identificarea melanomului de cancer de piele. Algoritmii proiectului SkinView detectează marginile unei alunițe, îi calculează culoarea, forma și alți parametri și decid dacă alunița este sau nu malignă. După aceea, aplicația trimite rezultatele către un medic. Astfel, putem spune că se va reduce timpul de procesare a datelor la mai puțin de 0,1 secunde și crește precizia diagnosticului la 80%.

### Vehicule autonome

Utilizarea viziunii computerizate împreună cu mai mulți senzori și camere video poate ajuta vehiculele să navigheze fie în aer (vehicule aeriene fără pilot), fie în traficul urban (mașini care se conduc singure). OpenCV oferă o mulțime de algoritmi pentru a detecta, clasifica și urmări persoane și obiecte. Acești algoritmi permit unui vehicul să identifice semnele de circulație, liniile benzilor de circulație, pietonii și alte mașini, ghidându-l astfel în siguranță până la destinație.

### Realitatea augmentată

Viziunea computerizată integrată în software-ul mobil poate alimenta aplicațiile de realitate augmentată (AR), permițând utilizatorilor să combine obiectele virtuale cu cele din lumea reală. Soluțiile AR bazate pe algoritmi OpenCV pot vizualiza proiecte de arhitectură și de design, pot îmbunătăți procesul de învățare în școli și universități și pot ajuta întreprinderile să își promoveze produsele și să optimizeze experiența clienților.

### Recunoașterea facială

Un sistem de recunoaștere facială detectează, clasifică și recunoaște persoanele după trăsăturile lor faciale. Această tehnologie este utilizată pe scară largă în diferite aplicații, inclusiv în aplicațiile de editare foto și video și în social media. Algoritmii Open CV de detectare a fețelor umane pot ajuta la autentificarea utilizatorilor de smartphone-uri, la verificarea identităților în sistemele de intrare și la menținerea siguranței oamenilor în locurile de adunare publică.

## **2.3.3. OpenCV în automatizarea caselor IoT**

Casele inteligente sunt sisteme de internet al obiectelor care ajută oamenii să gestioneze funcțiile casnice. Rețelele de dispozitive IoT pot controla luminile, pot regla temperatura interioară, pot uda plantele și pot porni televizorul. Viziunea computerizată este una dintre tehnologiile care fac casele inteligente - de exemplu, un frigider inteligent poate folosi camere pentru a verifica rezervele de alimente. Dezvoltatorii pot profita de o mare varietate de algoritmi și instrumente de viziune computerizată disponibile în biblioteca OpenCV pentru a crea proiecte de case inteligente.

Asigurarea securității este o parte integrantă a automatizării caselor IoT. Astfel, soluțiile de securitate inteligente pot ajuta părinții să aibă grijă de copiii lor. Implementarea aplicațiilor de viziune computerizată pentru detectarea persoanelor îmbunătățește siguranța în multe sisteme de alarmă și de interfonie video. Implementarea recunoașterii faciale OpenCV poate împiedica străinii să intre într-o casă sau într-un apartament.

Pe lângă protejarea locuințelor de intruși, este necesar să se asigure siguranța persoanelor care locuiesc singure și care nu pot avea întotdeauna grijă de ele însele. Sistemele de detectare a persoanelor prin viziune computerizată bazate pe algoritmi OpenCV și rețele neuronale pot

monitoriza de la distanță persoanele în vârstă și persoanele cu probleme de sănătate și dizabilități. În caz de urgență, acestea pot alerta rudele sau îngrijitorii.

Cea mai mare provocare a oricărui proiect legat de detecția umană în timp real în domeniul vederii computerizate este reducerea numărului de falsuri pozitive și obținerea unei precizii ridicate de detecție.

Inițial, se pot folosi algoritmi simpli de învățare automată OpenCV pentru a diferenția posturile, dacă o persoană merge, se întinde sau cade. Implementarea modelelor Markov ascunse (HMM) ajută să prezicem o cădere pe baza comportamentului anterior al unei persoane. Vom detecta astfel contururile corpului, vom defini centrul de masă al corpului și vom analiza abaterile de la aceste date cadru cu cadru. Cu cât deviațiile sunt mai explicite, cu atât mai mare este probabilitatea de cădere.

Pentru a obține o precizie mai bună în detectarea corpului uman, se pot construi rețele neuronale bazate pe parametri biometrici și date biomecanice. Pentru a antrena modelele, se vor folosi seturi de date sintetice cu ajutorul serviciilor de simulare. Unity, Gazebo și alte platforme de simulatoare oferă medii de testare care simplifică și accelerează procesul de instruire.

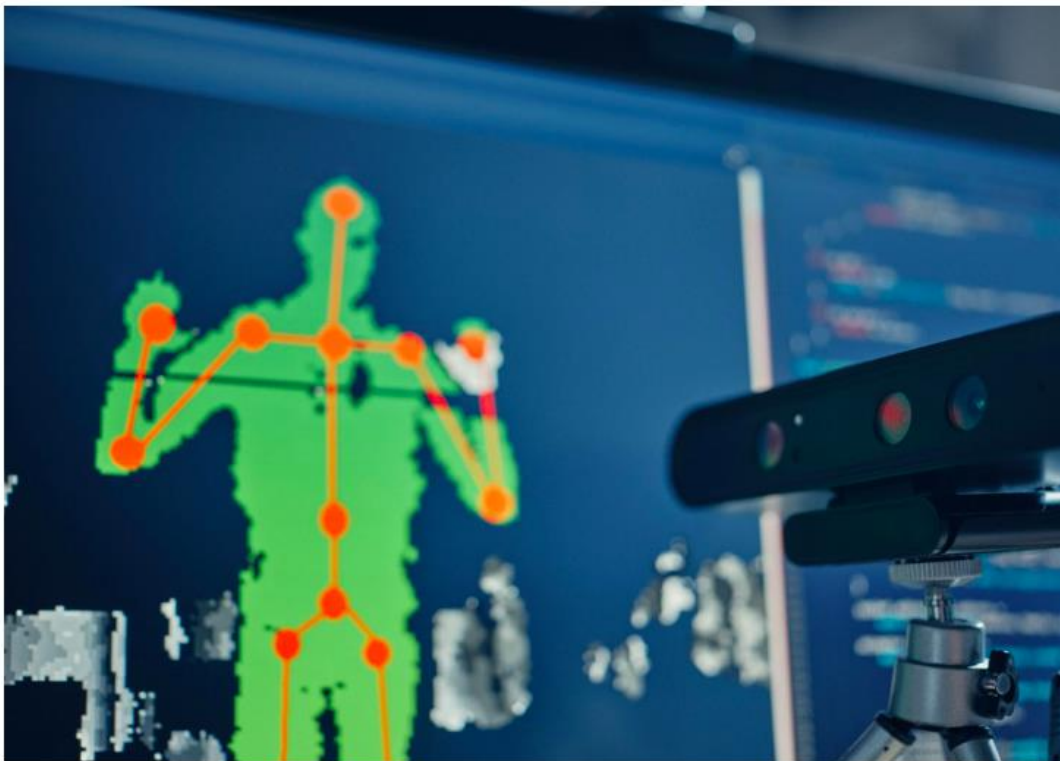


Figura 2.13: Sistem de prevenire a căderilor persoanelor bazat pe viziune computerizată pentru locuințe inteligente



În concluzie, implementarea detectării persoanelor cu ajutorul vederii computerizate poate deveni mai complicată în spațiile interioare, cum ar fi locuințele. O persoană poate fi ascunsă în spatele unei piese de mobilier, astfel încât camera nu va capta întregul corp și, prin urmare, sistemul nu va obține datele biometrice complete.

Cea mai eficientă metodă de rezolvare poate fi reprezentată de combinarea rețelelor neuronale cu arbori de decizie - algoritmi clasici de învățare automată incluși în OpenCV pentru detectarea mișcării. Acești algoritmi au avantaje semnificative: sunt rapizi și ușor de înțeles, pot învăța din cantități mici de date sau atunci când lipsesc unele date.

Rețelele IoT inteligente pentru locuințe ușurează viața oamenilor din multe puncte de vedere. Acestea automatizează rutinele zilnice, controlează energia și asigură siguranța gospodăriei. Casele inteligente moderne se bazează adesea pe tehnologii de învățare automată și de inteligență artificială. De exemplu, sistemele de monitorizare la distanță se pot baza pe algoritmi de viziune computerizată pentru detectarea persoanelor.

Un sistem de monitorizare la distanță integrat într-o soluție de automatizare a locuinței poate avea grijă de persoanele în vârstă și de persoanele cu dizabilități care trăiesc independent. Camerele de luat vederi urmăresc activitățile oamenilor, iar algoritmi analizează comportamentele acestora, identificând urgențele.

OpenCV este o alegere populară pentru proiectele IoT de automatizare a locuințelor cu viziune computerizată. Este un ecosistem open-source de instrumente, algoritmi și rețele neuronale care îndeplinesc numeroase funcții. OpenCV este una dintre bibliotecile pe care le folosim pentru a construi aplicații CV. De exemplu, algoritmi OpenCV de detectare a mișcării pot identifica și urmări persoane în timp real.

## Capitolul 3. Senzorii Arduino. Tipuri și aplicații

Atunci când un pasionat de electronică începe să proiecteze orice proiect, principalul factor de îngrijorare este problema compatibilității dintre hardware și IDE-urile software disponibile. Inventat în jurul anilor 2000, Arduino a venit ca un răspuns la astfel de probleme. Arduino este produsul efortului depus de un grup de absolvenți italieni și de un profesor de la institutul de design interactiv din Ivrea. Acest microcontroler poartă numele regelui italian Arduin de Ivrea din secolul al XI-lea. Factorul crucial în succesul acestei platforme de microcontroler este caracterul său Open-Source. Odată cu succesul acestui proiect, ulterior au fost concepute multe produse noi care sunt compatibile cu această platformă, cum ar fi senzorii Arduino.

### 3.1. Ce sunt senzorii Arduino?

Datorită naturii sale open-source, Arduino a devenit un fenomen global. Acesta a evoluat ca o platformă de prototipuri pentru pasionați, artiști, designeri și, mai ales, pentru studenții care sunt noi în lumea proiectelor electronice.



Figura 3.1.: Arduino

Arduino vine cu un microcontroler și un IDE software pentru a încărca codul în placa hardware. Văzând popularitatea lui Arduino în rândul pasionaților, au fost lansați mulți senzori compatibili cu Arduino. Există diverse tipuri de senzori Arduino disponibile pe piață. Acești senzori îl ajută pe Arduino să interacționeze cu mediul înconjurător și pentru proiectarea de noi aplicații.

## **Principiul de lucru**

Microcontrolerele care au precedat Arduino nu au un IDE software pentru încărcarea codului în hardware. Trebuia să se utilizeze un dispozitiv hardware separat pentru a încărca codul în hardware. Datorită acestei caracteristici de flexibilitate, este ușor de interfațat senzorii cu Arduino. Deoarece microcontrolerul oferă deja un IDE software pentru programare, singurul hardware necesar pentru a interfața acești senzori cu Arduino este placa Breadboard și firele de conectare. Codul poate fi scris în Arduino IDE și încărcat. Pentru interfațare sunt necesare sursa de alimentare, masa, placa de pâine și firele de conectare.

### **3.1.1 Aplicații ale senzorilor Arduino**

Există multe proiecte concepute cu ajutorul senzorului Arduino pentru diverse aplicații. Se spune că Arduino este folosit pentru a transforma o idee de vis în realitate. Modulul cu ultrasunete este utilizat pentru detectarea distanței fără contact. Acesta utilizează sonarul pentru funcționarea sa. Senzorul de evitare a obstacolelor cu infraroșu IR detectează obiectele care se află în fața sa și generează un semnal digital. Acesta este utilizat în roboți.

Higrometrul de sol este un senzor de umiditate a solului. Acesta generează un semnal digital atunci când umiditatea din sol crește peste o anumită valoare de prag. Instalația automată de autoapărare este proiectată folosind acest senzor cu Arduino. Senzorul microscopului este utilizat pentru a detecta sunetul. Acesta generează un semnal atunci când intensitatea sunetului detectat crește peste o anumită valoare de prag.

Senzorul digital de presiune barometrică este utilizat pentru a măsura presiunea absolută a mediului înconjurător. Înălțimea robotului sau a proiectilului poate fi măsurată cu ajutorul acestui senzor. Pentru detectarea luminii, se utilizează modulul senzor cu fotorezistență. Sistemul de lumină de securitate nocturnă utilizează acest senzor cu Arduino. Senzorul de temperatură este utilizat pentru a detecta temperatura ambientală.

Pentru a detecta gazele toxice, cum ar fi GPL, i- Butan, propan, alcool, etc... se utilizează senzorul de gaz MQ-2. Senzorul de ploaie este utilizat pentru monitorizarea vremii. Pentru a detecta flacăra și lumina obișnuită se utilizează senzorul de flăcără. Senzorul PIR este utilizat pentru a detecta mișcarea oamenilor și a animalelor de companie. Senzorul pentru ecran tactil este utilizat pentru proiectarea circuitului de atenuare a atingerii cu ajutorul Arduino.

### 3.2. Exemple de senzori Arduino

Există multe tipuri de senzori Arduino disponibile în prezent. Unele dintre ele sunt:

- HC- SR04 Modul cu ultrasunete
- IR Senzor infraroșu de evitare a obstacolelor
- Modul de detectare a higrometrului de sol
- Senzor de umiditate a solului
- Senzor cu microfon
- Senzor digital de presiune barometrică
- Senzor de fotorezistență
- Senzor termic digital - Senzor de temperatură
- Modul de codificare rotativă
- Senzor de gaz MQ-2
- SW-420 Senzor de mișcare
- Modul de buzzer pasiv
- Modul senzor de viteză
- Senzor de detectare a flăcării în infraroșu IR
- Modul releu cu 2 canale de 5V
- Modul de alimentare Breadboard 3,3V
- Senzor infraroșu piroelectric HC- SR501
- Modul accelerometru
- DHT11 Senzor de temperatură și umiditate
- Transmițător/receptor RF 433MHz

## HC- SR04 Modul cu ultrasunete

Un senzor de distanță cu ultrasunete HC-SR04 constă de fapt din două traductoare cu ultrasunete. Unul acționează ca un emițător care convertește semnalul electric în impulsuri sonore ultrasonice de 40 KHz. Celălalt acționează ca un receptor și ascultă impulsurile transmise. Atunci când receptorul primește aceste impulsuri, produce un impuls de ieșire a cărui lățime este proporțională cu distanța obiectului din față. Acest senzor asigură o detectare excelentă a distanței fără contact între 2 cm și 400 cm (~13 picioare) cu o precizie de 3 mm. Deoarece funcționează la 5 volți, poate fi conectat direct la un Arduino sau la orice alt microcontroler logic de 5V.

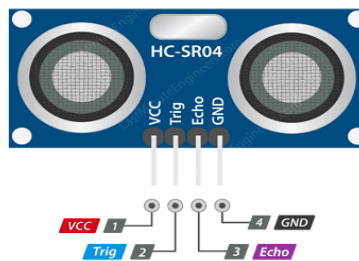


Figura 3.2: HC-SR04 Senzor cu ultrasunete

VCC alimentează senzorul cu ultrasunete HC-SR04. Se poate conecta la ieșirea de 5V de la Arduino. Pinul Trigger (Declanșator) este utilizat pentru a declanșa impulsurile sonore ultrasonice. Prin setarea acestui pin la HIGH timp de  $10\mu\text{s}$ , senzorul inițiază o rafală de ultrasunete. Pinul Echo (Eco) devine înalt atunci când este transmisă rafala ultrasonică și rămâne înalt până când senzorul primește un ecou, după care devine scăzut. Măsurând timpul în care pinul Echo rămâne ridicat, se poate calcula distanța. GND este pinul de masă.

