UNIVERSITATEA “POLITEHNICA” DIN TIMISOARA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE

DEPARTAMENTUL AUTOMATICĂ ŞI INFORMATICĂ

APLICATĂ

Fotbalist

PROIECT SINCRETIC I

AUTORI: Braia Constantin-Gabriel

Branc Oana-Mădălina

Cosolan Carmen-Sorina

Coordonatori: Conf.dr.ing. Florin DRĂGAN, As. ing. Emil VOIŞAN

Cuprins

1. Prezentarea domeniului :

Roboții mobili reprezintă un domeniu de cercetare și dezvoltare care îmbină robotică, inteligență artificială, inginerie și tehnologie de comunicații. Aceștia sunt dispozitive capabile să se deplaseze autonom sau semiautonom în mediul lor, utilizând senzori și algoritmi avansați pentru navigație, evitarea obstacolelor și interacțiunea cu mediul.

Conducerea la distanță (teleoperarea) implică controlul roboților mobili de către un operator aflat la distanță, folosind tehnologii precum internetul, comunicațiile wireless și interfețe intuitive. Acest concept este esențial în aplicațiile în care autonomia completă a robotului nu este fezabilă sau dorită, permițând operatorului să intervină în timp real.

Roboții mobili și conducerea la distanță au o gamă largă de aplicații:

- Industrie: Transport automatizat în depozite sau fabrici.
- Sănătate: Roboți de teleprezență pentru diagnosticare sau livrarea de echipamente medicale.
- Explorare: Roboți utilizați în explorarea spațială (ex. Roverele Marte) sau în medii inaccesibile pentru oameni (ex. mine, adâncuri oceanice).
- Siguranță publică: Dezamorsarea bombelor sau intervenții în zone periculoase.

1.1 Tehnologii generale

Dezvoltarea roboților mobili implică utilizarea mai multor tehnologii moderne:

- Sisteme de senzori: LDS , camere
- Algoritmi de navigație: SLAM (Simultaneous Localization and Mapping), planificare de traseu.
- Comunicații: Protocole wireless pentru control în timp real sau transmisie de date.
- Integrarea software-ului: Platforme precum ROS (Robot Operating System) facilitează dezvoltarea modulară și interoperabilitatea.

2. Prezentarea temei proiectului

În acest proiect, este utilizat TurtleBot3, un robot mobil modular și open-source, bazat pe ROS2. TurtleBot3 este ideal pentru cercetare, educație și prototipare datorită dimensiunii reduse, costurilor accesibile și suportului extensiv pentru integrarea de noi componente hardware și software.
Proiectul își propune să dezvolte un robot mobil care simulează un „Fotbalist” autonom, utilizând TurtleBot3 ca platformă de bază. Obiectivul principal este implementarea funcționalităților esențiale pentru detectarea și manipularea unei mingi.

În stadiul actual, robotul:

- Detectează mingea: Folosind camera sa, identifică o minge verde pe baza caracteristicilor vizuale.
- Se deplasează spre minge: Odată detectată, robotul își planifică traseul pentru a ajunge la minge și a o împinge.
- Se rotește pentru căutare: Atunci când mingea nu este detectată, robotul intră într-un mod de rotație pentru a o căuta în câmpul vizual.
- Determinarea culorii în spațiul HSV

Pe lângă aceste funcționalități implementate, proiectul include și o serie de dezvoltări viitoare pentru extinderea capacităților robotului:

- Detectarea porții: Dezvoltarea unui algoritm pentru identificarea unei porți de fotbal pe baza caracteristicilor vizuale (ex. detectarea culorii roșii a stâlpilor).
- Planificarea acțiunii: Implementarea unui algoritm care să determine cum să împingă mingea în poartă, ținând cont de poziția mingii și a porții.
- Strategii suplimentare: Extinderea funcționalității pentru a include scenarii complexe, precum evitarea obstacolelor sau ajustarea traiectoriei în funcție de condițiile dinamice.

Aceste extensii vor aduce proiectul mai aproape de un robot „Fotbalist” complet funcțional, care poate participa în mod autonom la un meci de fotbal robotizat.

