 **UNIVERSITATEA “POLITEHNICA” DIN TIMISOARA**

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**DEPARTAMENTUL AUTOMATICĂ ŞI INFORMATICĂ APLICATĂ**

**HIDE&SEEK**

**PROIECT SINCRETIC I**

**Iancu Mădălina**

**Călușeri Abel**

**Chițon Roberta**

**AUTORI:**

**Coordonatori:**

**Alexa Liviu-Aniel**

**Voisan Emil**

**Anul III AIA - an universitar 2024 / 2025**

# CUPRINS

1. Introducere..................................................................................3

2. Prezentarea temei..........................................................................4

3. Tehnologii utilizate........................................................................5

4. Ghidul Programatorului...................................................................7

5. Ghidul Utilizatorului......................................................................15

6. Testare și punere în funcțiune........................................................18

7. Prezentarea firmei.......................................................................21

8. Concluzii...................................................................................27

9. Bibliografie..............................................................................28

# Introducere

# Prezentarea domeniului: Roboții mobili și conducerea la distanță a acestora

Roboții mobili autonomi (AMR) reprezintă o inovație semnificativă în domeniul logisticii și producției, permițând automatizarea transportului de materiale și optimizarea fluxurilor de lucru. Acești roboți sunt capabili să navigheze independent, să detecteze și să evite obstacolele, precum și să își adapteze traseul în timp real, datorită sistemelor avansate de senzori și algoritmilor de navigație.

În ceea ce privește conducerea la distanță, aceasta implică controlul roboților mobili de către un operator aflat într-o locație diferită, folosind tehnologii de comunicație precum internetul și rețelele wireless. Teleoperarea este esențială în aplicații unde autonomia completă nu este posibilă sau dorită, permițând intervenția umană în timp real pentru sarcini complexe sau în medii imprevizibile.

**Aplicații ale roboților mobili și conducerii la distanță**

1. **Logistică și depozite**: Roboții mobili transportă bunuri și optimizează procesele de livrare
2. **Explorare**: Roboții mobili sunt utilizați în explorări planetare (de exemplu, roverele Marte) sau în medii periculoase pentru oameni.
3. **Militar**: Roboții mobili sunt folosiți pentru misiuni de recunoaștere și neutralizarea amenințărilor.

**Tehnologii implicate**

1. **Senzori**: Utilizați pentru percepția mediului (LIDAR, camere, senzori ultrasonici).
2. **Algoritmi de navigație**: Planificarea traseului, evitarea obstacolelor și localizare (SLAM - Simultaneous Localization and Mapping).
3. **Comunicare la distanță**: Protocoluri de rețea (Wi-Fi, Bluetooth, 5G) pentru control în timp real.
4. **Inteligență artificială**: Algoritmi de învățare pentru adaptare la medii dinamice.
5. **Platforme software**: ROS (Robot Operating System) pentru integrarea hardware și software.

Proiectul sincretic 'Hide and Seek' reprezintă o demonstrație practică a modului în care tehnologiile moderne pot fi utilizate pentru a crea roboți autonomi capabili să execute sarcini complexe, precum identificarea persoanelor și navigarea către acestea.   
  
Proiectul folosește platforma TurtleBot3, un robot mobil accesibil și flexibil, utilizat pe scară largă pentru cercetare și educație. Mediul de dezvoltare ROS 2 facilitează comunicarea între componentele robotului, iar bibliotecile Python permit implementarea algoritmilor de recunoaștere facială și de navigație autonomă.