### Cum funcționează?

Totul începe atunci când pinul de declanșare este setat HIGH timp de  $10\mu\text{s}$ . Ca răspuns, senzorul transmite o rafală ultrasonică de opt impulsuri la 40 kHz. Acest model de 8 impulsuri este special conceput pentru ca receptorul să poată distinge impulsurile transmise de zgomotul ultrasonic ambiental. Aceste opt impulsuri ultrasonice se deplasează prin aer, îndepărtându-se de emițător. Între timp, pinul de ecou trece la HIGH pentru a iniția semnalul de ecou înapoi.

Dacă aceste impulsuri nu sunt reflectate înapoi, semnalul de ecou se oprește și devine scăzut după 38ms (38 milisecunde). Astfel, un impuls de 38ms indică faptul că nu există niciun obstacol în raza de acțiune a senzorului.

În cazul în care acele impulsuri sunt reflectate înapoi, pinul de ecou devine scăzut imediat ce semnalul este recepționat. Acest lucru generează un impuls pe pinul de ecou a cărui lățime variază între 150  $\mu$ s și 25 ms, în funcție de timpul necesar pentru recepționarea semnalului.

### **Cablarea unui senzor HC-SR04 la un Arduino**

Conectarea senzorului HC-SR04 la Arduino este foarte ușoară. Se începe prin plasarea senzorului pe breadboard. Vom conecta pinul VCC la pinul de 5V de pe Arduino și pinul GND la pinul de masă. Apoi vor urma pinii de trigonometrie și de ecou la pinii digitali #9 și, respectiv, #10.

Când vom termina, ar trebui să avem ceva care să semene cu ilustrația prezentată mai jos.

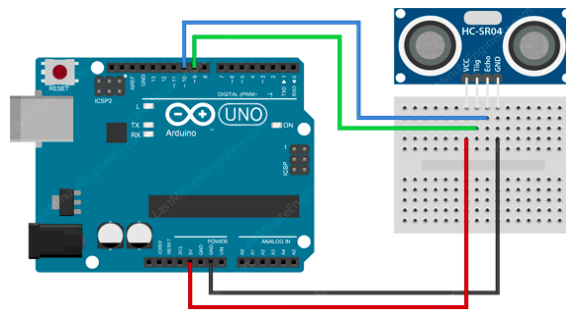


Figura 3.3: Conectare senzor la un Arduino

### **Senzor de evitare a obstacolelor în infraroșu IR**

Placa cu senzor infraroșu IR Obstacle Avoidance Sensor este o soluție necostisitoare pentru detectarea evitării obstacolelor pentru robotică, mașini inteligente și alte utilizări electronice.



Figura 3.4: Senzor de evitare a obstacolelor în infraroșu IR

Modulul cu infraroșu pentru senzori de obstacole are un emițător și un receptor IR încorporat care trimite energie IR și caută energia IR reflectată pentru a detecta prezența oricărui obstacol în fața modulului sensor. Modulul are un potențiometrul încorporat care le permite utilizatorilor să regleze intervalul de detecție. Senzorul are un răspuns foarte bun și stabil, chiar și în lumina ambientală sau în întuneric complet. Senzorii de evitare a obstacolelor vin de obicei în două tipuri - cu 3 și 4 pini.

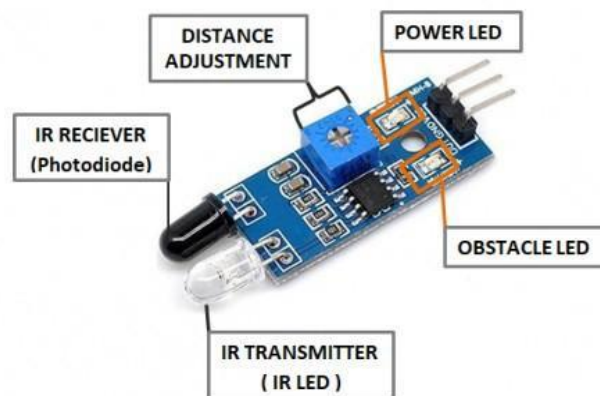


Figura 3.5: Mod de funcționare

### Principiul de funcționare

Transmițătorul IR trimite un semnal infraroșu care, în cazul unei suprafețe reflectante (de exemplu, de culoare albă), ricoșează în anumite direcții, inclusiv în cea a receptorului IR care captează semnalul detectând obiectul. Atunci când suprafața este absorbantă, semnalul IR nu este reflectat și obiectul nu poate fi detectat de senzor. Acest rezultat ar avea loc chiar dacă obiectul este absent.

Modulul are 3 pini - Vcc, GND și ieșirea. De asemenea, avem 2 LED-uri, unul este pentru alimentare adică se aprinde atunci când este conectat la alimentare. Celălalt este pentru detectarea obstacolelor.

### Conexiuni

1. Conectăm Vcc al modulului senzorului la Vcc al plăcii Arduino
2. Conectăm GND al modulului senzorului la GND al plăcii Arduino.
3. Conectăm pinul de ieșire al modulului senzorului la pinul 7 al plăcii Arduino.

4. Atunci când senzorul de evitare a obstacolelor detectează un obstacol, LED-ul se va aprinde. În caz contrar, acesta va fi stins.

### **Modul de detectare a higrometrului de sol**

Acesta este un senzor de apă simplu, poate fi utilizat pentru a detecta umiditatea solului atunci când modulul de deficit de umiditate a solului emite un nivel ridicat și, invers, emite un nivel scăzut. Utilizăm acest senzor pentru a construi un dispozitiv automat de udare a plantelor, astfel încât plantele din grădina noastră să nu mai aibă nevoie de oameni pentru a fi gestionate. Modulul de umiditate a solului este cel mai sensibil la umiditatea mediului ambiant, este utilizat în general pentru a detecta conținutul de umiditate al solului.

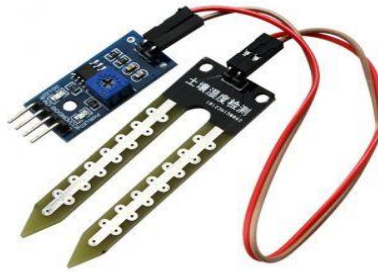


Figura 3.6: Modul de detectare a higrometrului de sol

Atunci când umiditatea solului este mai mică decât o valoare de prag stabilită, portul DO iese la nivel înalt. Atunci când umiditatea solului depășește valoarea de prag, modulul DO iese la nivel scăzut. Ieșirea digitală D0 poate fi conectată direct cu microcontrolerul. Ieșirea analogică AO și modulul AD conectate prin intermediul convertorului AD, putem obține valori mai precise ale umidității solului.

### **Senzor de umiditate a solului**

Senzorul de umiditate a solului este un tip de senzor utilizat pentru a măsura conținutul volumetric de apă din sol. Deoarece dimensiunea gravimetrică directă a umidității solului necesită eliminarea, uscarea, precum și cântărirea probei. Acești senzori măsoară conținutul volumetric de apă nu direct cu ajutorul unor alte reguli ale solului, cum ar fi constanta dielectrică, rezistența electrică, altfel interacțiunea cu neutronii și înlocuirea conținutului de umiditate.



Relația dintre proprietatea calculată, precum și umiditatea solului trebuie ajustată și se poate schimba în funcție de factori ecologici precum temperatura, tipul de sol, altfel conductivitatea electrică. Emisia de microunde care este reflectată poate fi influențată de umiditatea solului și poate fi utilizată în principal în agricultură și în teledetecție în cadrul hidrologiei.

### **Senzor cu microfon**

Senzorul de sunet al microfonului, după cum îi spune și numele, detectează sunetul. Acesta oferă o măsurătoare a intensității unui sunet. Există o mare varietate de astfel de senzori. În Figurade mai jos îl putem vedea pe cel mai frecvent utilizat cu Arduino.

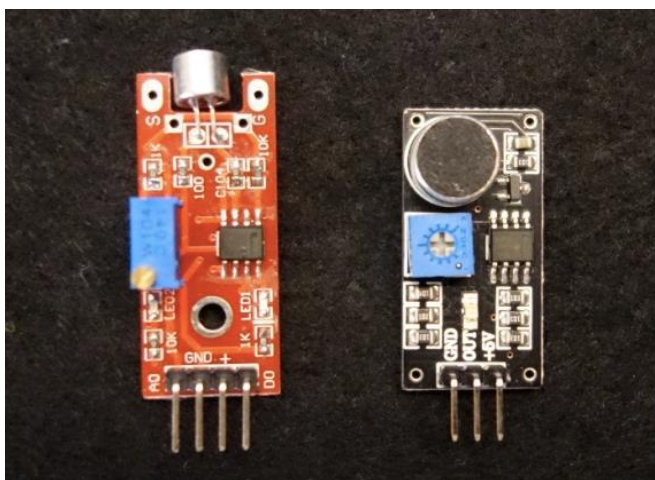


Figura 3.7: Senzor cu microfon

În partea stângă, vedem KY-038, iar în dreapta senzorul de sunet microfonic LM393. Ambele module de senzori au un potențiomtru încorporat pentru a regla sensibilitatea pinului de ieșire digitală.

### **Senzor digital de presiune barometrică**

Senzorul de presiune barometrică, cunoscut și ca o formă mai nouă a barometrului, este un instrument compatibil cu Arduino utilizat pentru măsurarea presiunii atmosferice în medii. Astfel de măsurători permit, în principal, prognozarea schimbărilor pe termen scurt ale vremii.

Odată cu schimbările de design de-a lungul anilor, senzorii de presiune barometrică sunt acum miniaturizați pentru utilizări pe smartphone-uri și plăci cu microcontroler, cum ar fi Arduino, unde unul dintre cele mai comune fiind BMP180, BMP280, BME280.

### **Senzor de fotorezistență**

Fotorezistoarele, cunoscute și sub denumirea de rezistențe dependente de lumină (LDR), sunt dispozitive sensibile la lumină, utilizate cel mai adesea pentru a indica prezența sau absența luminii sau pentru a măsura intensitatea luminii. În întuneric, rezistența lor este foarte mare, uneori de până la 1 M $\Omega$ , dar atunci când senzorul LDR este expus la lumină, rezistența scade dramatic, chiar până la câțiva ohmi, în funcție de intensitatea luminii.

LDR-urile au o sensibilitate care variază în funcție de lungimea de undă a luminii aplicate și sunt dispozitive neliniare. Ele sunt utilizate în multe aplicații, dar această funcție de detectare a luminii este adesea realizată de alte dispozitive, cum ar fi fotodiodele și fototranzistorii. Unele țări au interzis LDR-urile fabricate din plumb sau cadmiu din motive de siguranță pentru mediu.

### **Senzor termic digital - Senzor de temperatură**

Temperatura este cea mai des măsurată mărime de mediu. Acest lucru ar fi de așteptat, deoarece majoritatea sistemelor fizice, electronice, chimice, mecanice și biologice sunt afectate de temperatură. Anumite reacții chimice, procese biologice și chiar circuite electronice funcționează cel mai bine în intervale limitate de temperatură.

Temperatura este una dintre cele mai des măsurate variabile și, prin urmare, nu este surprinzător faptul că există multe modalități de a o detecta. Detectarea temperaturii se poate face fie prin contact direct cu sursa de încălzire, fie de la distanță, fără contact direct cu sursa, folosind în schimb energia radiată. Există o mare varietate de senzori de temperatură pe piața actuală, inclusiv termocuple, detectoare de temperatură cu rezistență (RTD), termistori, senzori cu infraroșu și senzori cu semiconductori.

### **Modul de codificare rotativă**

Codificatoarele rotative sunt utilizate în multitudinea de dispozitive comune pe care le vedem în fiecare zi. De la imprimante și obiective fotografice la mașini CNC și robotică, codurile rotative sunt mai aproape de noi decât credem. Cel mai popular exemplu de utilizare a unui codificator rotativ în viața de zi cu zi este butonul de control al volumului de la un radio auto.

Un codificator rotativ este un tip de senzor de poziție care convertește poziția unghiulară (rotația) unui buton într-un semnal de ieșire care este utilizat pentru a determina în ce direcție este rotit butonul. Există două tipuri de encodere rotative - absolute și incrementale. Codificatorul

absolut raportează poziția exactă a butonului în grade, în timp ce codicatorul incremental raportează câte trepte s-a deplasat arborele. Codicatorul rotativ utilizat în acest tutorial este de tip incremental.

### **Senzor de gaz MQ-2**

MQ2 este unul dintre senzorii de gaz utilizați în mod obișnuit în seria de senzori MQ. Este un senzor de gaz de tip MOS (Metal Oxide Semiconductor), cunoscut și sub numele de Chemiresistori, deoarece detecția se bazează pe modificarea rezistenței materialului de detecție atunci când gazul intră în contact cu materialul. Utilizând o rețea simplă de divizoare de tensiune, pot fi detectate concentrațiile de gaz.

Senzorul de gaz MQ2 funcționează la 5V DC și consumă aproximativ 800mW. Acesta poate detecta concentrații de GPL, fum, alcool, propan, hidrogen, metan și monoxid de carbon de la 200 la 10.000 ppm.

### **SW-420 Senzor de mișcare**

SW-420 este un modul de senzor de vibrații eficient din punct de vedere al costurilor, format din SW-420 și comparatorul LM393. Modulul este încorporat cu un senzor de vibrații SW-420, un potențiomtru de 10K pentru a modifica sensibilitatea și un comparator LM393 care produce o ieșire digitală netedă. Comparatorul LM393 utilizează o presetare pentru a detecta vibrațiile din mediul înconjurător și menține o stare.

Acesta emite o ieșire logică ridicată dacă sunt detectate vibrații, altfel va rămâne în starea implicită de nivel logic scăzut. Modulul este o interfață la îndemână, ușor de utilizat cu un dispozitiv care își găsește aplicațiile în proiectele DIY (DIY) împreună cu aplicațiile comerciale.

### **Modul de buzzer pasiv**

Modulul KY-006 Passive Piezoelectric Buzzer poate produce o gamă de tonuri de sunet în funcție de frecvența semnalului de intrare. Pentru a genera sunete cu un singur ton, vom utiliza modulul KY-012 Active Buzzer. Este compatibil cu microcontrolerele populare, cum ar fi Arduino, Raspberry Pi, ESP32 și altele.

Acest modul este format dintr-un buzzer pasiv și 3 pini de tip header de sex masculin. Îl putem utiliza pentru a genera tonuri între 1,5 și 2,5kHz, pornindu-l și oprindu-l în mod repetat la

diferite frecvențe, fie folosind întârzieri, fie PWM. Se va conecta cu semnalul modulului (S) la pinul 8 de pe Arduino și masa (-) la GND. Pinul din mijloc nu este utilizat.

### **Modul senzor de viteză**



Figura 3.8: Modul senzor de viteză

În Figura 3.8. avem un modul de senzor de viteză a motorului, obiectivul principal al lui fiind acela de a verifica rata unui motor electric. Modulul poate fi utilizat în asociere cu un microcontroler pentru detectarea vitezei motorului, numărarea impulsurilor, limitarea poziției etc. În principiu, orice contor de viteză măsoară pur și simplu rata la care se produce un anumit eveniment. De obicei, acest lucru se face prin numărarea evenimentelor pentru o anumită perioadă de timp (interval de integrare) și apoi prin simpla împărțire a numărului de evenimente la timp pentru a obține rata.