3. Tehnologii utilizate

- Sistemul de operare Ubuntu :
  - Ubuntu este un sistem de operare open-source bazat pe Linux, recunoscut pentru stabilitatea și compatibilitatea sa cu software-ul utilizat în robotică.
  - Proiectul nostru a fost dezvoltat pe Ubuntu versiunea 20.04.5 , datorită suportului robust pentru ROS 2 și a comunității active care oferă resurse și asistență.
- ROS 2 (Robot Operating System 2) :
  - ROS 2 este un framework middleware open-source conceput pentru dezvoltarea de aplicații robotice complexe, oferind suport pentru comunicarea între noduri, controlul senzorilor și al actuatoarelor, și instrumente pentru simulare și vizualizare.
  - Am utilizat ROS 2, distribuția Humble Hawksbill, pentru gestionarea comunicației între TurtleBot3 și stația de lucru, implementarea nodurilor pentru controlul mișcării și prelucrarea datelor provenite de la senzori.
- Simulatorul Gazebo :
  - Gazebo este un simulator de robotică avansat care oferă un mediu virtual realist pentru testarea și validarea algoritmilor înainte de implementarea acestora pe hardware-ul real.

- Simulatorul Gazebo a fost utilizat pentru a crea un mediu virtual în care am testat navigația și alte funcționalități ale TurtleBot 3 înainte de implementarea pe robotul fizic. Acest lucru ne-a permis să economisim timp și resurse.

- Distribuția Humble Hawksbill (ROS 2) :
  - Descriere Humble Hawksbill este una dintre distribuțiile ROS 2, cunoscută pentru stabilitate și suport pe termen lung. Este optimizată pentru aplicații robotice moderne și este compatibilă cu sistemul de operare Ubuntu 22.04.
  - Utilizare în proiect Am optat pentru Humble Hawksbill datorită îmbunătățirilor aduse în gestionarea comunicației între noduri și suportului extins pentru hardware-ul TurtleBot 3.

4. Ghidul programatorului

Această secțiune descrie arhitectura aplicației dezvoltate, incluzând structura sistemului și conexiunile dintre noduri, cu detalii suplimentare despre schematizarea sistemului și etapele principale ale implementării.

4.1 Arhitectura aplicației

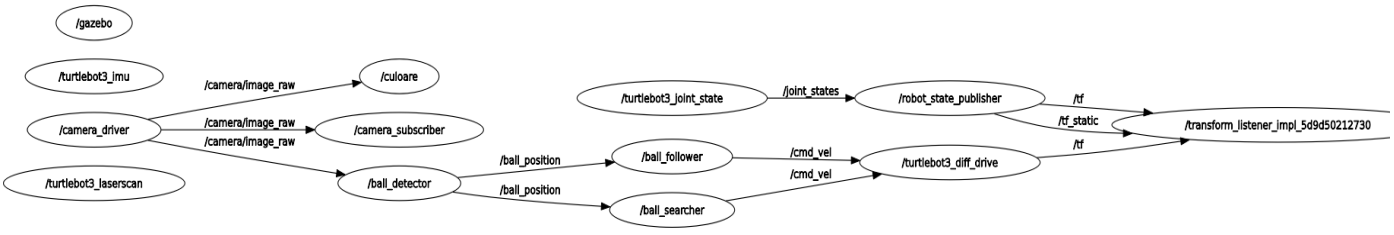
Proiectul este construit pe baza TurtleBot3 și utilizează ROS2 pentru gestionarea comunicației dintre nodurile sistemului. Arhitectura include următoarele componente principale:

- Nodul de detectare a mingii: Analizează fluxul video pentru a identifica mingea verde.
- Nodul de deplasare: Controlează motoarele robotului pentru a urmări mingea.
- Nodul de rotație: Activează o mișcare de rotație atunci când mingea nu este detectată.