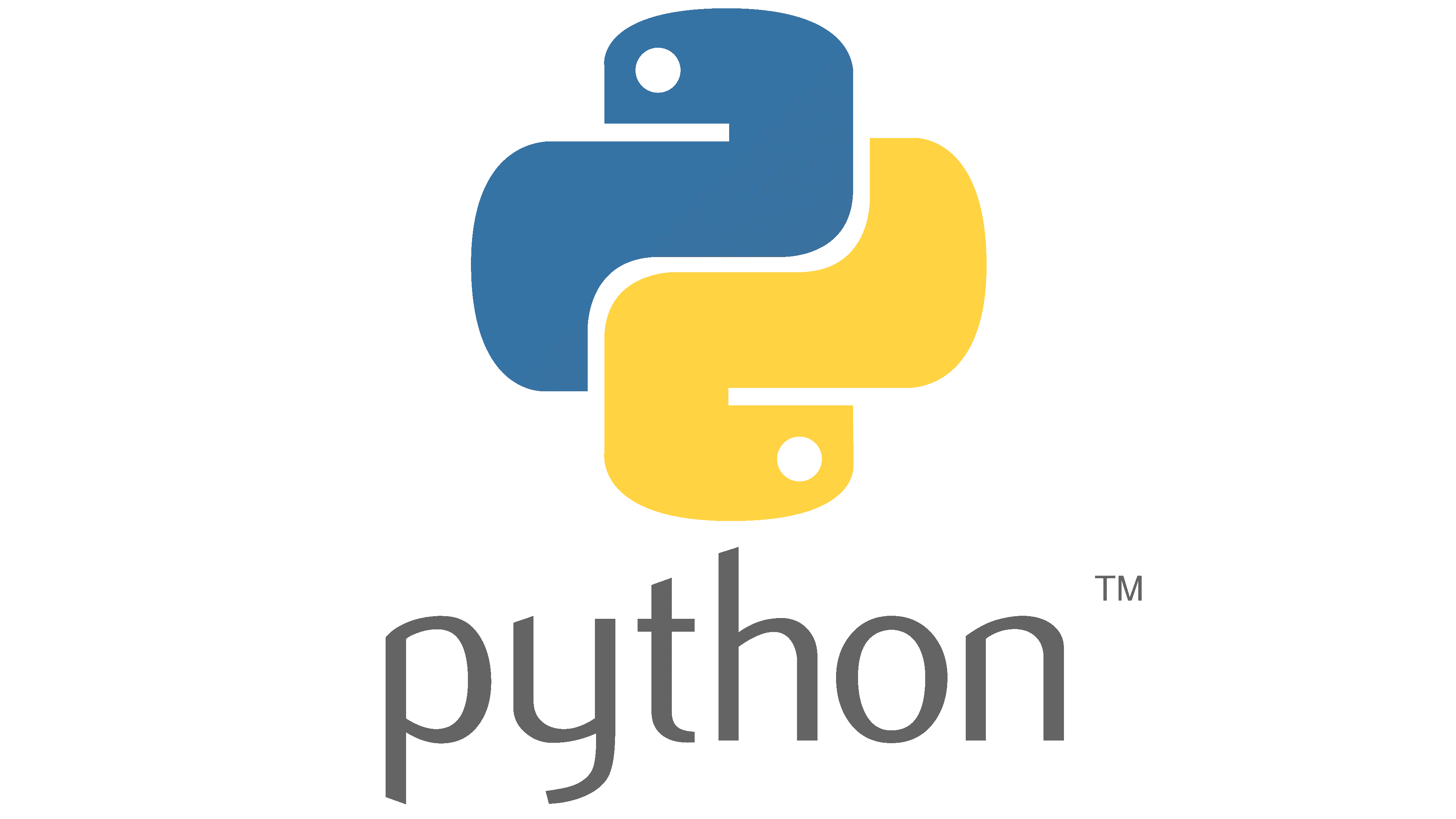
# 2. Prezentarea temei

Tema proiectului constă în dezvoltarea unei soluții capabile să recunoască persoane prin intermediul unui algoritm de recunoaștere facială și să navigheze către locațiile asociate acestora (fiecare persoană este caracterizată printr-un cub de o anumită culoare). Robotul TurtleBot3, utilizat în acest proiect, este echipat cu senzori și camere care îi permit să perceapă mediul înconjurător și să execute sarcini specifice.  
  
Proiectul este structurat pe trei direcții principale:  
- *Recunoaștere facială*: Utilizarea imaginilor pentru recunoașterea persoanelor asociate cuburilor într-un flux video live. Fiecare persoană are atribuit un cub de culoare: verde, albastru, roșu.  
- *Navigație autonomă*: Determinarea și urmărirea unei traiectorii optime către o țintă, utilizând datele de odometrie, senzori, Twist pentru mișcări liniare si angulare, dar și topicul cmd\_vel care ajută la mișcarea robotului.  
- *Interacțiune vocală și feedback-uri*: Notificarea utilizatorului despre starea și progresul sarcinilor executate de robot. În momentul în care robotul reușește să ajungă la unul din cele 3 cuburi (depinde de persoana pe care o vede in cameră),: următorul sunet va fi redat: “Found you!”.De asemenea, la rularea codului principal următorul sunet va fi redat: “Mă uit după ținte”. În terminal, se vor afișa următoarele mesaje: “Detected: numele persoanei detectate”- acest mesaj apare când robotul vede una din cele trei persoane în camera, precedat de textul:” Încă te caut! Rămâi pe loc, vin la tine!”. În momentul în care nu vede pe nimeni în camera, apare următorul text: “Poti fugi, dar nu te poti ascunde!”.

# 3. Tehnologii utilizate

Tehnologiile utilizate în acest proiect includ:  
  
- ***Python***: Limbajul principal pentru dezvoltarea algoritmilor de recunoaștere facială, navigație și interacțiune vocală.  
- ***ROS 2 (Robot Operating System)*:** Middleware pentru integrarea componentelor robotice și gestionarea comunicației între ele.  
- **Gazebo**: Simulator de mediu virtual care permite testarea robotului în condiții controlate.  
- Biblioteci specializate: ***OpenCV*** pentru procesarea imaginilor, ***face\_recognition*** pentru detectarea fețelor și ***pyttsx3*** pentru generarea mesajelor vocale.  
  
O scurtă detaliere despre fiecare în parte

a)Python



Python este un limbaj de programare interpretat, orientat pe obiecte și de nivel înalt, cu semantică dinamică. Structurile sale de date la nivel înalt, combinate cu tipizarea și legarea dinamică, îl fac atractiv pentru dezvoltarea rapidă de aplicații, precum și pentru utilizarea ca limbaj de scripting sau de "lipire" pentru conectarea componentelor existente.