Practic, modulul de senzor de viteză a motorului compatibil cu microcontrolerul descris este un dispozitiv simplu care produce trenuri de impulsuri procesate atunci când calea vizuală a senzorului său optic este întreruptă fizic de un fel de roată cu fante sau de un mecanism similar (un senzor optic constă în mod obișnuit dintr-o diodă emițătoare de lumină care asigură iluminarea și un fototranzistor care detectează prezența sau absența acestei iluminări). Senzorul optic transmisiv utilizat aici constă într-o diodă emițătoare de lumină infraroșie și un fototranzistor. Acest lucru previne atât interferențele cauzate de surse de lumină externă rătăcitoare, cât și faptul că cele două componente sunt potrivite pentru o anumită frecvență de radiație, ceea ce le face și mai imune la interferențe nedorite.

## **Senzor de detectare a flăcării în infraroșu IR**

Un modul senzor de flăcără sau un modul senzor de incendiu este un dispozitiv electronic de dimensiuni mici care poate detecta o sursă de foc sau orice altă sursă de lumină puternică. Acest senzor detectează, în principiu, lungimea de undă a luminii IR (infraroșu) între 760 nm - 1100 nm care este emisă de flăcără de incendiu sau de sursa de lumină. Sensorul de flăcără este prevăzut cu un senzor fototranzistor YG1006, care este un senzor de mare viteză și sensibilitate ridicată.

Pe piață sunt disponibile două tipuri de module de senzori de flăcără cu infraroșu IR, unul cu trei pini (D0, Gnd, Vcc) și altul cu patru pini (A0, D0, Gnd, Vcc), ambele putând fi utilizate cu ușurință cu Arduino și alte plăci cu microcontroler.

Două tipuri de senzori de flăcără sunt disponibile pe piață, ambele tipuri fiind alcătuite din aceleași componente. De asemenea, parametrii ambilor senzori, cum ar fi tensiunea de funcționare, cipul comparator de curent, tipul de senzor, gama de spectru, unghiul de detecție și intervalul de detecție, totul este similar. Dar, aceste module au o diferență majoră, și anume un modul este format din 3 pini (D0, Gnd, Vcc) și poate furniza doar o ieșire digitală. Un alt modul este format din 4 pini (A0, D0, Gnd, Vcc) și poate furniza ieșire digitală și ieșire analogică.

## **Modul releu cu 2 canale de 5V**

Modulul de releu cu 2 canale de 5V este o placă de interfață de releu, poate fi controlată direct de o gamă largă de microcontrolere, cum ar fi Arduino, AVR, PIC, ARM și așa mai departe. Folosește un semnal de control declanșat la nivel scăzut (3,3-5VDC) pentru a controla releul.

Declanșarea releului acționează contactele normal deschise sau normal închise. Este utilizat frecvent într-un circuit de control automat. Pentru a simplifica, este un comutator automat pentru a controla un circuit de curent mare cu un semnal de curent mic. 5 V interval de tensiune de intrare a semnalului de releu de 5V, 0-5V. VCC alimentare a sistemului. Releu JD-VCC în sursa de alimentare. JD-VCC și VCC pot fi un scurtcircuit.

## **Modul de alimentare Breadboard 3,3V**

Modulul de alimentare cu ieșire de 3,3V/5V MB102 Breadboard este o componentă de breadboard ușor de utilizat, foarte utilă, care poate fi adăugată la orice proiect legat de breadboard unde sunt necesare cerințe de alimentare de 5V, 3,3V sau ambele.

Ușurința sa de utilizare permite utilizatorilor să conecteze orice unitate de alimentare de curent continuu care are o ieșire de alimentare de 6,5-12 VDC de la o mufă de butoi. Placa are două canale independente de ieșire de alimentare pentru breadboard-uri. Aceste canale de alimentare pot fi configurate independent pentru operațiuni de 3,3V, 0V și 5V.

Modulul oferă, de asemenea, un comutator de apăsare pentru a opri și porni întregul modul de alimentare. O caracteristică suplimentară este o intrare USB cu două prize de 5V, două de 3,3V și 4 prize GND pentru cerințe suplimentare de pin de alimentare. LED-ul de alimentare va notifica utilizatorul cu privire la starea de disponibilitate a alimentării de intrare.

### **Senzor infraroșu piroelectric HC- SR501**

Fie că ne dorim să construim o alarmă antiefracție pentru casă sau o cameră de urmărire, sau poate că vrem să trezim recuzita animată de Halloween atunci când vin la ușă cei care fac farse, atunci ar trebui să luăm în considerare cu siguranță achiziționarea unui senzor cu infraroșu pasiv (PIR) HC-SR501.

Senzorul PIR ne permite să detectăm când o persoană sau un animal se mișcă în sau în afara razei de acțiune a senzorului. Acest senzor este cel pe care îl vom găsi în majoritatea sistemelor de securitate moderne, întrerupătoare automate de lumină, deschizătoare de uși de garaj și aplicații similare în care dorim să reacționăm la mișcare.

Toate obiectele, inclusiv corpul uman, la temperaturi peste zero absolut (0 Kelvin / -273,15 °C) emit energie termică sub formă de radiații infraroșii. Cu cât un obiect este mai cald, cu atât mai multe radiații emite. Această radiație nu este vizibilă pentru ochiul uman, deoarece este emisă la lungimi de undă infraroșii. Senzorul PIR este special conceput pentru a detecta astfel de niveluri de radiații infraroșii.

Un senzor PIR este alcătuit din două părți principale: un senzor piroelectric, pe care îl puteți vedea în imaginea de mai jos ca un metal rotund cu un cristal dreptunghiular în centru și o lentilă specială numită lentilă Fresnel, care focalizează semnalele infraroșii pe senzorul piroelectric.

### **Modul accelerometru**

Un accelerometru este un dispozitiv electromecanic care măsoară forța de accelerație. Acesta arată accelerația, numai datorită cauzei gravitației, adică forței g. Acesta măsoară accelerația în

unitatea g. Pe Pământ, 1g înseamnă că există o accelerație de 9,8 m/s<sup>2</sup>. Pe Lună, aceasta este 1/6 din cea a Pământului, iar pe Marte este 1/3 din cea a Pământului.

Accelerometrul poate fi utilizat pentru aplicații de detectare a înclinării, precum și a accelerației dinamice rezultate din mișcare, șocuri sau vibrații.

### DHT11 Senzor de temperatură și umiditate

DHT11 este un senzor digital de temperatură și umiditate de bază, foarte ieftin. Folosește un senzor de umiditate capacitiv și un termistor pentru a măsura aerul înconjurător și emite un semnal digital pe pinul de date (nu este nevoie de pinii de intrare analogici). Este destul de simplu de utilizat, dar necesită o sincronizare atentă pentru a prelua datele. Singurul dezavantaj real al acestui senzor este că puteți obține date noi de la el doar o dată la 2 secunde, astfel încât, atunci când utilizați biblioteca noastră, citirile senzorului pot fi vechi de până la 2 secunde.

În comparație cu DHT22, acest senzor este mai puțin precis, mai puțin exact și funcționează într-un interval mai mic de temperatură/umiditate, dar este mai mic și mai ieftin.

### Transmițător/receptor RF 433MHz

Acest mic modul este un emițător între doi. Este foarte simplu, așa cum pare. Inima modulului este rezonatorul SAW care este acordat pentru funcționarea la 433,xx MHz. Există un tranzistor de comutare și câteva componente pasive.

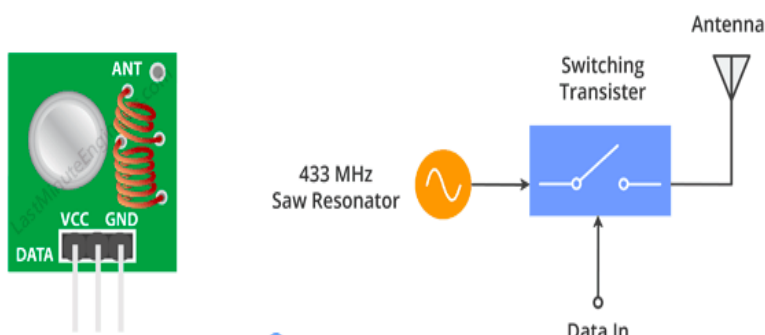


Figura 3.9: Modulul emițător și receptor RF de 433 MHz

Atunci când se aplică un nivel logic HIGH la intrarea DATA, oscilatorul funcționează producând o undă purtătoare de ieșire RF constantă la 433,xx MHz, iar când intrarea DATA este dusă la nivelul logic LOW, oscilatorul se oprește. Această tehnică este cunoscută sub numele de Amplitude Shift Keying.

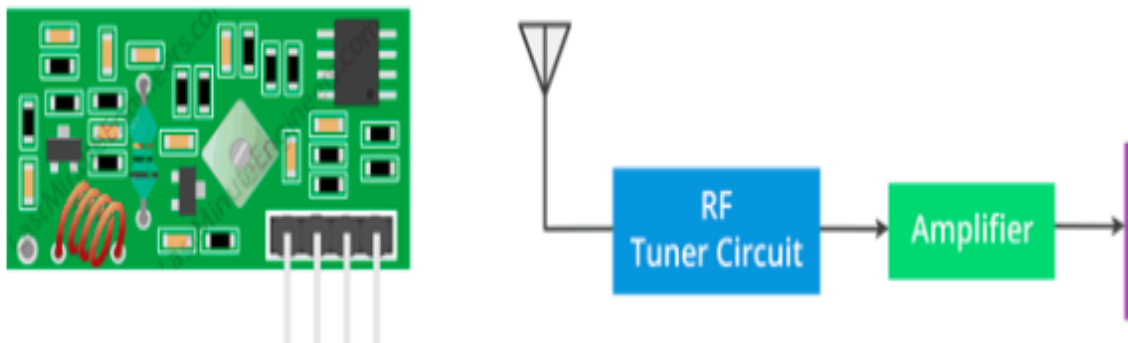


Figura 3.10: Modul receptor

În Figura 3.10. avem un modul receptor. Deși pare complex, este la fel de simplu ca și modulul emițător. Acesta este format dintr-un circuit acordat RF și câteva amplificatoare OP pentru a amplifica unda purtătoare primită de la emițător.

Semnalul amplificat este alimentat în continuare la o buclă de blocare a fazei (PLL - Phase Lock Loop) care permite decodorului să se "fixeze" pe un flux de biți digitali, ceea ce oferă o ieșire decodată mai bună și o imunitate mai bună la zgomot.

În concluzie, acești senzori Arduino au făcut posibilă implementarea multor proiecte electronice. Câteva exemple de astfel de proiecte sunt: Suntracker folosind LDR cu Arduino, Arduino alarmă pentru apa de ploaie, Robot controlat prin gesturi bazat pe accelerometru cu Arduino, Urmaș de linie bazat pe senzor IR, Alarmă de mișcare bazată pe senzor IR, Alarmă de ușă folosind senzorul cu ultrasunete, Măsurarea distanței folosind senzorul cu ultrasunete, Băț orb inteligent bazat pe Arduino, Senzor PIR pentru controlul aparatelor electrocasnice folosind Arduino etc.

Arduino este prima alegere pentru studenți și pentru persoanele care încep să se familiarizeze cu electronica pentru a concepe un nou proiect. Senzorii Arduino pot fi utilizați și cu alte microcontrolere. Arduino IDE conține diverse biblioteci utile pentru interfațarea diferitelor tipuri de senzori. Singurele excepții sunt senzorii care necesită o viteză de procesare mai mare decât poate oferi Arduino.



## **Capitolul 4 - Descriere soluție de implementare**

După ce am discutat în primele trei capitole despre partea teoretică, a venit momentul și pentru partea practică. Astfel, în acest capitol, vom pune în aplicare tot ce am prezentat în toate paginile mai sus. Așadar, vom împărți partea aceasta de lucrare în trei mari subcapitole. În primul subcapitol vom prezenta algoritmul de predicție și generarea comportamentului uman. În subcapitolul doi vom avea ca temă recunoașterea facială, iar în ultimul, vom dezbate subiectul reacțiilor pe care orice persoane le are pe parcursul vieții (exemple: tristețe, veselie, nervozitate). Și, să le luăm pe rând.

### **4.1. Algoritm de predicție și generarea comportamentului uman**

În această situație, procesul de simulare a fost gândit pentru 3 camere: baie (bathroom), camera de zi (dayroom), camera de noapte (bedroom). Fiecărei camere menționate anterior îi sunt atribuite anumite dispozitive reprezentative. În exemplul despre care vorbim, camera de noapte va cuprinde o lumină, un televizor și două laptopuri), baia va avea în subordine două lumini, iar camera de zi, o lumină și un televizor. În funcție de cerințele utilizatorului, noi putem adăuga mai multe dispozitive, implicit mai multe camere în aplicație.

Pentru implementarea acestui proces, am apelat la placa Arduino, doi senzori ultrasonici HC-SR04 și un laptop, ca parte hardware, iar ca parte software, am folosit limbajul de programare Python. Așa cum am menționat anterior, Python este cel mai utilizat limbaj în implementarea unor astfel de aplicații. Dacă am menționat ce am folosit și de ce am avut nevoie pentru întreaga operațiune, în următoarele paragrafuri vom vedea cum se implementează procesul.

#### **Descriere proces**

Procesul de studiu al algoritmului de predicție și generare a fost implementat în felul următor. În primul rând, am folosit algoritmul bazat pe modelul lui Markov, despre care am vorbit în primul capitol. Așa cum am observat și acolo, algoritmii modelului Markov au câteva proprietăți interesante, care fac atractivă utilizarea lor. În primul rând, AM este o schemă de calcul generală. Ea poate face tot ceea ce pot face alte scheme de calcul și, prin urmare, este foarte puternică. Este echivalent, de exemplu, cu mașina Turing. În al doilea rând, atunci când se definește un limbaj cu

ajutorul MA, trebuie să se scrie o listă de reguli care seamănă destul de mult cu regulile BNF. Cu toate acestea, spre deosebire de BNF, la orice set de reguli MA corespunde un parser unic.

Algoritmul bazat pe acest model descris mai sus poartă denumirea de LZ78. Schemele bazate pe LZ78 funcționează prin introducerea frazelor într-un dicționar și apoi, atunci când se găsește o apariție repetată a acelei fraze, se afișează indexul dicționarului în locul frazei. La fiecare pas, LZ78 va trimite la ieșire o pereche (i,a), unde i este un indice al frazei în dicționar, iar a este următorul simbol care urmează imediat după fraza găsită. Dicționarul este reprezentat ca un trie cu noduri numerotate. Dacă mergem de la rădăcină la un anumit nod, vom obține o frază din textul de intrare. În fiecare etapă, căutăm cea mai lungă frază din dicționar, care ar corespunde părții neprocesate a textului de intrare. Indexul acestei fraze, împreună cu simbolul care urmează părții găsite în textul de intrare, sunt apoi trimise la ieșire. Fraza veche extinsă cu noul simbol este apoi introdusă în dicționar. Această nouă frază este numerotată cu cel mai mic număr posibil. Codificarea va începe cu un arbore care are un singur nod, care reprezintă șirul gol. Tot în capitolul unu am prezentat și implementarea algoritmului în limbajul de programare. Pentru a vedea dacă acel algoritm funcționează, am scris propriul nostru cod.

Continuând aplicația noastră, am ajuns la secțiunea senzori. Senzorii ultrasonici, în general, sunt folosiți pentru a-ți oferi datele în momentul în care persoana schimbă camera. Dispozitivele din camere sunt inițializate pasiv, adică placa arduino nu e conectată la perifericele din casă pentru a putea citi dacă ele sunt oprite sau pornite. Dar, avem nevoie de această inițializare pasivă a lor, cu scopul de a putea genera comportamentul uman. Partea bună este că aceste dispozitive pot fi foarte ușor conectate la placă arduino într-o casă inteligentă.