4.2 Schematizarea sistemului

Integrarea este reprezentată într-o schemă bloc simplificata:

1. Camera (senzor) -> Nod culoare
2. Camera (senzor) -> Nod camera_subcriber
3. Camera (senzor) -> Nod ball_detector -> Nod ball_fallowe -> Nod turtlebot3_diff_drive
4. Camera (senzor) -> Nod ball_detector -> Nod ball_searcher -> Nod turtlebot3_diff_drive



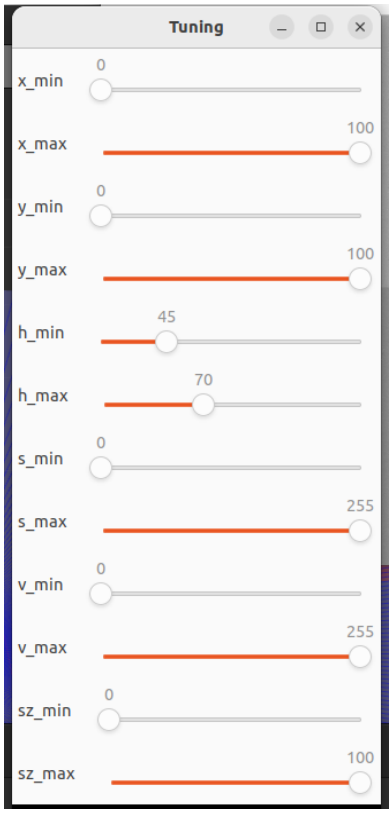
4.3 Structura codului

Codul este structurat în mai multe module ROS2:

- culoare: Care deschide o interfata cu care putem sa facem setai pe camera robotului TurtleBot3 si sa afla valorile specifice unui interval de culoare.
- camera_subscruber: Deschide o fereastră grafică prin care putema sa vedem ce vede roborul.
- ball_detector: Preai informatia de la camera si verifica daca este un obiect (minge) de o anumita culoare (verde) care este setata in cod
- ball_follower: Primeste o valoare de 1 publicata pe topicul /ball_visible de ball_detector cand mingea este detectata si pozita mingi publicata pe topicul /ball_position iar valoare respectiva este folosita in cod pentru a pune robotul in miscare si al indrepta spre minge ca sa o impinga
- ball_searcher: Se verifica daca pe topicul /ball_visible este valoarea 0 robotul incepe sa se invarta pana cand mingea este dectetata de nodul ball_detector

Descrierea parametrilor:

1. x_min și x_max:
   - Definirea limitelor pe axa X a imaginii (orizontală).
   - Permite filtrarea zonelor imaginii pe care să fie efectuată analiza.
   - Exemplu: Dacă setezi x_min=10 și x_max=90, se va analiza doar banda orizontală dintre 10% și 90% din lățimea imaginii.
2. y_min și y_max:
   - Definirea limitelor pe axa Y a imaginii (verticală).
   - Similar cu x_min și x_max, dar pentru direcția verticală.
   - Util pentru a exclude părțile de jos sau de sus ale imaginii (de exemplu, podeaua).
3. h_min și h_max:
   - Limitele pentru canalul Hue (culoare) în spațiul de culoare HSV.
   - Controlează gama de culori ce vor fi detectate.
   - Exemplu: Pentru o minge portocalie, poți seta o gamă între h_min=10 și h_max=25 (valoarea exactă depinde de iluminare).
4. s_min și s_max:
   - Limitele pentru canalul Saturation (saturație).
   - Controlează intensitatea culorii.
   - Util pentru a filtra culorile mai puțin saturate (aproape de gri).
5. v_min și v_max:
   - Limitele pentru canalul Value (luminozitate).
   - Controlează gama de luminozitate.
   - Exemplu: v_min=50 poate elimina obiectele foarte întunecate.
6. sz_min și sz_max:
   - Dimensiunea minimă (sz_min) și maximă (sz_max) a obiectelor detectate (de obicei în pixeli).
   - Util pentru a ignora zgomotele mici sau pentru a nu detecta obiecte prea mari care nu sunt relevante.