b)ROS2



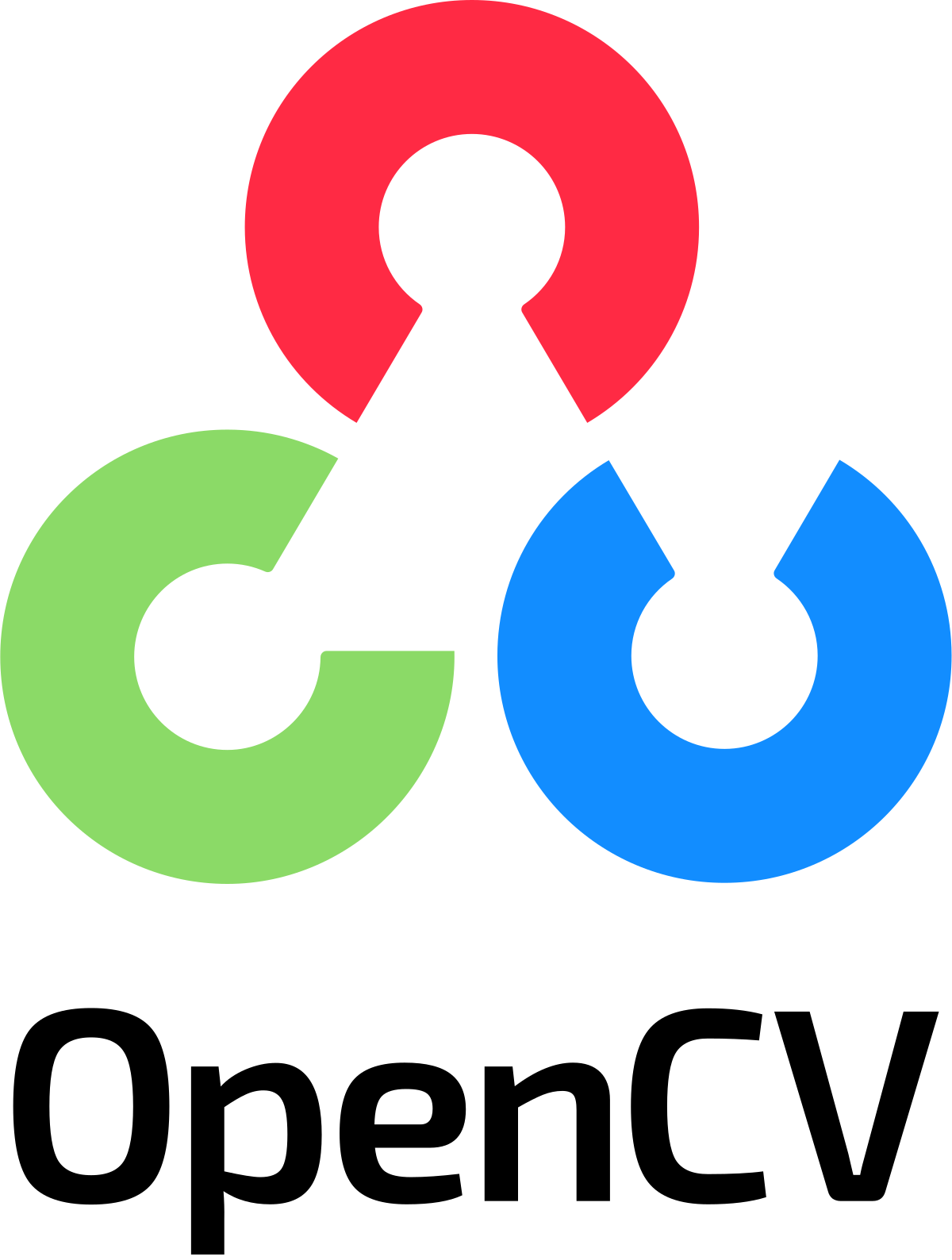
ROS 2 (Robot Operating System 2) este o suită de biblioteci și unelte software care facilitează dezvoltarea aplicațiilor robotice. Acesta oferă funcționalități esențiale precum abstractizarea hardware-ului, controlul dispozitivelor de nivel joasă, implementarea de funcții de utilitate, gestionarea mesajelor între procese și gestionarea pachetelor.

c)Gazebo



**Gazebo** este un simulator open-source 2D/3D pentru robotică, dezvoltat inițial în 2002. Acesta permite simularea precisă și eficientă a populațiilor de roboți în medii complexe, atât interioare, cât și exterioare. Gazebo oferă o simulare fizică de înaltă fidelitate, o suită de senzori și interfețe pentru utilizatori și programe.

d)OpenCV



**OpenCV** (Open Source Computer Vision Library) este o bibliotecă open-source destinată vederii computerizate și învățării automate. Dezvoltată inițial de Intel, OpenCV oferă peste 2.500 de algoritmi optimizați pentru diverse aplicații, inclusiv:

* **Recunoaștere facială**
* **Detectarea și recunoașterea obiectelor**
* **Analiza mișcării**
* **Reconstrucție 3D**
  + - **Realitate augmentată**