## **Implementare**

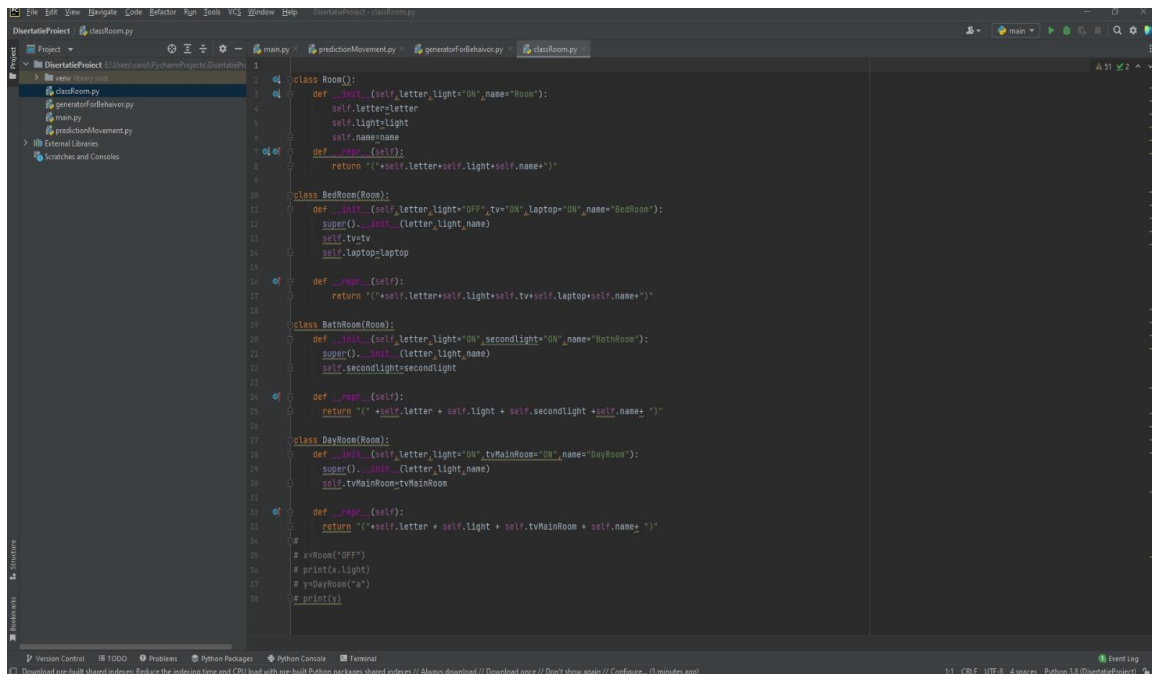
În aplicația din limbajul de programare, am folosit Orientarea pe Obiect („oriented object”) și două dintre cele mai importante proprietăți, respectiv abstractizarea și moștenirea. Vorbind de abstractizare, aceasta înseamnă ca este împărțită pe mai multe module:

- 1 - classRoom.py;
- 2 - generatorForBehaivor;
- 3 - main.py;
- 4 - predictionMovement.py.

## Modulul 1. classRoom.py

În acest modul se creează camerele casei inteligente. Pentru ca al nostru cod scris să fie optimizat, în modulul acesta am folosit proprietatea de moștenire. Și după cum se vede în atașamentul de mai jos, există o clasă „Room”, care are proprietăți generale, cum ar fi light (lumina), pe care le vor putea moșteni și celelalte clase. Cu alte cuvinte, de exemplu considerăm ca întotdeauna o camera trebuie să aibă lumină și orice cameră (dormitor, camera de zi/noapte) trebuie să aibă proprietatea iluminat.

În codul scris în imediata vecinătate, avem cele 4 clase, Room, BedRoom, BathRoom, DayRoom. Din aceste 4 clase, vom utiliza în aplicație doar 3 dintre ele, deoarece clasa „Room” va fi utilizată doar pentru moștenire. Moștenirea proprietăților în celelalte clase se face prin metoda `super()`.



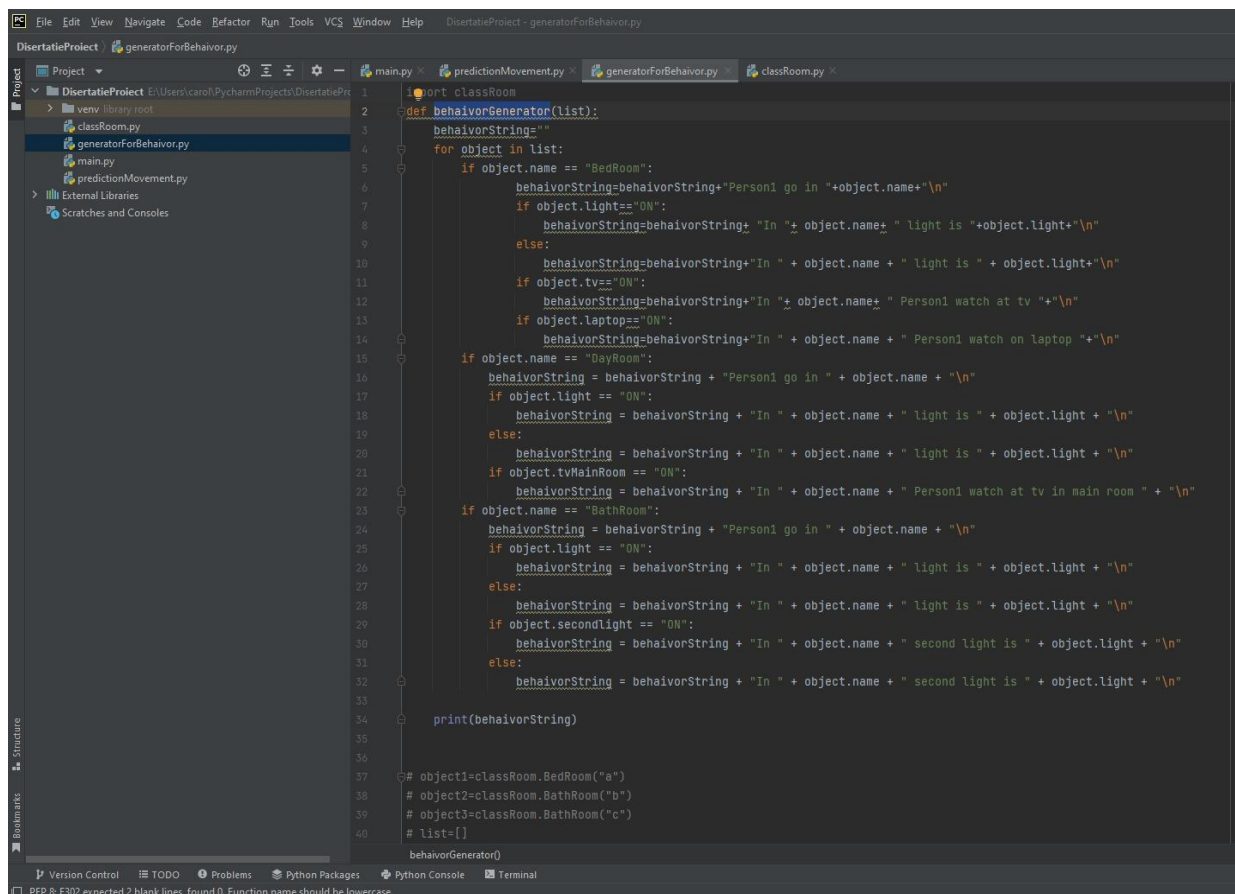
```
1 class Room():
2     def __init__(self, letter, light="ON", name="Room"):
3         self.letter=letter
4         self.light=light
5         self.name=name
6     def __repr__(self):
7         return "("+self.letter+self.light+self.name+")"
8
9 class BedRoom(Room):
10     def __init__(self, letter, light="OFF", tv="ON", laptop="ON", name="BedRoom"):
11         super().__init__(letter, light_name)
12         self.tv=tv
13         self.laptop=laptop
14     def __repr__(self):
15         return "("+self.letter+self.light+self.tv+self.laptop+self.name+")"
16
17 class BathRoom(Room):
18     def __init__(self, letter, light="ON", secondlight="ON", name="BathRoom"):
19         super().__init__(letter, light_name)
20         self.secondlight=secondlight
21     def __repr__(self):
22         return "("+self.letter + self.light + self.secondlight +self.name+ ")"
23
24 class DayRoom(Room):
25     def __init__(self, letter, light="ON", tvMainRoom="ON", name="DayRoom"):
26         super().__init__(letter, light_name)
27         self.tvMainRoom=tvMainRoom
28     def __repr__(self):
29         return "("+self.letter + self.light + self.tvMainRoom + self.name+ ")"
30
31 #
32 # x=Room("OFF")
33 # print(x.light)
34 # y=DayRoom("a")
35 # print(y)
```

Figura 4.1: Modulul classRoom.py

Analizând Figura 4.1, putem observa că fiecare camera are proprietățile ei, unele inițializate: (perifericele). Aceste proprietăți ale camerelor ne ajută atât în algoritmul de predicție, cât și pentru activitățile umane realizate în acea cameră. De exemplu, proprietatea Letter va fi folosită pentru algoritmul de predicție, iar celelalte proprietăți precum televizorul, laptopul, lumina vor fi folosite pentru generarea comportamentului uman.

## Modulul 2. generatorForBehavior

În acest modul, după cum am discutat și mai sus, se va genera comportamentul unei persoane, în funcție de interacțiunea cu dispozitivele din acea încăpere. Mai jos, vom prezenta și o generare a acestui comportament. În cazul în care aveam conectate dispozitivele pentru laptopuri și televizoarele la placa arduino, putea oferi o generare a comportamentului uman mult mult mai clară, având în vedere că ne putem folosi și de biblioteca „time” în Python.



```
1 import classRoom
2 def behaviorGenerator(list):
3     behaviorString=""
4     for object in list:
5         if object.name == "BedRoom":
6             behaviorString=behaviorString+"Person1 go in "+object.name+"\n"
7             if object.light=="ON":
8                 behaviorString=behaviorString+"In "+ object.name+ " light is "+object.light+"\n"
9             else:
10                 behaviorString=behaviorString+"In "+ object.name + " light is " + object.light+"\n"
11             if object.tv=="ON":
12                 behaviorString=behaviorString+"In "+ object.name+ " Person1 watch at tv "++"\n"
13             if object.laptop=="ON":
14                 behaviorString=behaviorString+"In "+ object.name + " Person1 watch on laptop "++"\n"
15         if object.name == "DayRoom":
16             behaviorString = behaviorString + "Person1 go in " + object.name + "\n"
17             if object.light == "ON":
18                 behaviorString = behaviorString + "In " + object.name + " light is " + object.light + "\n"
19             else:
20                 behaviorString = behaviorString + "In " + object.name + " light is " + object.light + "\n"
21             if object.tvMainRoom == "ON":
22                 behaviorString = behaviorString + "In " + object.name + " Person1 watch at tv in main room " + "\n"
23         if object.name == "BathRoom":
24             behaviorString = behaviorString + "Person1 go in " + object.name + "\n"
25             if object.light == "ON":
26                 behaviorString = behaviorString + "In " + object.name + " light is " + object.light + "\n"
27             else:
28                 behaviorString = behaviorString + "In " + object.name + " light is " + object.light + "\n"
29             if object.secondlight == "ON":
30                 behaviorString = behaviorString + "In " + object.name + " second light is " + object.light + "\n"
31             else:
32                 behaviorString = behaviorString + "In " + object.name + " second light is " + object.light + "\n"
33     print(behaviorString)
34
35 # object1=classRoom.BedRoom("a")
36 # object2=classRoom.BathRoom("b")
37 # object3=classRoom.BathRoom("c")
38 # list=[]
39 behaviorGenerator()
```

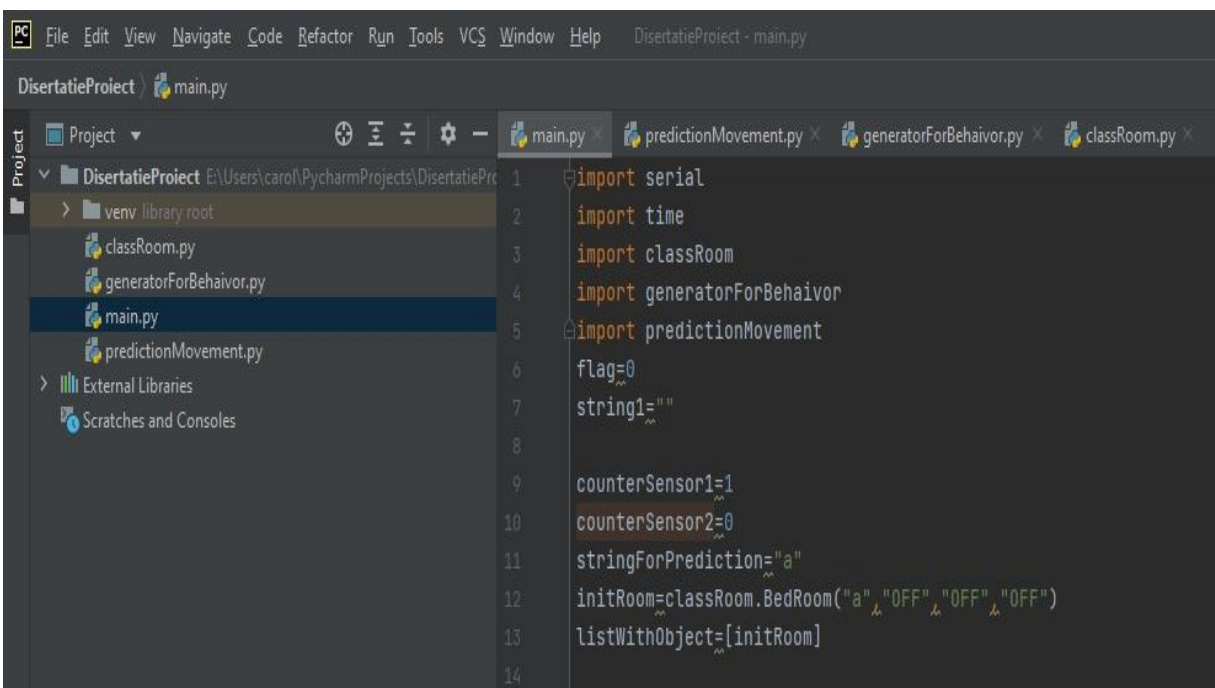
Figura 4.2: Modulul generatorForBehavior

În situația de față, putem oferi informații și de când până când a fost utilizat un anumit dispozitiv, adică unde a fost făcută o anumită activitate. În modul, vom străbate lista de obiecte prin instrucțiunea repetitivă „pentru” (for) în ordinea generată de către senzorii ultrasonici și, în funcție de proprietate name() a camerelor, vom verifica tipul lor (dacă este bathroom, bedroom sau dayroom), cu scopul de a putea inspecta activitățile care s-au realizat în acea cameră.

### Modulul 3. Main.py

Acest modul este creierul programului, adică din acest modul se rulează toate submodulele din aplicație. Pentru a putea citi datele transmise de senzorii ultrasonici conectați la placa arduino, a trebuit să importăm în python o librărie numită „serial”.

Aceasta ne permite să ne conectăm la IDE-ul de Arduino și să citim output-ul care este afișat în Serial Monitoring. Implementarea celor doi senzori ultrasonici este făcută în IDE-ul de Arduino, iar în modulul nostru se va crea stringul care va fi utilizat în algoritmul de predicție și o lista cu clasele, ce vor conține diferite activități care au fost făcute de către persoane.



```
1 import serial
2 import time
3 import classRoom
4 import generatorForBehavior
5 import predictionMovement
6 flag=0
7 string1=""
8
9 counterSensor1=1
10 counterSensor2=0
11 stringForPrediction="a"
12 initRoom=classRoom.BedRoom("a" "OFF" "OFF" "OFF")
13 listWithObject=[initRoom]
14
```

Figura 4.3: Importuri, librării și inițializare parametrii

Pentru a genera stringul folosit la predicția umană, fiecărei camere i se va alocă o literă. Așa că, pentru bedroom avem alocată litera a, pentru dayroom avem alocată litera b iar pentru bathroom, avem alocat litera c.

