



UNIVERSITATEA "POLITEHNICA" DIN TIMISOARA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

DEPARTAMENTUL AUTOMATICĂ ȘI INFORMATICĂ APLICATĂ

HIDE AND SEEK

PROIECT SINCRETIC I

AUTORI:

Marc GHERZAN

Daniel GABOR

Coordonatori:

Conf.dr.ing. Florin DRĂGAN, As. ing. Alexa Liviu-Aniel

Anul III AIA - an universitar 2025 / 2026





CUPRINS

1. Introducere	1
2. Prezentarea temei.....	1
3. Tehnologii utilizate.....	2
4. Ghidul programatorului.....	3
5. Ghidul utilizatorului.....	4
6. Testare si punere in functiune.....	5
7. Concluzii	5
.	
.	
Bibliografie.....	6

|

1. Introducere

În contextul actual al industriei 4.0 și al avansului tehnologic rapid, roboții mobili autonomi au devenit o componentă esențială în domenii variate, de la logistică și explorare spațială, până la asistență domestică. O provocare majoră în robotica rămâne interacțiunea om-mășină (HRI - Human-Robot Interaction). Metodele tradiționale de control, precum joystick-urile sau tastaturile, sunt adesea contraintuitive și necesită echipamente dedicate.

Prezentul proiect propune o abordare modernă a teleoperării și autonomiei: controlul unui robot mobil prin gesturi naturale ale mâinii și capacitatea acestuia de a lua decizii autonome bazate pe mediul înconjurător. Proiectul explorează conceptul de "Vedere Artificială" (Computer Vision) aplicat în bucla de control a unui sistem robotic, eliminând necesitatea dispozitivelor fizice de comandă.

2. Prezentarea temei

Tema proiectului constă în realizarea unui sistem de tip "Hide and Seek", în care un robot mobil (TurtleBot3) interacționează cu un mediu simulat prin intermediul unei camere web și operează într-un mediu simulat.

Obiectivele specifice ale temei sunt:

1. **Teleoperare prin gesturi:** Robotul trebuie să interpreteze semnalele mâinii operatorului (număr de degete, mână stângă/dreaptă) pentru a executa comenzi de mișcare sau schimbare de stare.
2. **Navigare autonomă reactivă:** În modul "Căutare", robotul trebuie să se deplaseze autonom prin mediu, evitând obstacolele folosind datele de la senzorul LiDAR.
3. **Detecția țintei:** Identificarea vizuală a unui obiect de interes (o cutie roșie/doză de suc) folosind camera de la bordul robotului și procesare de imagine în timp real.
4. **Sistem de stări:** Implementarea unui automat de stări finite (FSM) pentru a gestiona tranzițiile între modurile de așteptare, numărare, căutare și celebrare a găsirii țintei.



3.Tehnologii utilizate

Pentru realizarea acestui proiect sincretic, s-au utilizat urmatoarele resurse hardware si software:

Platforma Robotica (Simulare):

- **TurtleBot3 (Model Waffle Pi):** Un robot mobil diferential echipat cu un senzor LiDAR 360 (LDS-01) pentru masurarea distantelor si o camera video Raspberry Pi integrata pentru perceptia vizuala.
- **Gazebo:** Simulator fizic 3D care permite testarea algoritmilor intr-un mediu controlat, replicand fizica reala (gravitatie, frecare, coliziuni).

Software si Medii de Programare:

- **Sistem de Operare:** Ubuntu 22.04 LTS (Jammy Jellyfish).
- **Middleware:** ROS 2 Humble Hawksbill (Robot Operating System) – asigura comunicatia intre noduri prin topic-uri (Publish/Subscribe).
- **Limbaj de programare:** Python 3.10.
- **Biblioteci principale:**
 - *MediaPipe:* Dezvoltat de Google, utilizat pentru detectia scheletului mainii si extragerea celor 21 de puncte caracteristice (landmarks).
 - *OpenCV (Open Source Computer Vision Library):* Utilizat pentru procesarea fluxului video, conversia spatiilor de culoare (RGB la HSV) si detectia contururilor obiectelor.
 - *NumPy:* Pentru calcule matematice si manipularea matricelor de date senzoriale.



4. Ghidul programatorului

Aplicatia este structurata sub forma unui pachet ROS 2 denumit turtle_controller, continand nodul principal hide_and_seek_node.

Arhitectura Generala:

Sistemul functioneaza pe baza unui flux de date continuu:

1. **Input Senzorial:** Camera Laptop (Gesturi) + LiDAR Robot (Distante) + Camera Robot (Imagine).
2. **Procesare (Nodul Central):**
 - Sub modul Vision: Analizeaza gesturile mainii.
 - Sub modul Logica: Masina de Stari Finite (FSM).
 - Sub modul Navigare: Algoritm de evitare obstacole (Front/Left/Right logic).
3. **Output:** Comenzi de viteza (geometry_msgs/Twist) trimise pe topicul /cmd_vel.