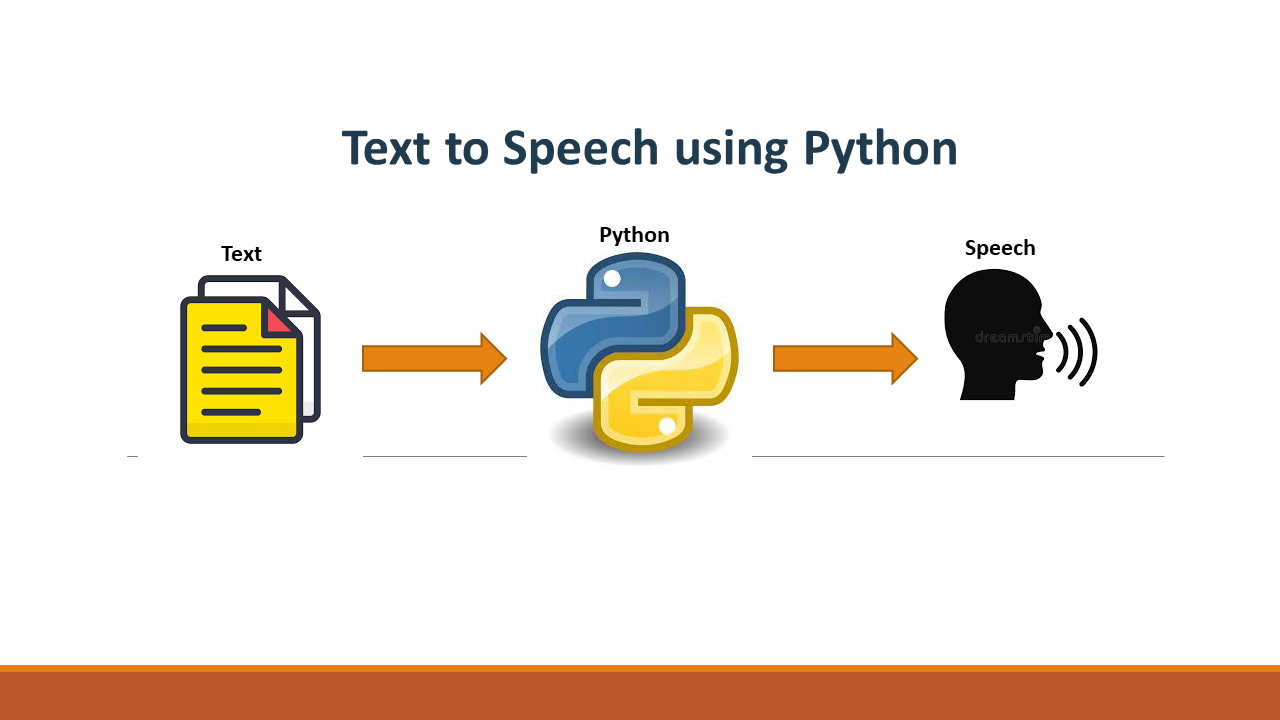
Biblioteca este scrisă în C++ și oferă interfețe pentru limbaje precum Python, Java și MATLAB/Octave, facilitând dezvoltarea aplicațiilor pe multiple platforme, inclusiv Windows, Linux, macOS, Android și iOS

e) **face\_recognition**

**face\_recognition** este o bibliotecă Python open-source care facilitează recunoașterea și manipularea fețelor în imagini. Construită pe baza bibliotecii dlib, aceasta utilizează învățarea profundă pentru a oferi funcționalități precum:

* **Detectarea fețelor**: Identificarea tuturor fețelor prezente într-o imagine.
* **Extracția trăsăturilor faciale**: Determinarea poziției și conturului ochilor, nasului, gurii și bărbiei fiecărei persoane.
* **Recunoașterea fețelor**: Compararea unei fețe necunoscute cu un set de fețe cunoscute pentru a stabili identitatea.

f) pyttsx3

 **pyttsx3** este o bibliotecă Python pentru conversia textului în vorbire (Text-to-Speech - TTS). Spre deosebire de alte biblioteci similare, funcționează offline și este compatibilă cu Python 2 și 3.

**Caracteristici principale:**

* **Conversie TTS offline**: Nu necesită conexiune la internet pentru a funcționa.
* **Compatibilitate multiplatformă**: Suportă Windows, macOS și Linux.
* **Personalizare**: Permite ajustarea ratei de vorbire, volumului și selectarea diferitelor voci disponibile în sistem.

4. Ghidul Programatorului

**a. Descriere Generală a Aplicației**

Aplicația este concepută pentru a permite unui robot TurtleBot să recunoască fețe umane utilizând o cameră, să navigheze autonom către pozițiile corespunzătoare fețelor cunoscute și să evite obstacolele utilizând datele primite de la un senzor LiDAR.

Această aplicație combină funcționalități ROS 2, procesare de imagine („face\_recognition”), algoritmi de navigație, și tehnologii de sinteză vocală („pyttsx3”) pentru a crea un sistem de recunoaștere facială autonom.

O imagine care conține text, captură de ecran, Software multimedia, calculator

Descriere generată automat

**b. Arhitectura Generală a Aplicației**

Aplicația este structurată pe mai multe componente interconectate:

1. **Recunoașterea Facială**:
   * Utilizează biblioteca „face\_recognition” pentru detectarea și identificarea fețelor.
   * Funcționalități principale:
     + Încărcarea imaginilor cu fețe cunoscute.
     + Compararea fețelor detectate cu cele cunoscute pentru identificare.
2. **Navigația Autonomă**:
   * Bazată pe sistemul de mișcare ROS 2 („/cmd\_vel”).
   * Include un algoritm pentru a calcula direcția și distanța necesară pentru a ajunge la o țintă.
3. **Evitarea Obstacolelor**:
   * Utilizează datele de la senzorul LiDAR publicate pe topicul „/scan”.
   * Detectează obstacole și ajustează traiectoria pentru a le ocoli.
4. **Feedback-ul Vocal**:
   * Sinteza vocală este implementată folosind biblioteca „pyttsx3” pentru a oferi utilizatorului informații despre starea robotului.
5. **Interfața Video**:
   * O cameră este utilizată pentru capturarea imaginii necesare detectării faciale.