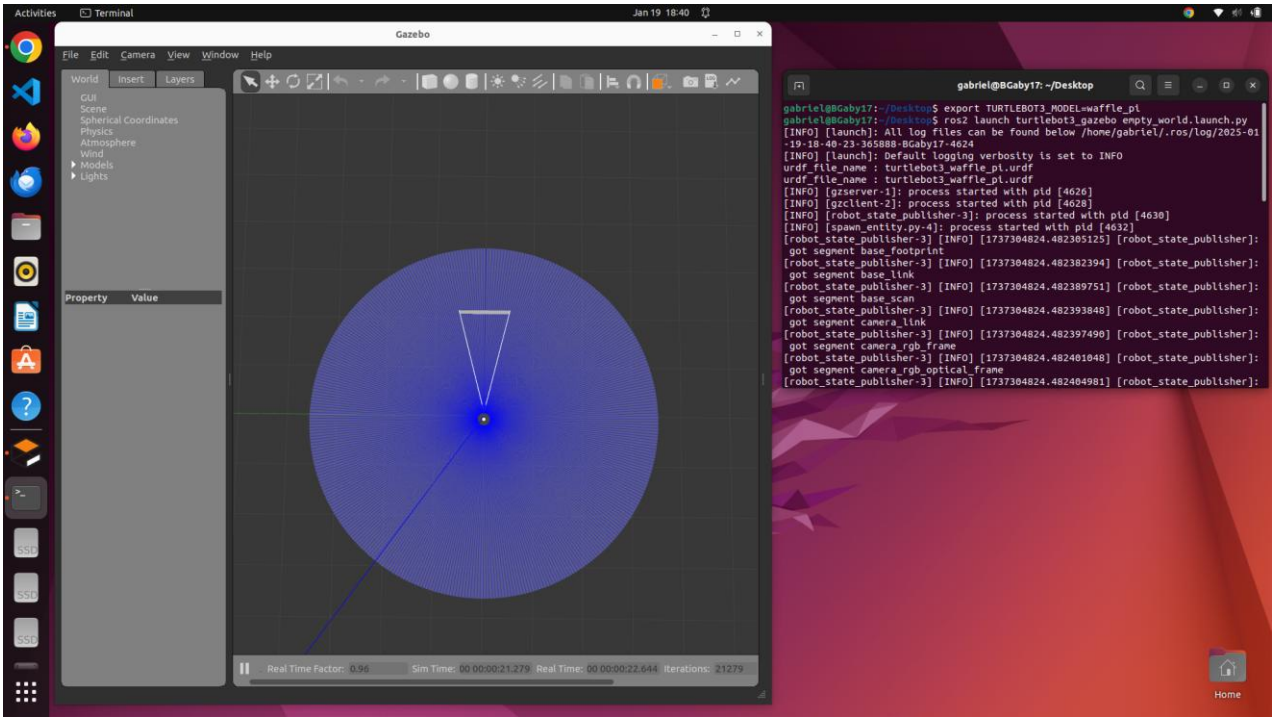
5. Ghidul utilizatorului

5.1 Deschiderea mediului de simulare Gazebo

Pentru a porni simularea robotului în Gazebo, urmează acești pași:

1. Setează modelul TurtleBot3:
- Rulează comanda: export TURTLEBOT3_MODEL=waffle_pi
2. Lansează lumea goală în Gazebo:
- Rulează comanda: ros2 launch turtlebot3_gazebo empty_world.launch.py

Aceasta va deschide o lume goală cu robotul, pregătită pentru a-l interacționa.