Considerăm că persoana va porni din camera corespondentă literei a, iar în momentul în care o persoana va trece din camera a în camera b, senzorii ultrasonici vor înregistra mișcarea. Camera b va fi trecerea comună între camera a și camera c. Este de menționat faptul că între camera a și camera c nu se poate trece direct.

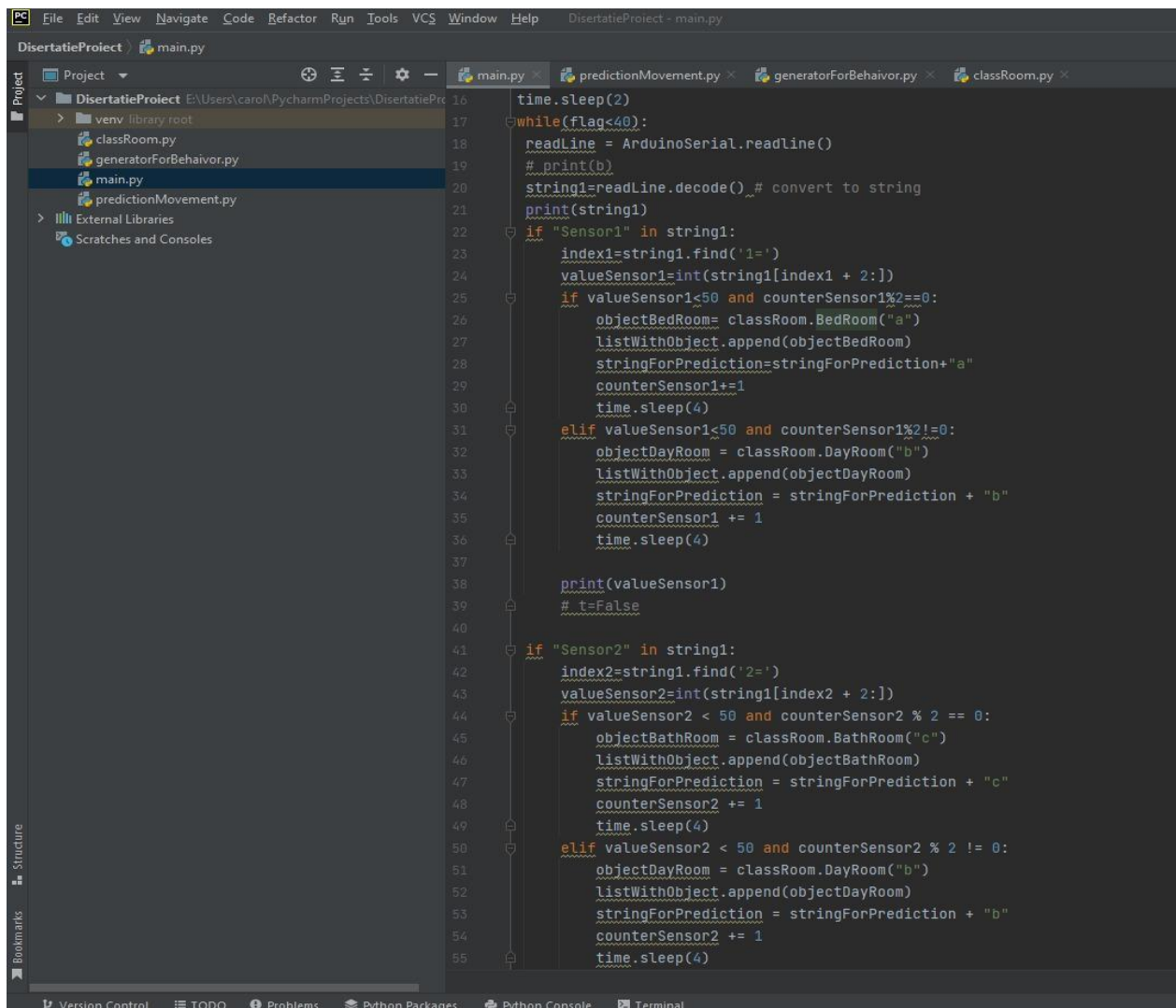


Figura 4.4: Instrucțiunea „cât timp”, generarea stringului și a listei de camere

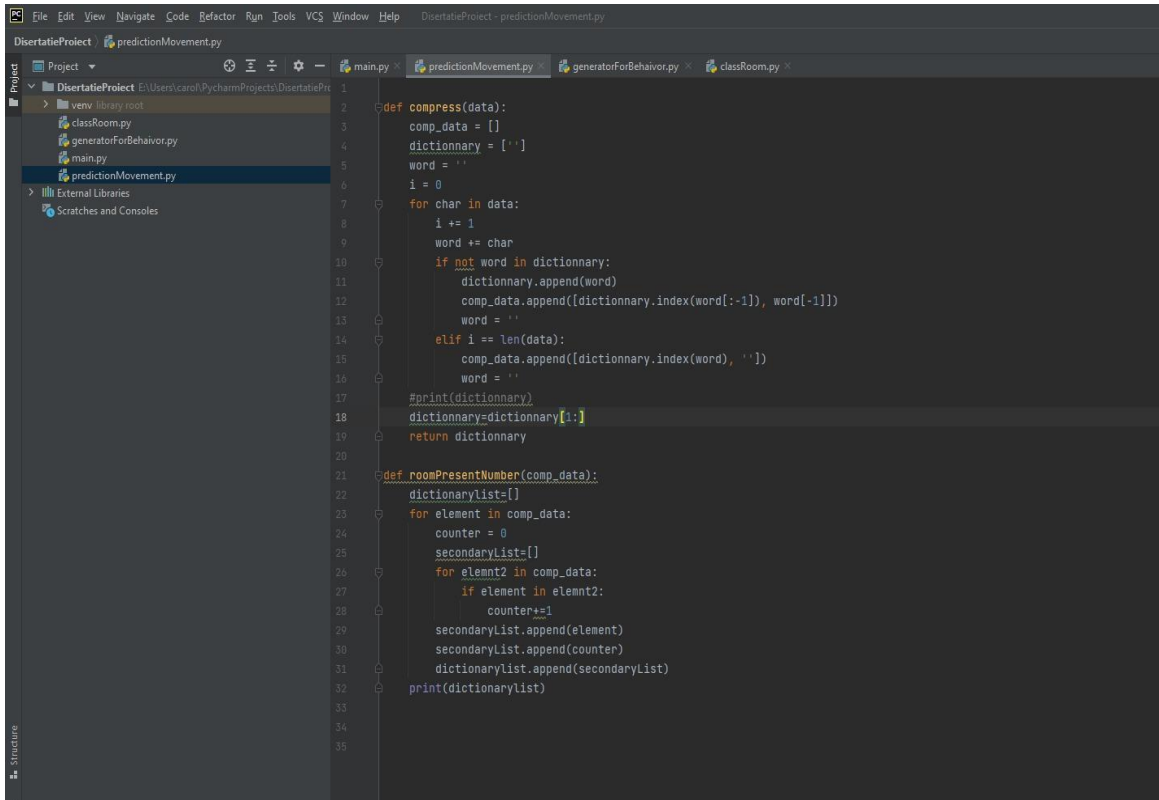
La rularea simulării, am folosit o instrucțiune repetitivă, respectiv „cât timp”. În cazul de față, pentru a vedea rezultatele, această instrucțiune este repetată de 40 de ori. Dar ea poate fi setată și în funcție de timp sau în funcție de când dorește utilizatorul să oprească aplicația.

La finalul acestei instrucțiuni „cat timp”, stringul generat va fi trimis către modulul predictionMovemment.py, iar lista cu obiectele, unde exista activitățile, va fi trimisă în generatorForBehaivor.

Se poate folosit biblioteca „time” pentru a avea o predicție mult mai bună și în funcție de timpul petrecut în diferitele camere.

## Modulul 4. predictionMovement.py

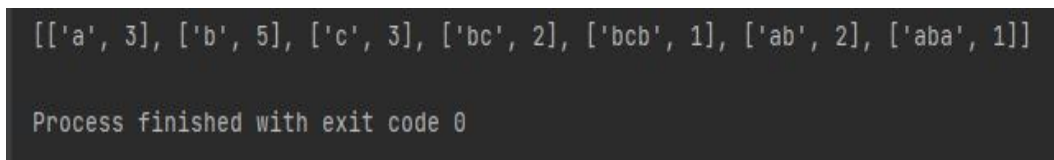
Mai multe detalii despre acest modul am prezentat în profunzime în capitolul unu. Acolo, așa cum se poate observa, am avut scris un pseudocod, pe care noi l-am implementat în limbajul de programare Python. În imaginea de mai jos vom vedea și rezultatul acestuia.



```
1 def compress(data):
2     comp_data = []
3     dictionary = {}
4     word = ''
5     i = 0
6     for char in data:
7         i += 1
8         word += char
9         if not word in dictionary:
10            dictionary[word] = i
11            comp_data.append([dictionary[word], word])
12            word = ''
13        elif i == len(data):
14            comp_data.append([dictionary[word], word])
15            word = ''
16    #print(dictionary)
17    dictionary = dictionary[1:]
18    return dictionary
19
20 def roomPresentNumber(comp_data):
21     dictionaryList = []
22     for element in comp_data:
23         counter = 0
24         secondaryList = []
25         for element2 in comp_data:
26             if element in element2:
27                 counter += 1
28             secondaryList.append(element)
29             secondaryList.append(counter)
30         dictionaryList.append(secondaryList)
31     print(dictionaryList)
```

Figura 4.5: Modulul predictionMovement.py

Tot în primul capitol (subcapitolul doi) am vorbit despre un dicționar, compus din litere, despre care spuneam la vremea respectivă că ne ajută la implementare. În Figura următoare, vom observa cum apare acel dicționar generat în aplicația noastră.

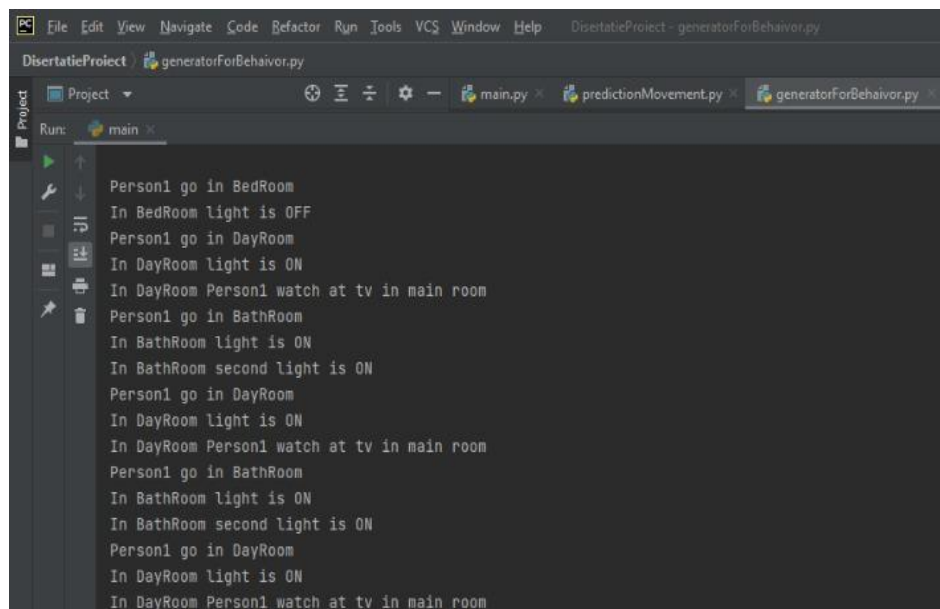


```
[[ 'a', 3], [ 'b', 5], [ 'c', 3], [ 'bc', 2], [ 'bcb', 1], [ 'ab', 2], [ 'aba', 1]]

Process finished with exit code 0
```

Figura 4.6: Dicționar generat în aplicație

Dacă urmărim toți pașii prezentați mai sus, sigur vom obține un program care rulează, fără să conțină erori. Mai jos, avem afișat și rezultatul acestuia.



```
DisertatieProiect - generatorForBehavior.py
DisertatieProiect generatorForBehavior.py
Project
Run: main
Person1 go in BedRoom
In BedRoom light is OFF
Person1 go in DayRoom
In DayRoom light is ON
In DayRoom Person1 watch at tv in main room
Person1 go in BathRoom
In BathRoom light is ON
In BathRoom second light is ON
Person1 go in DayRoom
In DayRoom light is ON
In DayRoom Person1 watch at tv in main room
Person1 go in BathRoom
In BathRoom light is ON
In BathRoom second light is ON
Person1 go in DayRoom
In DayRoom light is ON
In DayRoom Person1 watch at tv in main room
```

Figura 4.7: Rezultatul programului - Generare comportament uman

## 4.2. Recunoașterea emoțiilor umane folosind Inteligența Artificială

Scopul acestui proiect este acela de a putea recunoaște emoțiile unei persoane, adică dacă zâmbește, dacă este trist, dacă este nervos etc, folosind Deep Learning (reprezentat de rețelele neuronale convenționale), submodul al Inteligenței Artificiale. Pentru acest exemplu am selectat șapte tipuri de emoții: furie, dezgustare, frică, fericire, tristețe, surprindere și neutralitate (chipul fără nicio emo)

Modelul este antrenat pe setul de date FER-2013, care a fost publicat la Conferința Internațională pentru Învățare Mașină (ICML). Acest set de date este constituit din 35887 de imagini, în tonuri de gri, cu dimensiunea 48x48, reprezentând cele șapte emoții de mai sus.

### Mod de implementare

Procesul de implementare a fost realizat în limbajul de programare Python. Am importat mai multe librării, precum librăria TensorFlow și librăria Keras, folosite pentru Inteligența Artificială (AI), dar și librăria Open CV, utilizată în viziunea computerizată. Pentru a lămurii și acest ultim concept menționat, putem spune că viziunea computerizată este un proces prin care putem înțelege atât modul de stocare a fotografiilor și videoclipurilor, cât și tipul de manipulare și preluare a datelor din ele. Mai mult decât atât, o putem descrie ca fiind o bază sau chiar cea mai mare parte utilizată pentru inteligența artificială.



Computer Vision joacă un rol major în mașinile de conducere automată, robotică, precum și în aplicațiile de corectare a fotografiilor. Această implementare detectează în mod implicit toate expresiile faciale din fluxul camerei web. Cu un simplu CNN cu patru straturi, precizia textului a ajuns 63,2% în 50 de epoci.

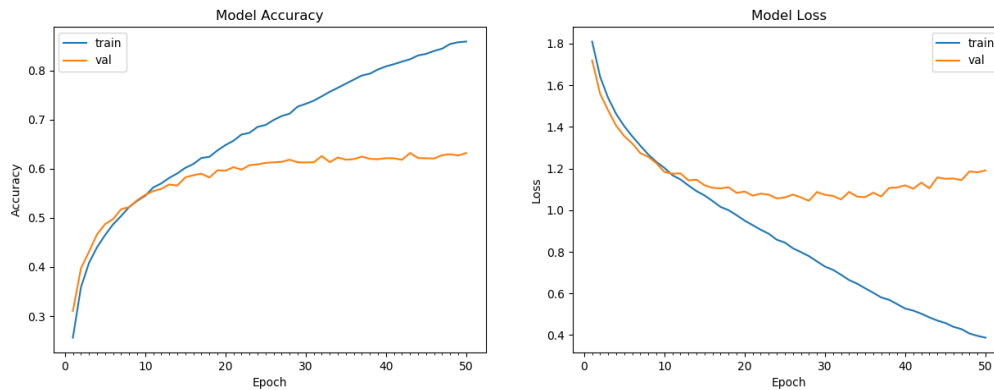


Figura 4.8: Figura grafică a preciziei

## Algoritm implementare

În primul rând, metoda cascadei implementată pe o față normală este utilizată pentru a detecta chipurile din fiecare cadru al fluxului webcam. Regiunea imaginii care conține fața este redimensionată la 48x48 și este transmisă ca intrare către CNN. Rețeaua emite astfel o listă de scoruri pentru cele șapte emoții despre care am vorbit mai sus. Emoția cu cel mai mare scor (scorul maxim) se va afișa pe ecran.