**c. Structura Codului**

Codul aplicației este organizat în următoarele module principale:

* **turtlebot\_face\_nav.py**: Conține logica principală pentru recunoaștere facială, navigație și evitarea obstacolelor.
* **Fișiere de Resurse**:

O imagine care conține text, captură de ecran, Chip de om

Descriere generată automat

* + Imagini („pozaAbel.jpeg”, „pozaMadalina.jpeg”, și „pozaRoberta.png”).
  + Fișier audio („found\_sound.mp3”).
* **Configurări ROS**:
  + package.xml și setup.py pentru configurarea pachetului ROS.

**d. Fluxul de Date**

1. Camera preia imaginea mediului.
2. Biblioteca „face\_recognition” detectează fețele și le compară cu fețele cunoscute.
3. Dacă este identificată o față cunoscută, aplicația calculează drumul către coordonatele țintei asociate.
4. Senzorul LiDAR verifică obstacolele din drum.
5. Robotul se mișcă și oferă feedback vocal pe parcursul deplasării.

**e. Structura Bazei de Date**

Baza de date este implementată sub forma unui dicționar Python în cod:

self.targets = {

"Caluseri Abel": (2.0, -6.0),

"Iancu Madalina": (0.0, -4.0),

"Roberta Chiton": (-2.0, -6.0)

}

Aceasta mapează numele fețelor cunoscute la coordonatele lor țintă.

**f. Schema bloc generală**

O imagine care conține text, captură de ecran, diagramă, Font

Descriere generată automat

O imagine care conține text

Descriere generată automat

**g. Ierarhia de clase**

TurtleBotFaceNav este clasa principală ce coordonează mișcarea robotului, recunoașterea facială și evitarea obstacolelor.

* Metodele folosite aferente clasei sunt următoarele :
  + \_init\_(): Încarcă fețele cunoscute.

O imagine care conține text, captură de ecran, software

Descriere generată automat

* + update\_position(): Actualizează poziția curentă a robotului.

O imagine care conține text, captură de ecran, Font

Descriere generată automat

* + navigate\_to(): Navighează robotul către o țintă.

O imagine care conține text, captură de ecran, software

Descriere generată automat

* + move\_turtlebot(): Detectează fețele și comanda robotului să se miște.

O imagine care conține text, captură de ecran, software

Descriere generată automat

5. Ghidul Utilizatorului

**a. Descriere Generală**

Seekbot 3 este o aplicație prietenoasă, concepută pentru a permite robotului TurtleBot să recunoască fețe umane și să navigheze autonom către pozițiile acestora. Utilizatorii pot utiliza acest sistem pentru a interacționa cu robotul în moduri inovatoare și practice.

**b. Cerinte de Sistem**

1. **Hardware**:
   * Robot TurtleBot cu senzor LiDAR și cameră integrată.
   * Calculator cu sistem de operare Linux (Ubuntu 20.04).
2. **Software**:
   * ROS 2 instalat (de preferat, distribuția Foxy).
   * Python 3.8 sau mai nou.
   * Biblioteci suplimentare: OpenCV, face\_recognition, pyttsx3, numpy.

**c. Instalare**

1. Asigură-te că toate dependențele sunt instalate (mai sus, în secțiunea de configurare).
2. Clonează repository-ul aplicației și navighează în directorul proiectului:

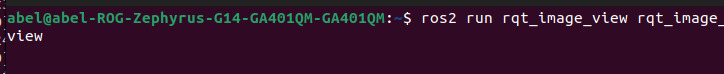
O imagine care conține text, captură de ecran, software, număr

Descriere generată automat

git clone <repository-url>

cd turtlebot\_face\_nav

1. Rulează aplicația folosind comanda ROS:



ros2 run <nume\_pachet> face\_nav\_node

**d. Folosirea Aplicației**

1. **Inițializarea Robotului**:
   * Conectarea robotului la calculatorul principal prin rețea locală.
   * Asigurarea funcționalității tuturor conexiunilor hardware.
2. **Detectarea Fețelor**:
   * Plasarea unei persoane cunoscute în fața camerei.
   * Verificare: dacă robotul identifică corect numele persoanei pe baza mesajelor afișate în terminal.
3. **Navigația către Țintă**:
   * Robotul va calcula automat drumul spre coordonatele asociate feței detectate.
   * Asigurarea spațiului liber sau cu obstacole ușor evitabile.
4. **Feedback-ul Robotului**:
   * Robotul va reda un mesaj vocal când ajunge la țintă (“Am ajuns la țintă!”).
   * Obstacolele sunt evitate automat, iar utilizatorul va fi informat despre acest lucru.

**e. Exemple de Utilizare**

* **Scenario 1**: Un utilizator dorește ca robotul să găsească o persoană specifică într-o cameră mare.
* **Scenario 2**: Robotul este utilizat pentru a demonstra recunoașterea facială într-un mediu educațional.
* **Scenario 3**: Robotul poate livra un mesaj vocal unei persoane recunoscute.

**f. Depanare**

1. **Probleme cu Recunoașterea Faciale**:
   * Asigurare că imaginile utilizate pentru fețe cunoscute sunt clare.
   * Ajustare luminozitatea mediului pentru a îmbunătăți precizia camerei.
2. **Probleme cu Navigația**:
   * Verificare conexiunea dintre robot și sistemul ROS.
   * Asigurare că senzorul LiDAR funcționează corect.
3. **Mesaje de Eroare**:
   * Consultare a log-urilor ROS pentru informații detaliate despre erorile apărute.