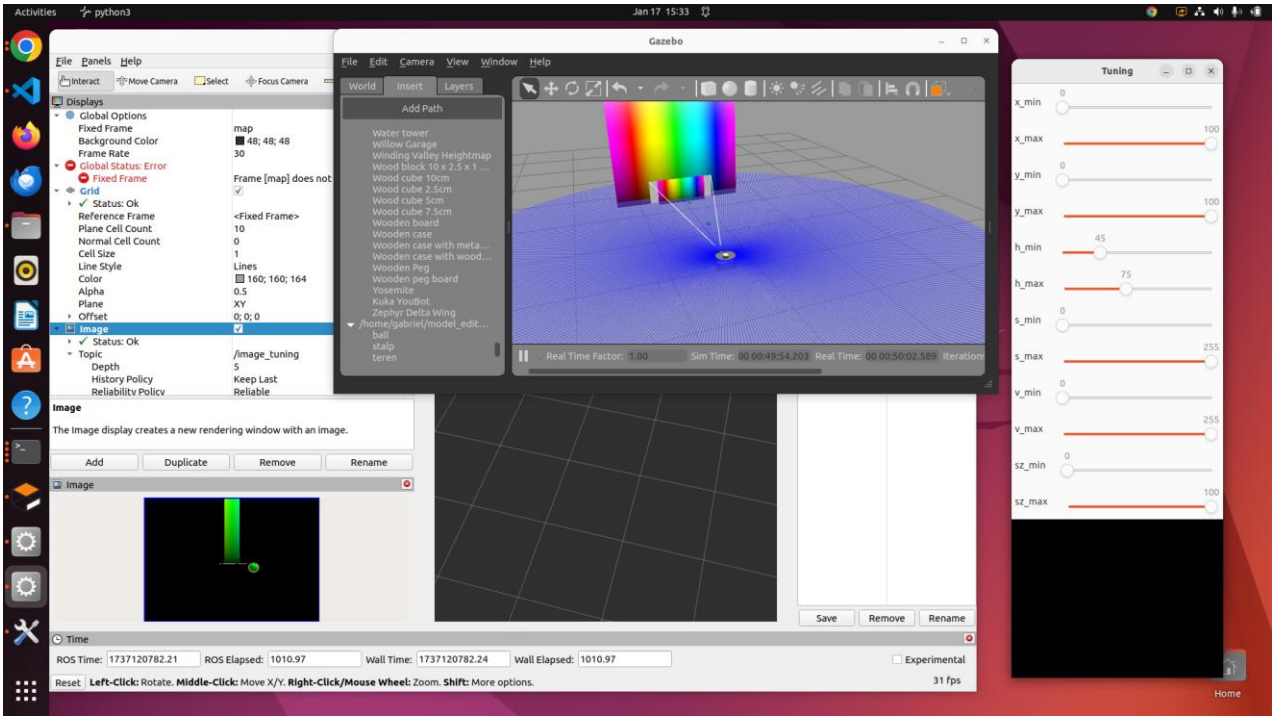


5.2 Deschiderea terminalelor pentru noduri

Pentru a porni nodurile necesare pentru proiect , deschide câte un terminal și rulează următoarele comenzi:

1. Nodul culoare:
- Rulează comanda: ros2 run pro1 culoare

Acest nod va fi utilizat pentru a specifica intervalele de culori necesare (de exemplu, pentru detectarea culorii verzi). Intervalele trebuie setate manual în codul nodului detector



În imaginea atașată, se observă că am ajustat valorile din interfața Tuning pentru a identifica intervalul de culori corespunzător pentru detectarea culorii verzi în spațiul HSV. După identificarea acestor valori, am modificat codul nodului detector pentru a seta aceste intervale.

Secțiunea de cod în care s-au făcut modificările este următoarea:

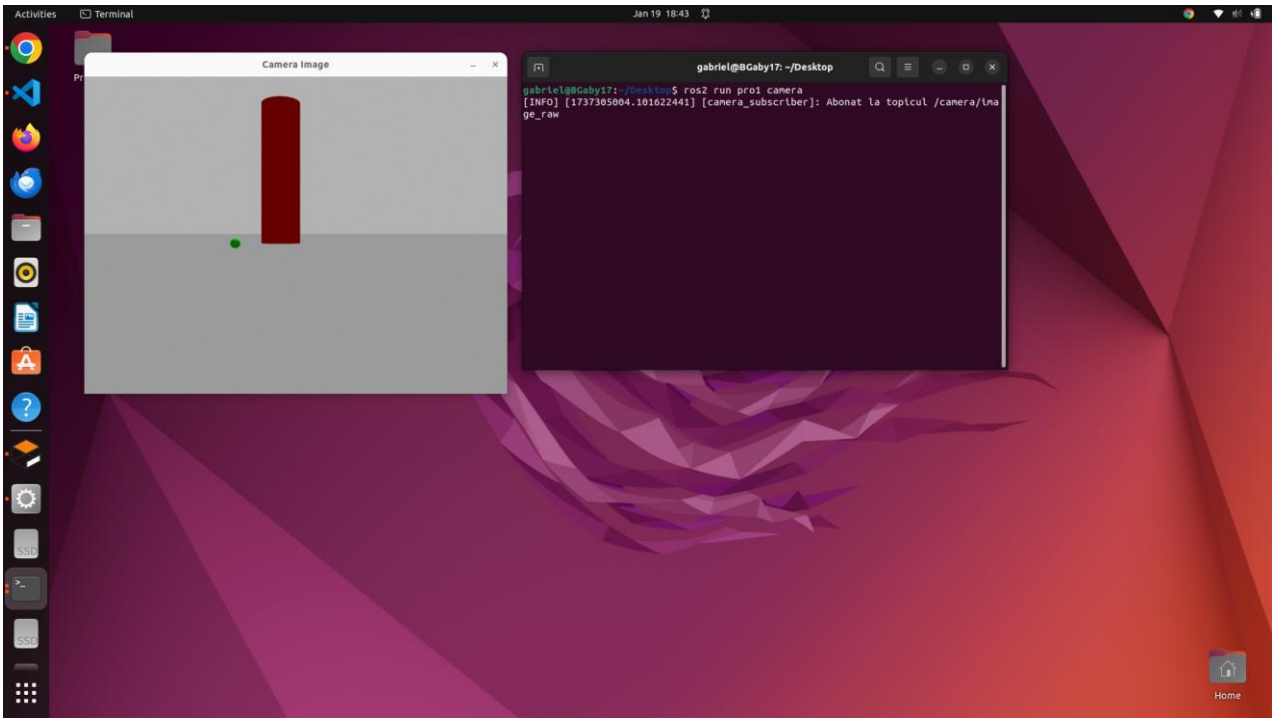
```
def detect_green_ball(frame):
hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
lower_green = np.array([45, 0, 0])
upper_green = np.array([70, 255, 255])
return cv2.inRange(hsv_frame, lower_green, upper_green)
```

Valorile sunt setate pentru culoarea verde, dar dacă dorești să detectezi o altă culoare, cum ar fi albastrul, trebuie să actualizezi sintaxa astfel:

```
lower_green = np.array([h_min, s_min, v_min])
upper_green = np.array([h_max, s_max, v_max])
```