Ca și structură, aplicația este formată din folderul data, unde avem imaginile care reprezintă stările persoanelor implicate, fiind folosite ulterior pentru deep learning, haarcascade\_frontalface\_default.xml (utilizat în metoda de detectare a feței), modulul emotions.py (împărțit în două secțiuni) și model.h5, generat în urma antrenării, mai exact în urma folderului cu fotografii. Ceea ce mai putem menționa aici este faptul că în modulul emotions.py avem importurile librărilor care se pot vedea mai jos.

```
dataset_prepare.py x emotions.py x axis.py x
import numpy as np
import argparse
import matplotlib.pyplot as plt
import cv2
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D
from keras.optimizers import Adam
from keras.layers import MaxPooling2D
from keras.preprocessing.image import ImageDataGenerator
import os
```

Figura 4.9: Importare librării

Cel mai important în acest modul sunt cele două secțiuni în care ne putem alege felul în care să ruleze aplicația. Alegem „modul train”, dacă dorim întâi generarea fișierului model.h5 sau alegem „display”, în cazul în care avem deja fișierul model.h5. În situația în care vom rula în „modul train” programul va trece prin 50 de epoci de învățare al algoritmului și va genera graficul de acuratețe și pierderile ale modelului. O epocă în învățarea automată înseamnă o trecere completă a setului de date de antrenament prin algoritm. Acest număr de epoci este un hiperparametru important pentru algoritm. Specifică numărul de epoci sau treceri complete ale întregului set de date de antrenament, care trec prin procesul de antrenament sau de învățare al algoritmului. Implementarea codului în limbajul de programare Python se poate vedea în imaginea de mai jos.

```
# If you want to train the same model or try other models, go for this
if mode == "train":
    model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.0001, decay=1e-6), metrics=['accuracy'])
    model_info = model.fit_generator(
        train_generator,
        steps_per_epoch=num_train // batch_size,
        epochs=num_epoch,
        validation_data=validation_generator,
        validation_steps=num_val // batch_size)
    plot_model_history(model_info)
    model.save_weights('model.h5')
```

Figura 4.10: Cod metoda „train”

În modulul „display”, aplicația va porni camera și va putea recunoaște emoțiile de pe fata unei persoane. Cum se realizează implementarea în limbajul Python, se va observa în Figura 4.11.

```
# emotions will be displayed on your face from the webcam feed
elif mode == "display":
    model.load_weights('model.h5')

    # prevents openCL usage and unnecessary logging messages
    cv2ocl.setUseOpenCL(False)

    # dictionary which assigns each label an emotion (alphabetical order)
    emotion_dict = {0: "Angry", 1: "Disgusted", 2: "Fearful", 3: "Happy", 4: "Neutral", 5: "Sad", 6: "Surprised"}

    # start the webcam feed
    cap = cv2.VideoCapture(0)
    while True:
        # Find haar cascade to draw bounding box around face
        ret, frame = cap.read()
        if not ret:
            break
        facecasc = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = facecasc.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)

        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y-50), (x+w, y+h+10), (255, 0, 0), 2)
            roi_gray = gray[y:y+h, x:x+w]
            cropped_img = np.expand_dims(np.expand_dims(cv2.resize(roi_gray, (48, 48)), -1), 0)
            prediction = model.predict(cropped_img)
            maxindex = int(np.argmax(prediction))
            cv2.putText(frame, emotion_dict[maxindex], (x+20, y-60), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

        cv2.imshow('Video', cv2.resize(frame, (1000, 960), interpolation=cv2.INTER_CUBIC))
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()
```

Figura 4.11: Cod metoda „Display”

Mai jos, avem și câteva imagini cu rularea aplicației și outputul ei.

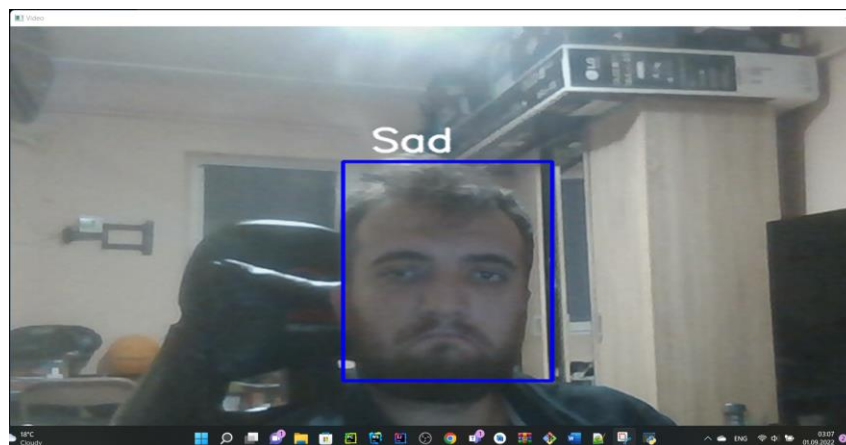


Figura 4.12: Supărat

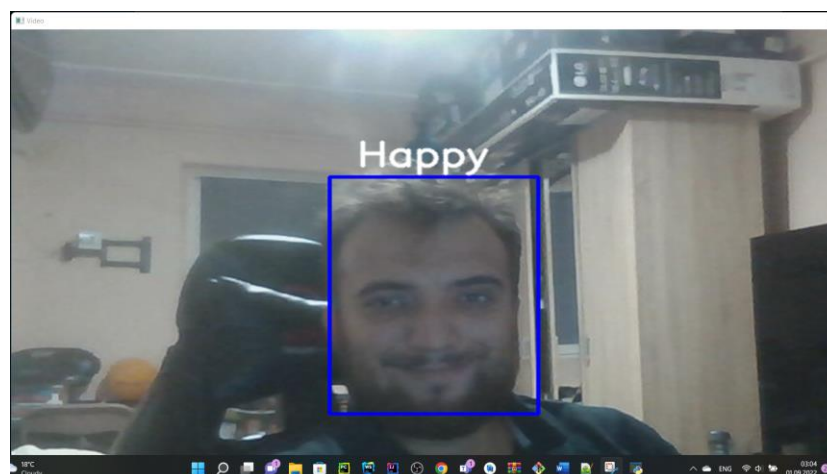


Figura 4.13: Fericit

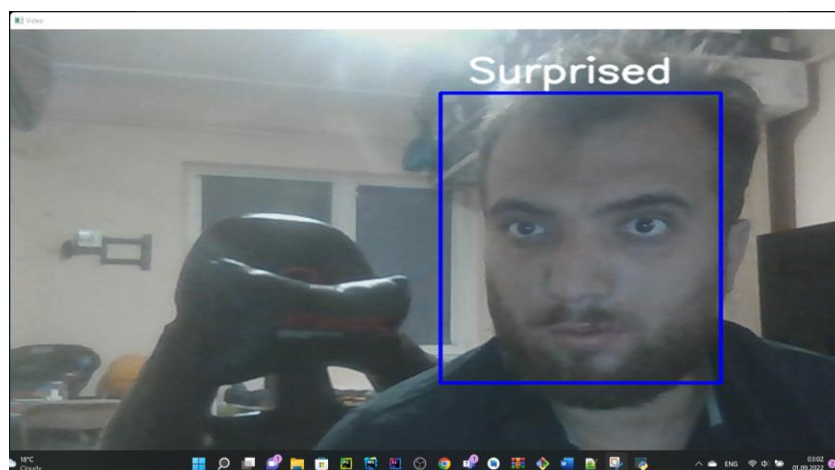


Figura 4.14: Mirat

### **4.3. Recunoașterea facială în casele inteligente**

Recunoașterea feței este o tehnică de identificare sau de verificare a feței din imaginile digitale sau din cadrul video. Un om poate identifica rapid fețele fără prea mult efort. Este o sarcină fără efort pentru noi, dar este o sarcină dificilă pentru un computer. Există diverse complexități, cum ar fi rezoluția scăzută, ocluzia, variațiile de iluminare etc. Acești factori afectează foarte mult acuratețea calculatorului de a recunoaște mai eficient fața. În primul rând, este necesar să înțelegem diferența dintre detectarea feței și recunoașterea feței.

#### **Detectarea feței**

Detectarea fețelor este în general considerată ca fiind găsirea fețelor (locație și dimensiune) într-o imagine și extragerea acestora pentru a fi utilizate de algoritmul de detectare a fețelor.

#### **Recunoașterea fețelor**

Algoritmul de recunoaștere a fețelor este utilizat pentru a găsi caracteristicile care sunt descrise în mod unic în imagine. Imaginea facială este deja extrasă, decupată, redimensionată și, de obicei, convertită în scala de griuri.

Există diverși algoritmi de detectare și recunoaștere a fețelor. Aici vom învăța despre detectarea feței folosind algoritmul în cascadă HAAR.

#### **4.3.1 Conceptul de bază al algoritmului de cascadă HAAR**

Cascada HAAR este o abordare de învățare automată în care o funcție în cascadă este antrenată pe baza unui număr mare de imagini pozitive și negative. Imaginile pozitive sunt acele imagini care conțin fețe, iar imaginile negative nu conțin fețe. În detectarea fețelor, caracteristicile imaginii sunt tratate ca informații numerice extrase din imagini care pot distinge o imagine de alta.

Aplicăm fiecare caracteristică a algoritmului pe toate imaginile de antrenament. Fiecărei imagini i se acordă o pondere egală la început. Se găsește cel mai bun prag care va clasifica fețele ca fiind pozitive sau negative. Pot exista erori și clasificări greșite. Selectăm caracteristicile cu o rată de eroare minimă, ceea ce înseamnă că acestea sunt caracteristicile care clasifică cel mai bine imaginile cu sau fără chipuri. Toate dimensiunile și locațiile posibile ale fiecărui nucleu sunt utilizate pentru a calcula multitudinea de caracteristici.

### 4.3.2 Detectia HAAR-Cascade în OpenCV

Open CV oferă atât formatorul, cât și detectorul. Putem antrena clasificatorul pentru orice obiect, cum ar fi mașinile, avioanele și clădirile, utilizând Open CV. Există două stări primare ale clasificatorului de imagini în cascadă: prima este antrenarea și cealaltă este detectarea. Open CV oferă două aplicații pentru a antrena clasificatorul în cascadă: `opencv_haartraining` și `opencv_traincascade`. Aceste două aplicații stochează clasificatorul în format de fișier diferit.

Pentru instruire, avem nevoie de un set de eșantioane. Există două tipuri de eșantioane:

Eșantion negativ: Se referă la imagini fără obiect.

Eșantioane pozitive: Este o imagine legată cu obiecte detectate.

Un set de eșantioane negative trebuie pregătit manual, în timp ce colecția de eșantioane pozitive este creată cu ajutorul utilitarului `opencv_createsamples`.

#### Eșantion negativ

Eșantioanele negative sunt prelevate din imagini arbitrare. Probele negative sunt adăugate într-un fișier text. Fiecare linie a fișierului conține un nume de fișier de imagine (în raport cu directorul fișierului de descriere) al eșantionului negativ. Acest fișier trebuie creat manual. Imaginile definite pot fi de dimensiuni diferite.

#### Eșantion pozitiv

Eșantioanele pozitive sunt create de utilitarul `opencv_createsamples`. Aceste eșantioane pot fi create dintr-o singură imagine cu un obiect sau dintr-o colecție anterioară. Este important de reținut că avem nevoie de un set de date mare de eșantioane pozitive înainte de a-l da utilitarului menționat, deoarece acesta aplică doar transformarea de perspectivă.

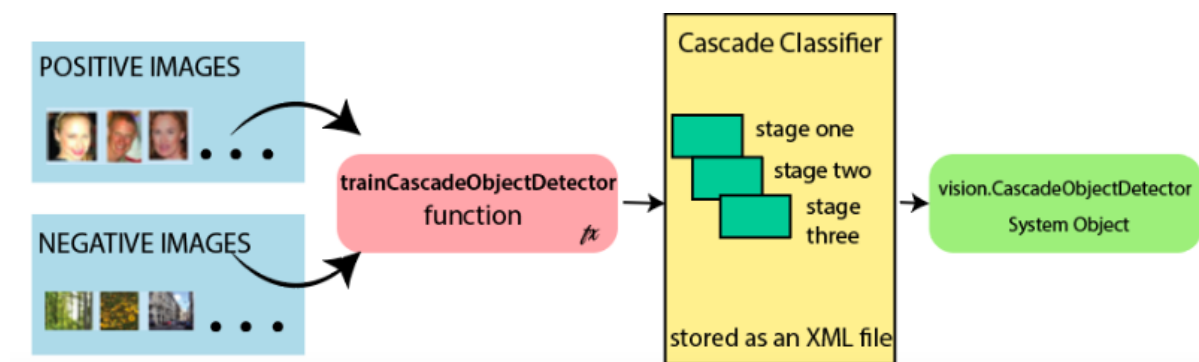


Figura 4.15: Clasificarea Cascadă

Aici vom discuta despre detecție. Open CV conține deja diverși clasificatori pre-antrenați pentru față, ochi, zâmbet etc. Aceste fișiere XML sunt stocate în dosarul `opencv/data/haarcascades/`. Să înțelegem următorii pași:

**Pasul – 1:** în primul rând, trebuie să încărcăm clasificatorii XML necesari și să încărcăm imaginile de intrare (sau video) în modul gri.

**Pasul – 2:** după ce am convertit imaginea în scala de gri, putem face manipularea imaginii, unde imaginea poate fi redimensionată, decupată, estompată și, dacă este necesar, mai clară. Următorul pas este segmentarea imaginii; identificăm obiectele multiple dintr-o singură imagine, astfel încât clasificatorul să detecteze rapid obiectele și fețele din imagine.

**Pasul – 3:** algoritmul de caracteristici haar-Like este utilizat pentru a găsi locația fețelor umane în cadru sau în imagine. Toate fețele umane au unele proprietăți universale comune ale fețelor, cum ar fi regiunea ochilor este mai întunecată decât pixelii vecinilor săi, iar regiunea nasului este mai luminoasă decât regiunea ochilor.

**Pasul – 4:** în acest pas, extragem caracteristicile din imagine, cu ajutorul detecției marginilor, al detecției liniilor și al detecției centrului. Apoi, furnizăm coordonatele x, y, w, h, care formează o cutie dreptunghiulară în imagine pentru a arăta locația feței. Acesta poate face o cutie dreptunghiulară în zona dorită în care detectează fața.