Cu acest ghid, utilizatorii pot instala, configura și utiliza aplicația cu succes, iar programatorii pot îmbunătăți sau extinde funcționalitățile existente.

6. Testarea Aplicației

**a. Scopul Testării**

Scopul testării este de a verifica funcționalitatea, stabilitatea și performanța aplicației Seekbot 3, asigurându-se că toate modulele funcționează corect și că aplicația îndeplinește cerințele specificate.

**b. Tipuri de Testare**

1. **Testare Unitară**:
   * Testarea componentelor individuale, precum funcția de recunoaștere facială și algoritmul de navigație.
2. **Testare de Integrare**:
   * Verificarea interacțiunii între subsistemele aplicației, cum ar fi recunoașterea facială și evitarea obstacolelor.
3. **Testare Funcțională**:
   * Validarea funcționalităților principale: identificarea fețelor, navigarea autonomă și evitarea obstacolelor.
4. **Testare de Performanță**:
   * Evaluarea timpului de răspuns pentru recunoașterea facială și viteza de navigație.

**c. Rezultate Așteptate**

1. Robotul identifică corect fețele cunoscute și navighează autonom către coordonatele asociate. Mai multe fețe sunt detectate simultan. Robotul ar trebui să prioritizeze fața cea mai apropiată de camera sa.

O imagine care conține Chip de om, persoană, piele, Frunte

Descriere generată automat

În cazul de mai sus, robotul o vede pe Mădălina și urmează să meargă la cubul asociat ei (adica la cel verde).

În cazul în care robotul nu vede pe nimeni în cameră, vor apărea următoarele mesaje:

O imagine care conține text, captură de ecran, Font

Descriere generată automatO imagine care conține text, captură de ecran, Font, informație

Descriere generată automat

Când cubul căutat este găsit, va fi generat un mesaj de confirmare, precum și timpul în care s-a efectuat operația de găsire.

O imagine care conține text, captură de ecran, Software multimedia, calculator

Descriere generată automat

La finalizarea căutării, se poate observa că obstacolele sunt detectate și evitate fără erori sau coliziuni, iar performanța generală a aplicației rămâne constantă, fără întârzieri semnificative. Totodată, feedback-ul vocal este emis în momentele corespunzătoare.

**e. Rezultatele Testării**

* Testele vor fi considerate reușite dacă toate cerințele specificate în scenariile de testare sunt îndeplinite fără probleme majore.
* În cazul unor deficiențe, aplicația va fi ajustată și testată din nou pentru a asigura conformitatea cu specificațiile.

7. Prezentarea firmei

**Prezentarea echipei dezvoltatoare**

**1. Roberta Chiton: Inițializarea nodului și recunoașterea facială**

Am lucrat la configurarea nodului ROS 2 și integrarea funcționalităților de recunoaștere facială, o componentă esențială pentru proiect. Această secțiune stabilește bazele aplicației, pregătind infrastructura necesară pentru detectarea și identificarea fețelor.

**Responsabilități principale:**

1. Configurarea publisher-ului ROS pentru controlul mișcării robotului și a subscriber-ului pentru odometrie.
2. Integrarea bibliotecii face\_recognition pentru încărcarea și procesarea imaginilor fețelor cunoscute.
3. Inițializarea camerei pentru capturarea imaginilor în timp real.

**Detalii tehnice:**

* Am definit lista de fețe cunoscute și am încărcat imagini relevante, generând encodinguri pentru fiecare față.
* Camera video este inițializată folosind OpenCV, fiind folosită pentru a captura fluxul video în timp real.

**Exemplu de cod:**

****

**Contribuția mea:**

Am creat fundamentul pentru detectarea fețelor, configurând infrastructura ROS și asigurând funcționalitatea de procesare a imaginilor. Acest modul permite robotului să identifice corect persoanele și să inițieze navigația.

**2. Abel Caluseri: Navigarea robotului către țintă**

M-am ocupat de dezvoltarea logicii de navigare a robotului, implementând funcții pentru deplasarea autonomă către coordonatele specifice. Activitatea mea include procesarea poziției curente a robotului și calcularea traiectoriei optime către țintă.