Ierarhia de Clase:

Clasa principala HideAndSeekNode mosteneste clasa de baza rclpy.node.Node. Metodele principale sunt:

- `__init__`: Initializeaza subscriberii (/scan, /camera/image_raw) si publisherul (/cmd_vel).
- `timer_loop()`: Functia recurenta (10Hz) care proceseaza imaginea de la webcam si decide actiunea curenta.
- `count_fingers(landmarks, label)`: Algoritmul care determina numarul de degete ridicate si face distinctia intre mana stanga si cea dreapta pe baza pozitiei degetului mare relativ la articulatie.
- `get_scan_regions()`: Imparte datele LiDAR in trei sectoare (Stanga, Fata, Dreapta) pentru decizii de navigare.
- `camera_callback(msg)`: Proceseaza imaginea de la robot pentru a detecta culoarea rosie (tinta).

Structura Datelor:

Variabilele de stare sunt gestionate intern:

- `self.state`: String (valori: "IDLE", "COUNTING", "SEEKING", "HINT_LEFT", "HINT_RIGHT", "FOUND").

`self.laser_ranges`: Lista de valori float reprezentand distantele masurate de LiDAR.



5.Ghidul utilizatorului

Aplicatia permite controlul robotului printr-o interfata vizuala simpla. La rularea programului, se va deschide o fereastră intitulată "Game Master" care afisează fluxul video al utilizatorului și starea curentă a robotului.

Legenda Gesturi:

Numar Degete	Mana	Actiune Robot	Descriere
0 degete / pumn	Oricare	STOP	Robotul se opreste imediat. Mod de siguranta.
1 Deget (Index)	Oricare	NUMARARE	Robotul se opreste, se roteste si asteapta 5 secunde ("numara").
5 Degete (Palma)	Oricare	CAUTARE (SEEK)	Robotul intra in mod autonom. Evita obstacole si cauta tinta.
2 Degete	Mana Stanga	VIRAJ STANGA	Comanda manuala. Forteaza robotul sa vireze la stanga.
2 Degete	Mana Dreapta	VIRAJ DREAPTA	Comanda manuala. Forteaza robotul sa vireze la dreapta.

Deoarece camera web actioneaza ca o oglinda, ridicarea fizica a mainii drepte poate fi interpretata vizual ca mana stanga. Sistemul a fost calibrat pentru a corecta aceasta inversare, astfel incat ridicarea mainii drepte reale sa determine virajul la dreapta.



6. Testare si punere in functiune

Instalare:

1. Se copiaza pachetul turtle_controller in directorul src al spatiului de lucru ROS 2.
2. Se instaleaza dependentele Python: pip3 install mediapipe opencv-python.
3. Se compileaza proiectul: colcon build --symlink-install.

Lansare:

Se deschid doua terminale:

Terminal 1 (Simulare):

```
export TURTLEBOT3_MODEL=waffle_pi
```

```
ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
```

Terminal 2 (Control):

```
source install/setup.bash
```

```
ros2 run turtle_controller hand_control
```

Scenariu de Test:

S-a plasat un obiect rosu (Coke Can) in spatele unui perete in Gazebo. La semnalul de "5 degete", robotul a navigat autonom evitand doi pereti, a identificat culoarea rosie in spectrul HSV, a oprit motoarele si a afisat mesajul de succes, validand functionalitatea completa.

7. Concluzii

Proiectul a demonstrat cu succes posibilitatea controlului intuitiv al unui robot mobil cu ajutorul atat al senzorilor de pe robot, cat si cu ajutorul interpretarii miscarilor mainii.

Principalele provocari intampinate au fost gestionarea zgomotului senzorilor in simulare, conflictele de dependente software, detectarea corecta a mainii (dreapta / stanga), gasirea unui algoritm potrivit pentru cautarea automata ca robotul sa nu ramana blocat. Pe viitor, sistemul poate fi imbunatatit prin adaugarea algoritmilor de cartografiere (SLAM) pentru ca robotul sa nu caute orb, ci sa exploreze metodic spatiul necunoscut.



Bibliografie

[Lug22] Lugaresi, C., et al. (2022) MediaPipe: A Framework for Building Perception Pipelines, Google Research;

[Ope23] Open Source Robotics Foundation. (2023) ROS 2 Humble Documentation, docs.ros.org;

[Rob20] Robotis Co. (2020) TurtleBot3 e-Manual, emanual.robotis.com;

[Bra00] Bradski, G. (2000) The OpenCV Library, Dr. Dobb's Journal of Software Tools;

[Mac02] MacQueen, J. (2002) Some methods for classification and analysis of multivariate observations, Proc. 5th Berkeley Symp. Math. Statist. Prob.;