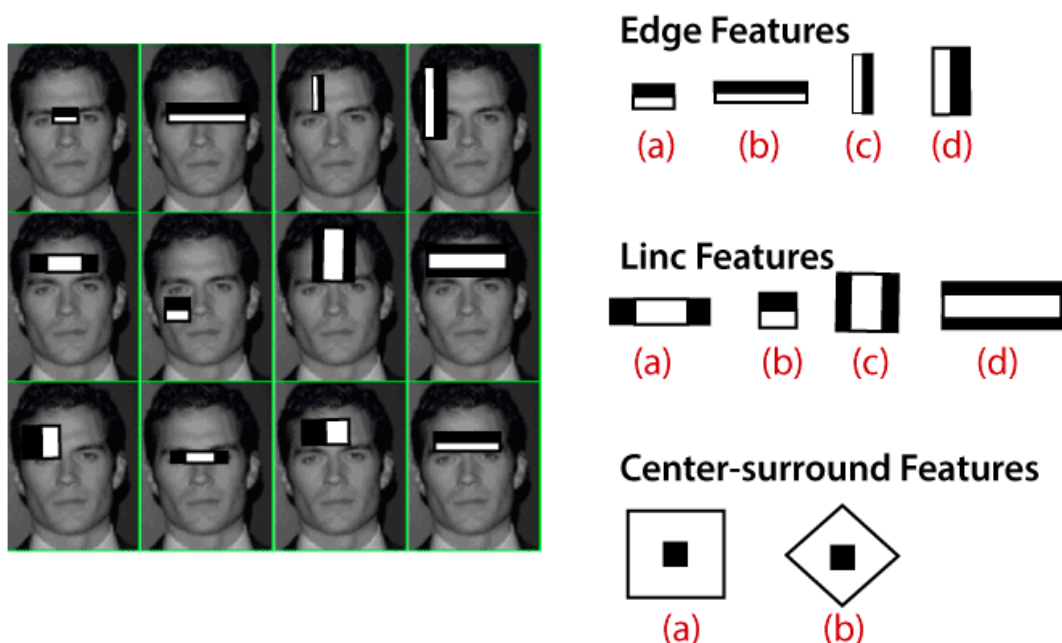


Figura 4.16: Tipuri de detectare

### 4.3.3. Recunoașterea feței folosind OpenCV

Recunoașterea fețelor este o sarcină simplă pentru oameni. Recunoașterea cu succes a fețelor tinde spre recunoașterea eficientă a trăsăturilor interioare (ochi, nas, gură) sau a trăsăturilor exterioare (cap, față, linia părului). În acest caz, întrebarea este cum codifică creierul uman aceste caracteristici?

David Hubel și Torsten Wiesel arată că creierul nostru are celule nervoase specializate care răspund la caracteristici locale unice ale scenei, cum ar fi liniile, unghiul marginilor sau mișcarea.

Creierul nostru combină diferitele surse de informații în modele utile; nu vedem imaginea ca fiind dispersată. Dacă definim recunoașterea fețelor într-un cuvânt simplu, "recunoașterea automată a fețelor constă în extragerea acelor caracteristici semnificative dintr-o imagine și punerea lor într-o reprezentare utilă, apoi efectuarea unei clasificări pe baza acestora".

Ideea de bază a recunoașterii fețelor se bazează pe caracteristicile geometrice ale unei fețe. Aceasta este abordarea fezabilă și cea mai intuitivă pentru recunoașterea fețelor. Primul sistem automatizat de recunoaștere a fețelor a fost descris în funcție de poziția ochilor, urechilor, nasului. Aceste puncte de poziționare se numesc vector de caracteristici (distanța dintre puncte).

Recunoașterea feței se realizează prin calcularea distanței euclidiene dintre vectorii de caracteristici ai unei imagini de probă și ai unei imagini de referință. Prin natura sa, această metodă este eficientă în cazul schimbărilor de iluminare, dar are un dezavantaj considerabil. Înregistrarea corectă a creatorului este foarte dificilă.

Sistemul de recunoaștere a feței poate funcționa în principiu în două moduri:

**Autentificarea sau verificarea unei imagini faciale:** compară imaginea facială de intrare cu imaginea facială a utilizatorului, care este necesară pentru autentificare. Este o comparație 1x1.

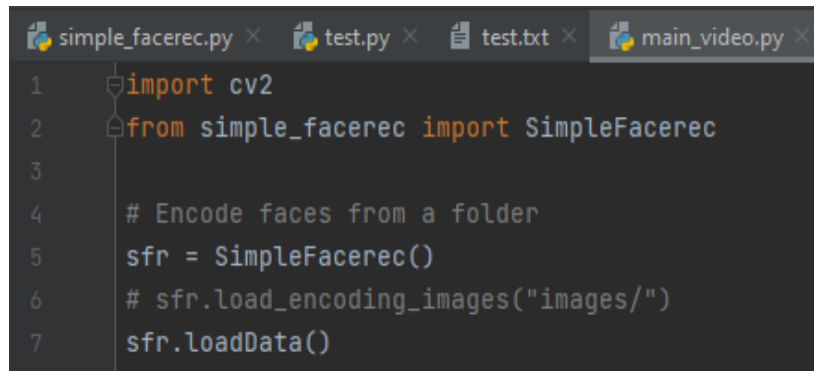
**Identificare sau recunoaștere facial:** compară imaginile faciale de intrare dintr-un set de date pentru a găsi utilizatorul care se potrivește cu acea față de intrare. Este o comparație 1xN.

### Implementarea în Python

Aplicațiile este formată din două module: `main_video.py` și `simple.facerec.py`. Modulul de `main_video.py` îl putem considera modulul principal, de unde programul rulează, iar în modulul `simple.facerec.py` se află funcțiile pentru rularea modulului principal, cel menționat anterior. În

modulul de bază, în prima secțiune a programului se fac importurile librărilor, adică librăria Open CV și modulul simple.facerec.py, necesare rulării aplicației.

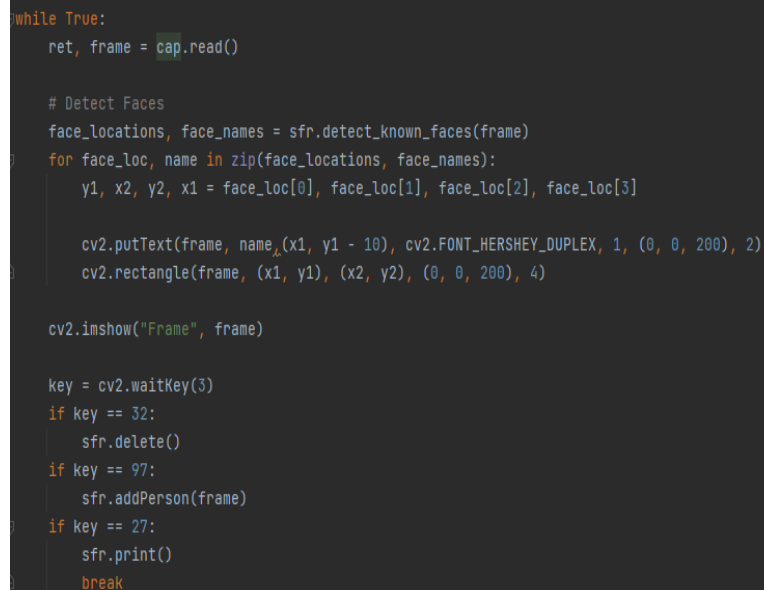
Pe lângă încărcările de librării avem și încărcarea fețelor deja cunoscute din baza de date a recunoașterii faciale. Adăugarea în baza de date a feței unei persoane noi se poate adăuga fie din imagini, fie în timp real, în timp ce camera este pornită. Mai jos se poate vedea și implementarea acestor noțiuni prezentate în cod.



```
1 import cv2
2 from simple_facerec import SimpleFacerec
3
4 # Encode faces from a folder
5 sfr = SimpleFacerec()
6 # sfr.load_encoding_images("images/")
7 sfr.loadData()
```

Figura 4.17: Importate librării și încărcare bază de date

Următoarea secțiune și, totodată cea mai importantă, este rularea programului în sine din care se apelează funcții din modulul simple.facerec.py. În Figurade mai jos se va putea observa implementarea ei în limbajul de programare Python.



```
while True:
    ret, frame = cap.read()

    # Detect Faces
    face_locations, face_names = sfr.detect_known_faces(frame)
    for face_loc, name in zip(face_locations, face_names):
        y1, x2, y2, x1 = face_loc[0], face_loc[1], face_loc[2], face_loc[3]

        cv2.putText(frame, name, (x1, y1 - 10), cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 200), 2)
        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 200), 4)

    cv2.imshow("Frame", frame)

    key = cv2.waitKey(3)
    if key == 32:
        sfr.delete()
    if key == 97:
        sfr.addPerson(frame)
    if key == 27:
        sfr.print()
        break
```

Figura 4.18: Main program



Cea mai esențială funcție apelată în main este detect\_known\_faces. Aceasta va compara fețele din imagine cu cele din baza de date. În cazul în care o față nu este cunoscută, va apărea ca unknow. Dacă fața este recunoscută, va apărea numele modelului cunoscut.

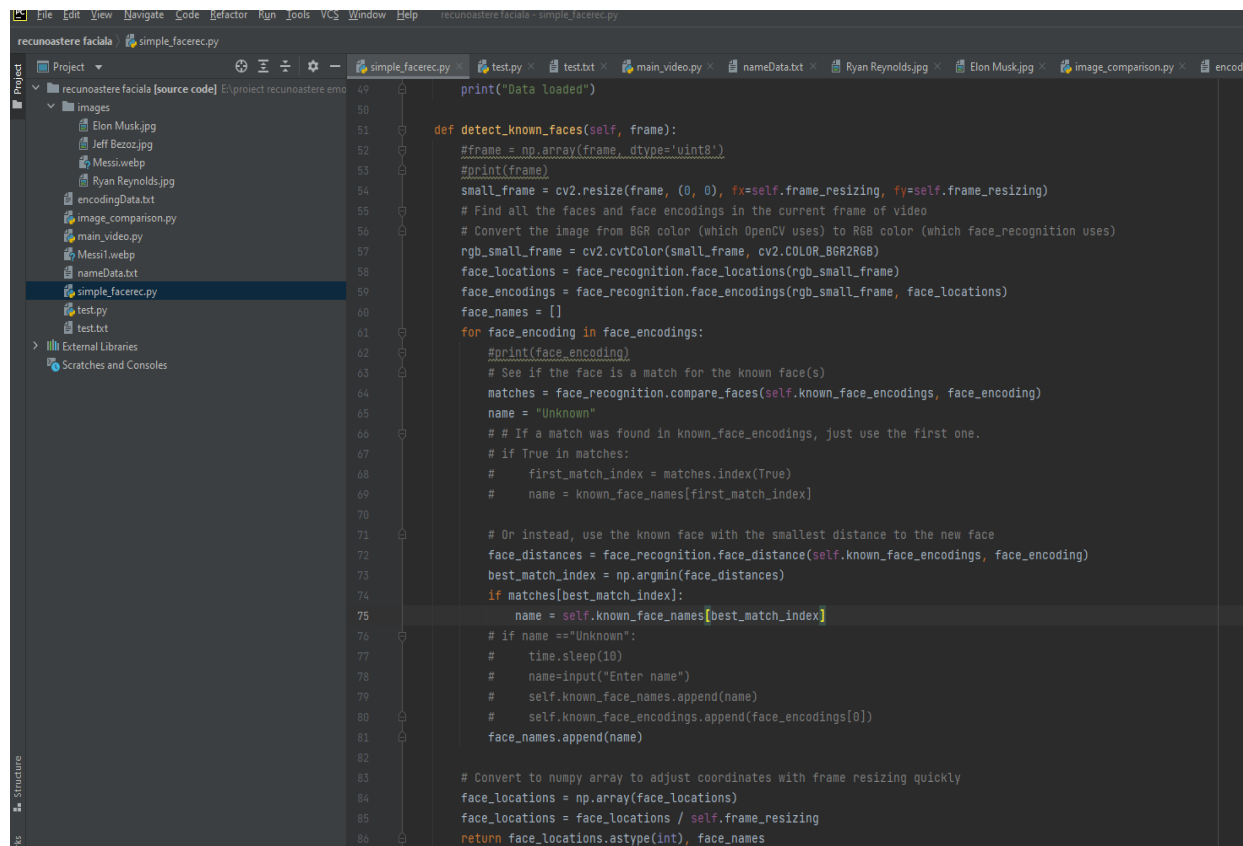


Figura 4.19: Funcția de comparare a fețelor cunoscute

În modulul main, după cum se vede, se pot apela și funcțiile addPerson și delete, care sunt folosite pentru adăugarea și ștergerea unei persoane în timp real, din baza de date. Funcțiile se apelează doar în momentul în care utilizatorul apasă pe tastatură litera corespunzătoare în Ascii a numărului 32 - pentru ștergere și 97 - pentru adăugare.

```

def addPerson(self, frame):
    small_frame = cv2.resize(frame, (0, 0), fx=self.frame_resizing, fy=self.frame_resizing)
    # Find all the faces and face encodings in the current frame of video
    # Convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)
    rgb_small_frame = cv2.cvtColor(small_frame, cv2.COLOR_BGR2RGB)
    face_locations = face_recognition.face_locations(rgb_small_frame)
    face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)
    name = input("Enter name")
    self.known_face_names.append(name)
    self.known_face_encodings.append(face_encodings[0])

def delete(self):
    name=input("input name to delete")
    for index, pers in enumerate(self.known_face_names):
        if name in pers:
            self.known_face_names.pop(index)
            self.known_face_encodings.pop(index)
            print(self.known_face_names)
            break

```

Figura 4.20: Funcția de adăugare și de ștergere a unei persoane

În momentul în care aplicația se va rula, rezultatul ei va arăta ca în imaginea de mai jos.

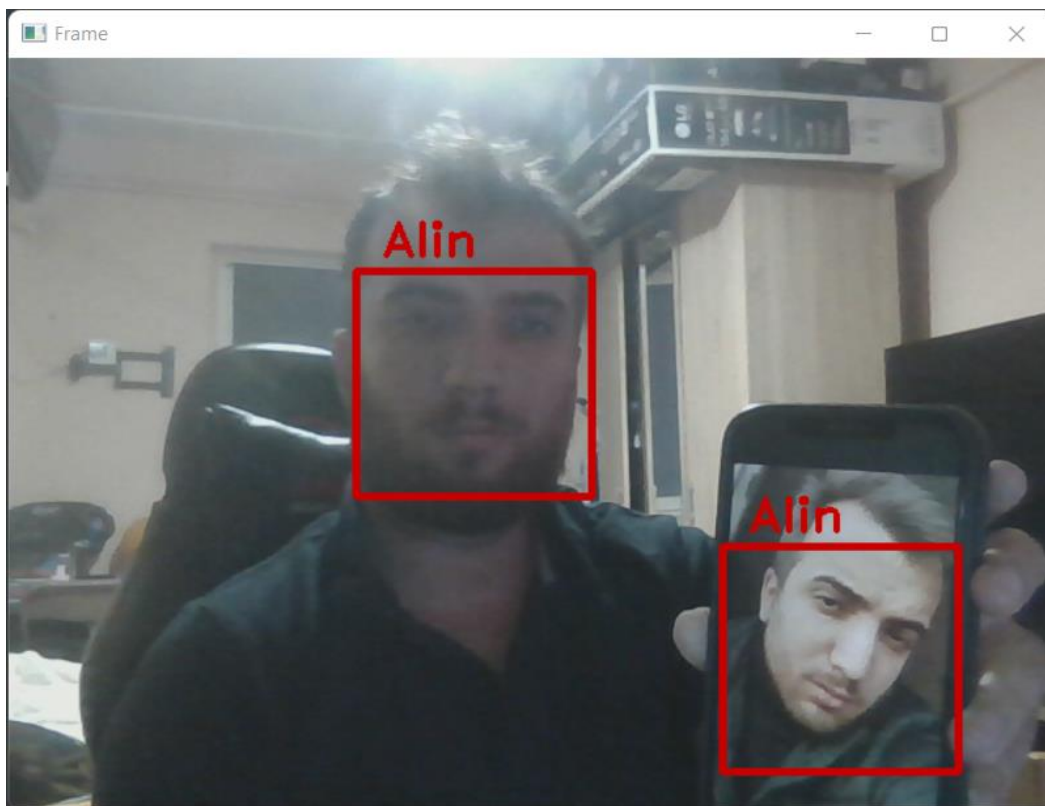


Figura 4.21: Recunoaștere facială

## Arhitectura sistemului

Arhitectura va fi compusă din 2 nivele:

- High-level: unde va acționa algoritmul de predicție, în funcție de camerele casei inteligente;
- Low-level: care va acționa pe camera în care este utilizatorul prezent. Apoi, aplicația va prelua interacțiunea cu dispozitivele din acea încăpere, emoțiile utilizatorului, dar și protagonistul aflat în plin proces de recunoaștere facială, în cazul în care mai multe persoane locuiesc în casă.