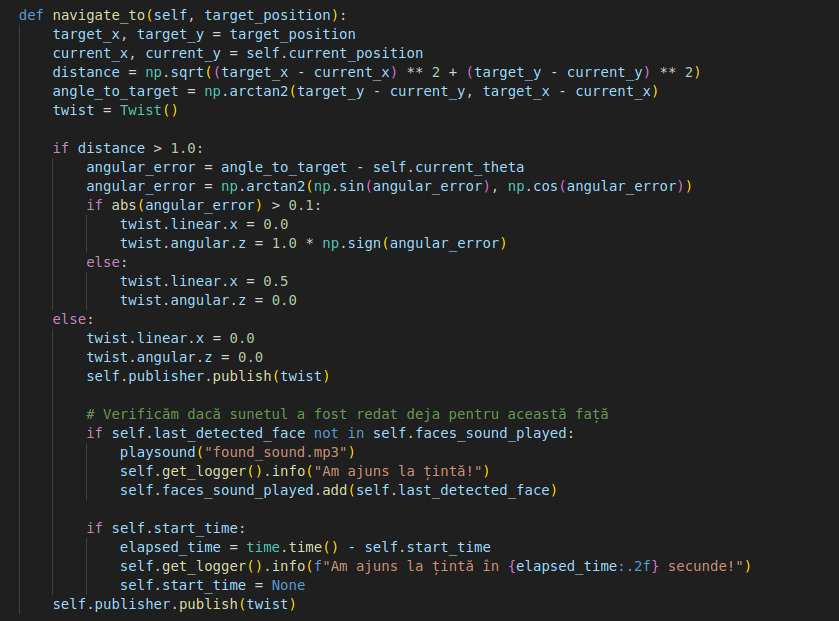
**Responsabilități principale:**

1. Utilizarea datelor de odometrie pentru a calcula poziția curentă a robotului.
2. Calcularea distanței și unghiului de orientare către țintă.
3. Trimiterea comenzilor de mișcare către robot prin publisher-ul ROS.

**Detalii tehnice:**

* Am implementat funcția navigate\_to, care calculează distanța dintre poziția curentă și cea dorită și ajustează orientarea robotului pentru a urmări traseul corect.

**Exemplu de cod:**

**Contribuția mea:**

Am asigurat navigația precisă a robotului, dezvoltând logica prin care acesta se deplasează autonom către pozițiile dorite. Datorită acestei funcționalități, robotul poate atinge cu succes ținta identificată.

**3. Mădălina Iancu: Recunoașterea facială și interacțiunea**

Eu m-am ocupat de implementarea algoritmilor pentru procesarea imaginilor și detecția fețelor. Partea mea a constat în prelucrarea fluxului video capturat de robot, identificarea fețelor și interacția cu acestea.

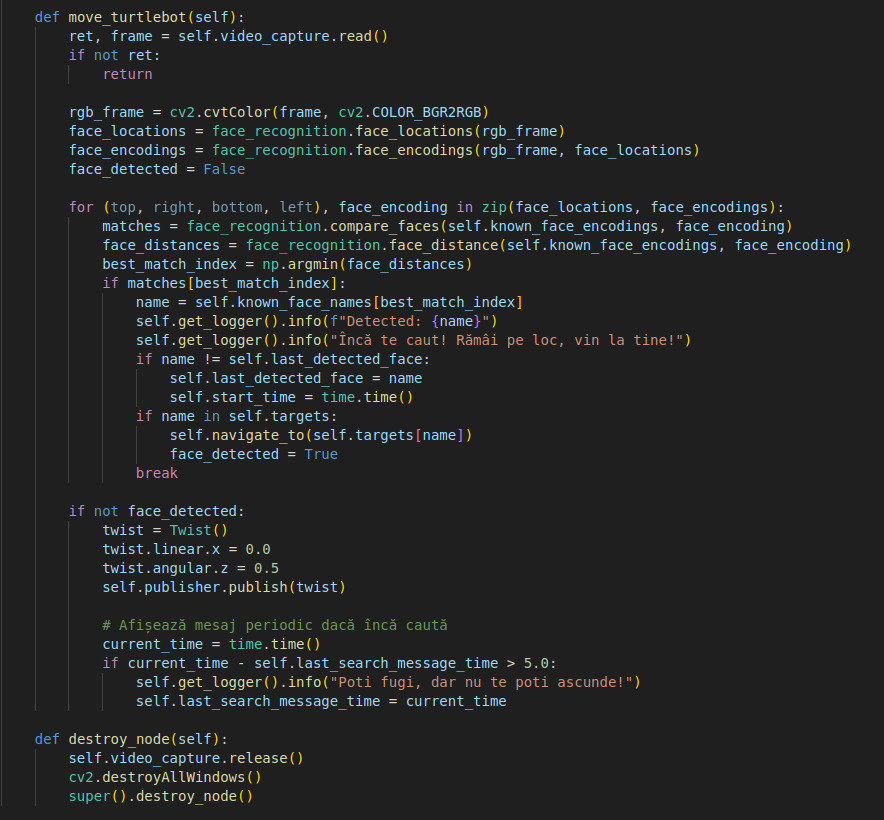
**Responsabilități principale:**

1. Procesarea imaginilor capturate de cameră pentru a detecta fețe în timp real.
2. Compararea fețelor detectate cu baza de date a persoanelor cunoscute.
3. Activarea logicii de navigație către feța identificată.