### Arhitectura high-level

Considerăm exemplul implementat anterior, unde casa are trei camere, cu literele a,b,c corespondente fiecăruia. Schema reprezentativă pentru aceasta aplicație se poate observa mai jos.

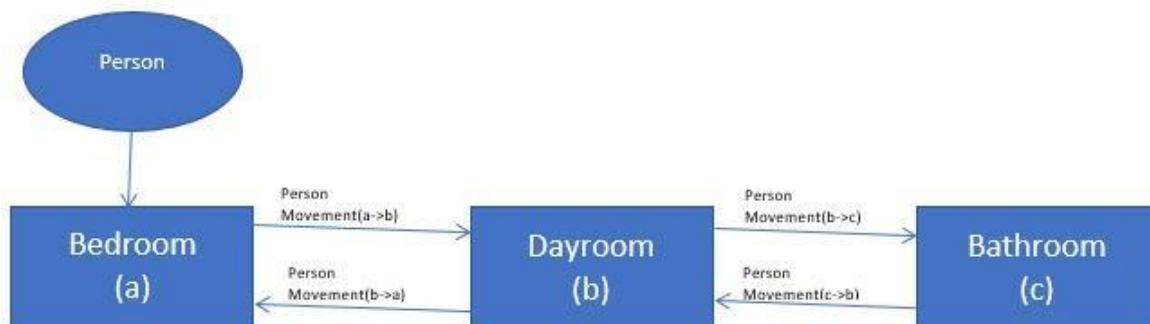


Figura 4.22. Arhitectura high-level

Și ca să înțelegem mai bine întregul proces, vom da un exemplu: spunem că Persoana 1 este deja prezentă în camera a. Algoritmul de predicție ne va memora fiecare mutare a utilizatorului dintr-o cameră în alta, printr-un șir de caractere (string de litere). În cazul nostru, șirul va arăta în felul următor: „abcbcbab”.

Ulterior, algoritmul va prelucra stringul și ne va genera un dicționar, care ne va ajuta să știm cât de des utilizatorul este atât într-o anumită cameră, cât și în celelalte camere, perioada în care acesta a trecut dintr-o cameră în alta, dar și timpul pe care l-a petrecut în acel loc.

## Arhitectura low-level

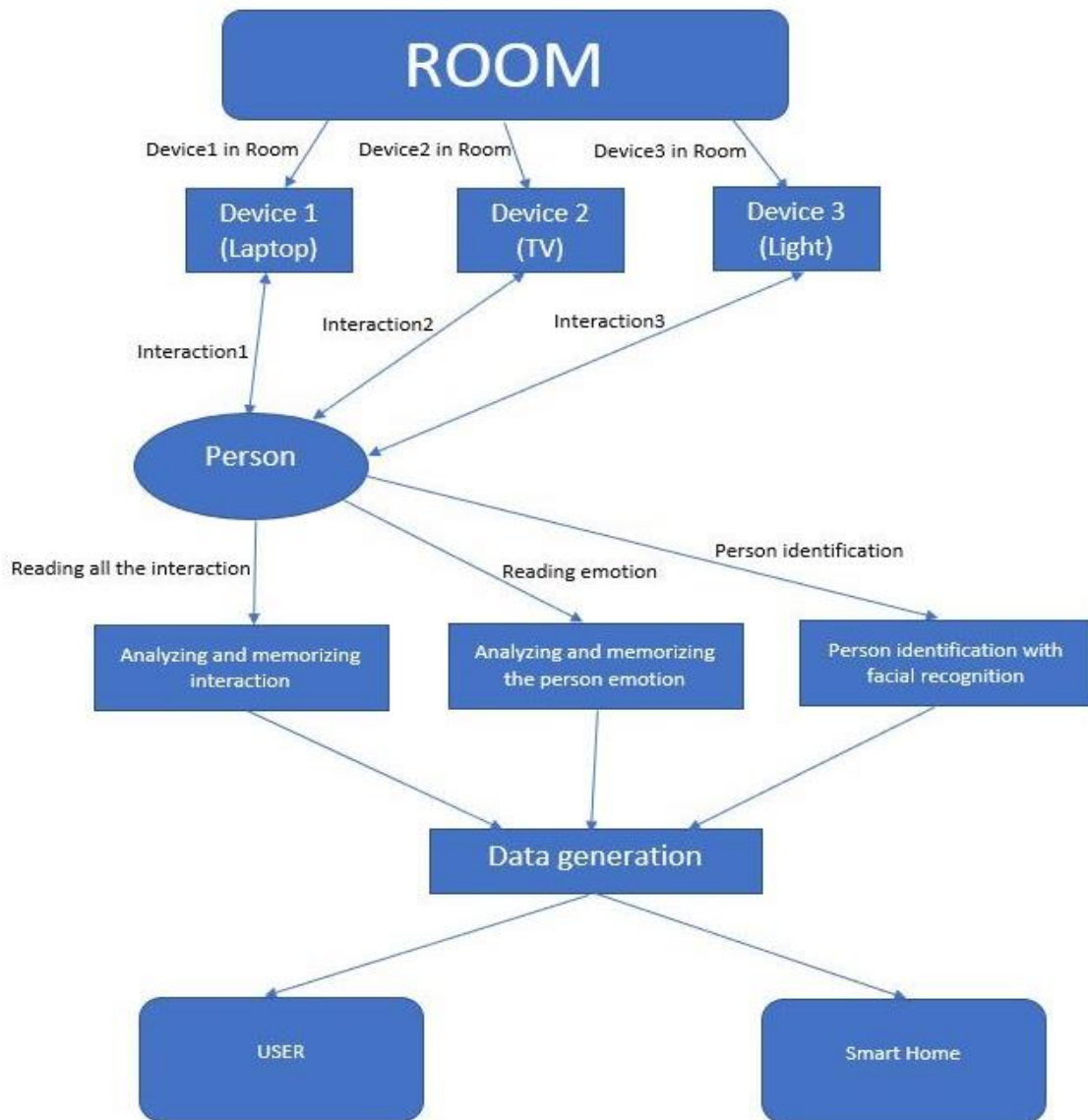


Figura 4.23. Arhitectura low-level

Aplicația la nivelul low-level, după cum se observă în schemă, va conține camerele cu dispozitivele specifice acestora și va prelucra atât interacțiunea persoanei cu acele obiecte, cât și emoțiile utilizatorului identificat de recunoașterea facială. Aceste date vor fi memorate de aplicație, apoi vor putea fi generate după o anumită perioadă de timp aleasă de utilizator.

Datele generate de aplicație pot fi folosite de utilizator și de casa inteligentă, iar scopul lor este de a ne ajuta atât în câștigarea timpului în urma prelucrării datelor persoanei cu dispozitivele, cât și în oferirea diferitelor activități, în cazul în care starea noastră emoțională nu este una bună.

Lista de activități va proveni din datele analizate de aplicație, în urma emoțiilor pozitive.

## CONCLUZII

Având în vedere implementările de la capitolul patru a algoritmului de predicție, generare a comportamentului, citirea emoțiilor, cât și recunoașterea facială, putem spune că acestea ne pot ajuta în foarte multe moduri într-o casă inteligentă. Și, mai mult decât atât, totul implementat într-un singur limbaj de programare.

Pentru a putea înțelege în ce privințe acestea ne pot ajuta, o să enumerăm câteva și vom explica cum pot realiza asta. Algoritmul de predicție ne poate învăța obiceiurile noastre într-o casă. Combinat cu Machine Learning, care este submodul al Inteligenței Artificiale, ne va oferi timp în plus prin automatizarea diferitelor activități repetitive, precum pornirea aparatului de cafea la ora la care ne trezim.

Generarea comportamentului nostru ne ajută în a vedea ce activități facem. Prin aceste activități, vom vedea dacă avem un stil de viață sănătos, putem face optimizări de timp, dar și economie de energie.

Recunoașterea facială și recunoașterea emoțiilor ne sunt de folos atât în privința securității, cât și în recunoașterea diverselor activități care ne plac sau ne displac.

Drept urmare, prin intermediul acestei teze de cercetare, am reușit să evidențiez faptul că, odată cu trecerea timpului, tehnologia a cunoscut o progresie constantă iar sistemul implementat în paginile de mai sus ne poate oferi o perspectivă a comportamentului uman într-o casă inteligentă, dar și soluții de îmbunătățire a acestuia, creând totodată și un mediu ambiental inteligent.

## BIBLIOGRAFIE

### Surse electronice

- \*\*\* <https://www.javatpoint.com/face-recognition-and-face-detection-using-opencv?fbclid=IwAR2ru0WLpDoKILs6QVB9otvrhmxkC2YbVgw8HgGdDNCQOvDdNXSKEtIS33g> (accesare august 2022);
- \*\*\* [http://www.stringology.org/DataCompression/lz78/index\\_en.html](http://www.stringology.org/DataCompression/lz78/index_en.html) (accesare august 2022);
- \*\*\* <https://cs.stanford.edu/people/eroberts/courses/soco/projects/data-compression/lossless/lz78/concept.htm> (accesare august 2022);
- \*\*\* <https://medium.com/swlh/how-data-compression-works-exploring-lz78-e97e539138> (accesare august 2022);
- \*\*\* <https://www.sciencedirect.com/science/article/pii/S002200007280014X> (accesare august 2022);
- \*\*\* <https://cleste.ro/senzor-ultrasonic-hc-sr04.html> (accesare august 2022);
- \*\*\* [https://cleste.ro/arduino/senzori.html?page=9&utm\\_medium=GoogleAds&utm\\_campaign=&utm\\_source=&gclid=Cj0KCQjwjbyYBhCdARIsAArC6LIKd5YoIHmK1j4CsbAXSw1XE6zmqKMv-xu7ygKKWk-VxTdzeGvcEGMaAm3XEALw\\_wcB](https://cleste.ro/arduino/senzori.html?page=9&utm_medium=GoogleAds&utm_campaign=&utm_source=&gclid=Cj0KCQjwjbyYBhCdARIsAArC6LIKd5YoIHmK1j4CsbAXSw1XE6zmqKMv-xu7ygKKWk-VxTdzeGvcEGMaAm3XEALw_wcB) (accesare august 2022);
- \*\*\* <https://lastminuteengineers.com/433mhz-rf-wireless-arduino-tutorial/> (accesare august 2022);
- \*\*\* <https://www.adafruit.com/product/386> (accesare august 2022);
- \*\*\* <https://www.electronicwings.com/sensors-modules/adxl335-accelerometer-module> (accesare august 2022);
- \*\*\* <https://randomnerdtutorials.com/rf-433mhz-transmitter-receiver-module-with-arduino/> (accesare august 2022);
- \*\*\* <https://lastminuteengineers.com/pir-sensor-arduino-tutorial/> (accesare august 2022);
- \*\*\* <https://www.optimusdigital.ro/ro/senzori-senzori-pir/106-modul-senzor-pir-hc-sr501.html> (accesare august 2022).
- \*\*\* <https://components101.com/modules/5v-mb102-breadboard-power-supply-module> (accesare august 2022).
- \*\*\* <https://www.electroduino.com/ir-infrared-flame-sensor-module/> (accesare august 2022).
- \*\*\* <https://www.teachmemicro.com/lm393-ir-module-motor-speed-sensor/> (accesare august 2022).
- \*\*\* <https://eepower.com/resistor-guide/resistor-types/photo-resistor/> (accesare august 2022).
- \*\*\* <https://www.seeedstudio.com/blog/2019/12/30/what-is-barometric-pressure-sensor-and-arduino-guide-to-get-started/> (accesare august 2022)
- \*\*\* <https://www.python.org/doc/essays/foreword/> (accesare august 2022).
- \*\*\* <https://www.unixmen.com/guido-van-rossum-python-creator/> (accesare august 2022).
- \*\*\* [https://www.w3schools.com/python/python\\_intro.asp](https://www.w3schools.com/python/python_intro.asp) (accesat august 2022).
- \*\*\* <https://www.activestate.com/blog/how-to-automate-your-home-with-python/> (accesare august 2022).
- \*\*\* <https://hackernoon.com/top-8-python-libraries-for-machine-learning-and-artificial-intelligence-y08id3031> (accesat august 2022).
- \*\*\* [https://www.academia.edu/44783049/Internet of Things and Smart Home Security](https://www.academia.edu/44783049/Internet_of_Things_and_Smart_Home_Security) (accesat august 2022)/

## Surse fizice

- [1] D. N. Monekosso, P. Remagnino, and Y. Kuno, *Intelligent Environments: Methods, Algorithms and Applications*, Springer, 2009, ch. 1, pp. 1-11.
- [2] P. Gorniak and D. Poole, "Predicting future user actions by observing unmodified applications," *AAAI-00 Proceedings*, 2000.
- [3] M. Hartmann and D. Schreiber, "Prediction algorithms for user actions," *LWA'07*, pp. 349-354, 2007.
- [4] S. K. Das, D. J. Cook, A. Bhattacharya, E. O. H. Iii, and T.-Y. Lin, "The role of prediction algorithms in the mavhome smart home architecture," *IEEE Wireless Communications*, December 2002.
- [5] K. Gopalratnam and D. J. Cook, "Active Lezi: an incremental parsing algortihm for sequential prediction," *IEEE Intelligent Systems*, vol. 22, no. 1, pp. 52-58, 2007.
- [6] S. Laxman, P. S. Shastri, and K. P. Unnikrishnan, "Fast algorithms for frequent episode discovery in event sequences," presented at the Third Workshop on Mining Temporal and Sequential Data Seattle, August 2004.
- [7] Dixit, A., Naik, A., *Use of Prediction Algorithms in Smart Homes*, Person, 2001, pp 157-162.



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

Subsemnatul RAN CAROL ALIN

legitimat cu C1 seria T2 nr. 522030

CNP 1940618250015

autorul lucrării STUDIUL ȘI ÎMBUNĂȚIREA COMPORTAMENTULUI  
UMAN ÎNTR-O CASĂ INTELEGENȚĂ

elaborată în vederea susținerii examenului de finalizare a studiilor de DISERTAȚIE organizat de către Facultatea \_\_\_\_\_ din cadrul Universității

Politehnica Timișoara, sesiunea SEPTEMBRIE a anului universitar 2022, coordonator CONF. UNIV. DR. ING. HUGURĂȚĂ MOCOȘAN luând în

considerare conținutul art. 34 din Regulamentul privind organizarea și desfășurarea examenelor de licență/diplomă și disertație, aprobat prin HS nr. 109/14.05.2020 și cunoscând faptul că în cazul constatării ulterioare a unor declarații false, voi suporta sancțiunea administrativă prevăzută de art. 146 din Legea nr. 1/2011 – legea educației naționale și anume anularea diplomei de studii, declar pe proprie răspundere, că:

- această lucrare este rezultatul propriei activități intelectuale,
- lucrarea nu conține texte, date sau elemente de grafică din alte lucrări sau din alte surse fără ca acestea să nu fie citate, inclusiv situația în care sursa o reprezintă o altă lucrare/alte lucrări ale subsemnatului.
- sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.
- această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență/diplomă/disertație.

Timișoara,

Data

05.09.2022

Semnătura

[Semnătură]

\* Declarația se completează „de mână” și se inserează în lucrarea de finalizare a studiilor, la sfârșitul acesteia, ca parte integrantă.