**Detalii tehnice:**

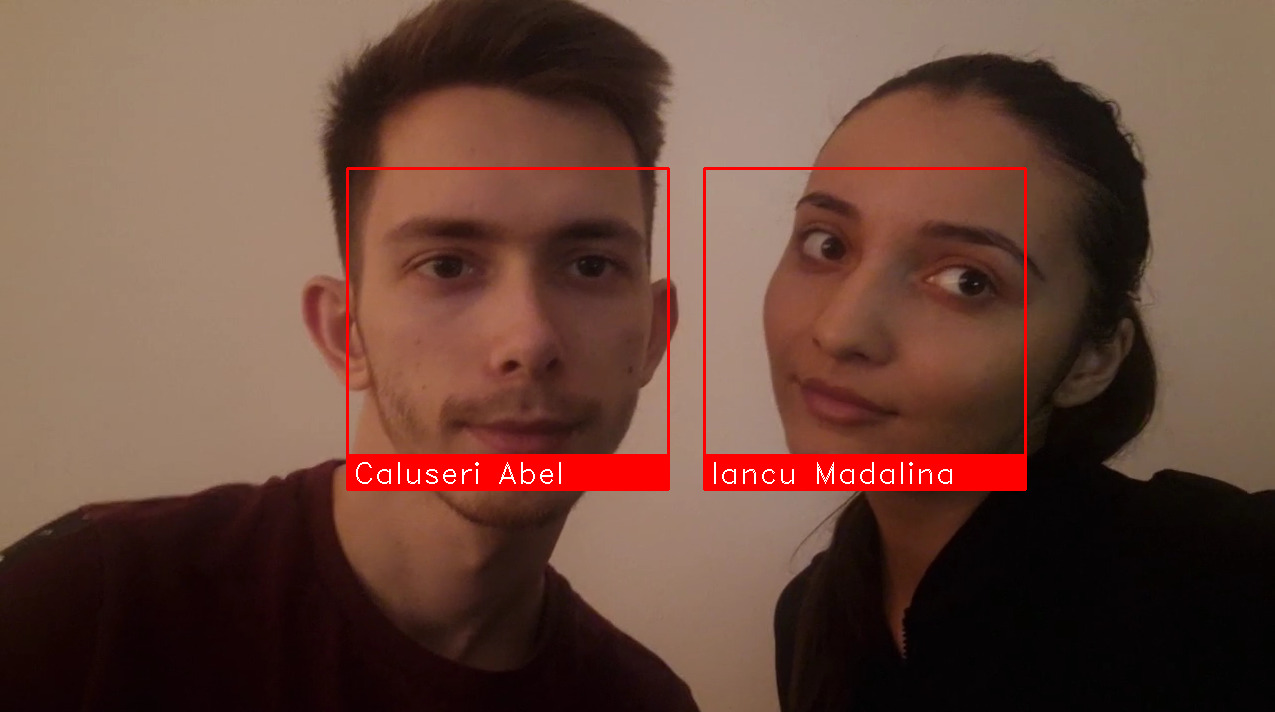
Am implementat funcția move\_turtlebot, care gestionează procesarea imaginilor și detecția fețelor. Dacă o față este recunoscută, se inițiază navigația către poziția asociată.

**Exemplu de cod:**

****

**Explicația funcției:**

* **Capturarea imaginilor:** Robotul capturează cadre de la cameră și le procesează într-un format RGB.
* **Detecția fețelor:** Folosind biblioteca face\_recognition, sunt identificate locațiile fețelor și sunt extrase encodingurile acestora.
* **Identificarea fețelor:** Encodingurile detectate sunt comparate cu cele cunoscute, iar cel mai apropiat rezultat este selectat.
* **Navigarea:** Dacă o față cunoscută este identificată, robotul inițiază navigația către locația asociată.
* **Fallback:** Dacă nu este detectată nicio față, robotul se rotește pentru a continua căutarea.



**Contribuția mea:**

Am dezvoltat funcția de recunoaștere facială și interacțiune, permițând robotului să identifice persoane în timp real și să reacționeze corespunzător. Această funcționalitate este crucială pentru succesul proiectului.

8. Concluzie

Proiectul “Hide&Seek” are la bază scopul folosirii inteligenței artificiale pentru recunoașterea facială a persoanelor din proiect si găsirea lor pe harta din Gazebo sub formă de cuburi.Robotul este capabil să identifice culorile cuburilor și persoanele care stau în fața camerei apoi folosind navigația autonomă să găsească persoana aflată în fața camerei.

Aceste acțiuni complexe sunt:

1. **Identificarea persoanelor**
2. **Corelarea persoanelor cu cuburile**
3. **Navigația către cuburi**
4. **Feedback vocal și text**
5. **Interacțiune în timp real**

Proiectul „Hide&Seek” evidențiază în mod clar sinergia dintre robotică și inteligența artificială, demonstrând cum aceste tehnologii pot colabora pentru a aborda provocări practice și pentru a facilita interacțiuni inovatoare între roboți și oameni. Această platformă deschide numeroase oportunități pentru îmbunătățiri ulterioare, cum ar fi dezvoltarea unui sistem avansat de evitare a obstacolelor, integrarea unor senzori LIDAR de înaltă performanță sau extinderea capacității de recunoaștere pentru a include mai multe persoane și locații într-un mediu diversificat.

9.Bibliografie

[DEV24] <https://dev.to/mr_nova/text-to-speech-with-python-a-beginners-guide-to-pyttsx3-2pie>

[GG24] <https://www.geeksforgeeks.org/a-complete-guide-to-face-detection-and-face-recognition-in-2024/>

[WWW01] <https://en.wikipedia.org/wiki/OpenCV>

[Gaz22]: <https://gazebosim.org/home>

[ROS22] <https://docs.ros.org/en/humble/Installation.html>

[WWW02] <https://en.wikipedia.org/wiki/Python_%28programming_language%29>

[PyB] <https://www.python.org/doc/essays/blurb/>

[KUKA22] <https://www.kuka.com/ro-ro/produse-servicii/amr-robotica-mobila-autonoma/mobile-robot-systems/cobot-mobil-autonom-kmr-iisy>

[StartUp22] <https://start-up.ro/robotii-mobili-autonomi-cum-pot-fi-imbunatatite-operatiunile-facute-de-acestia/>

[Digi22] <https://www.digikey.ro/ro/articles/autonomous-mobile-robot-types-and-applications>

[GitHub] <https://github.com/ageitgey/face_recognition>