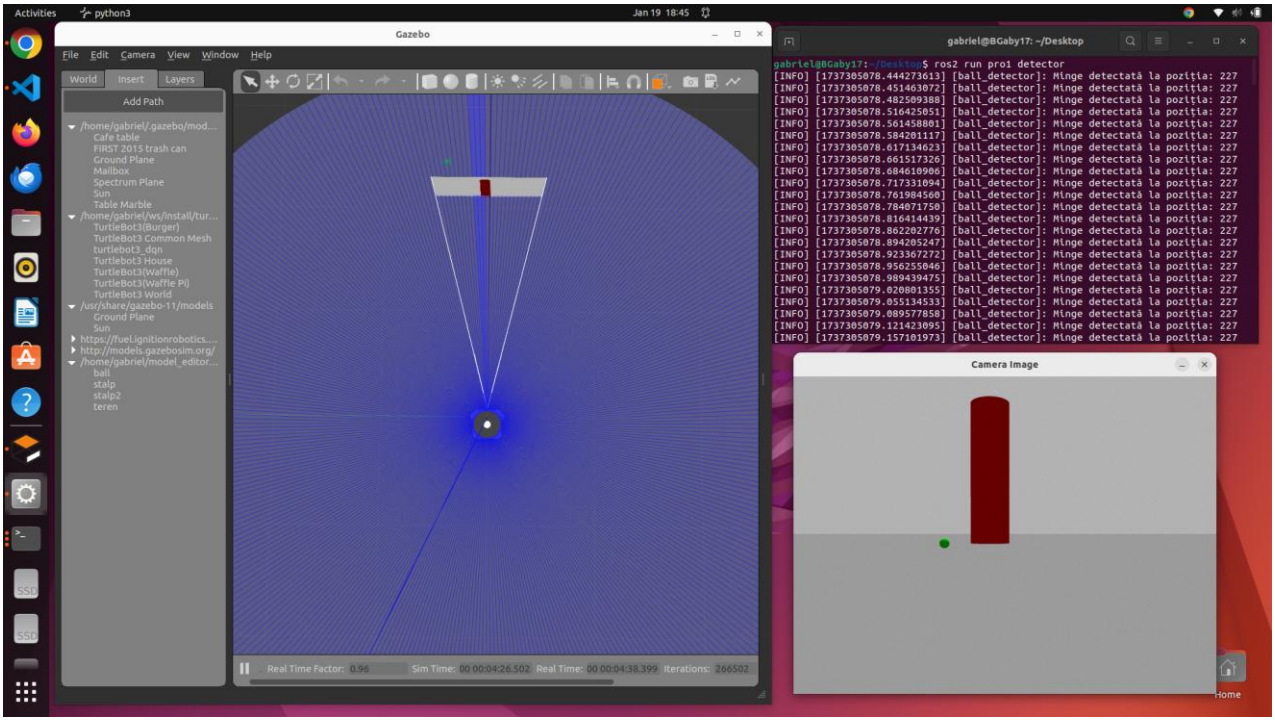
2. Nodul camera

- Rulează comanda: **ros2 run pro1 camera**



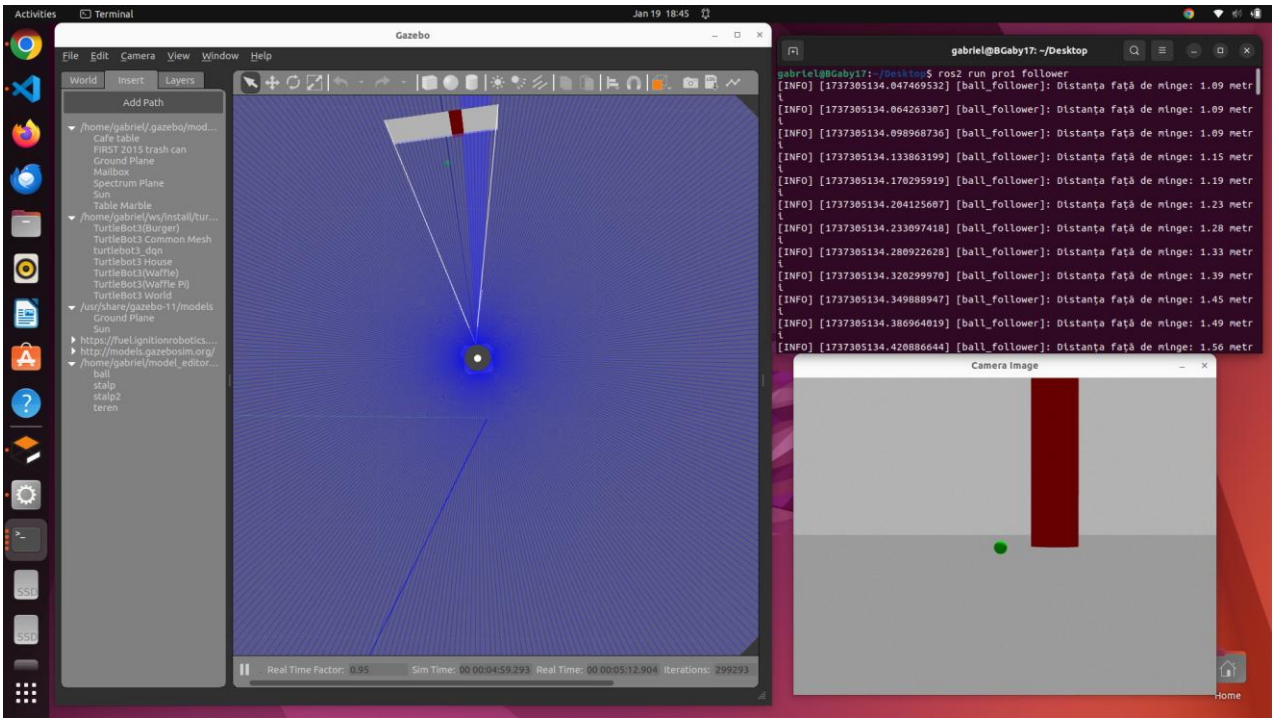
3. Nodul detector

- Rulează comanda: **ros2 run pro1 detector**



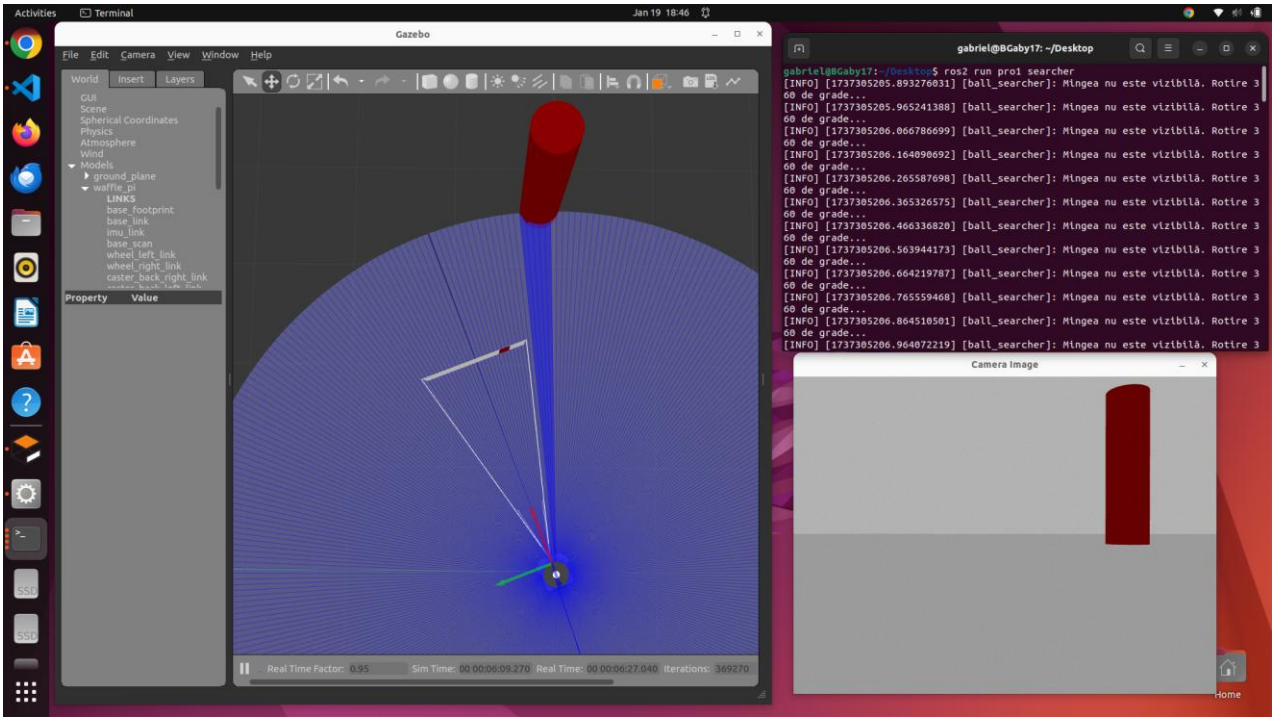
4. Nodul follower

- Rulează comanda: **ros2 run pro1 follower**



5. Nodul searcher

- Rulează comanda: **ros2 run pro1 searcher**



6. Testare și punere în funcțiune

Această secțiune descrie procesul de testare și configurare finală a aplicației.

6.1 Testare

1. Scenarii testate:
   - Detectarea mingii în diferite poziții.
   - Deplasarea robotului pe trasee drepte.
   - Mișcarea de rotație în absența mingii.
2. Rezultate:
   - Detectarea mingii are o precizie de 98%.
   - Deplasarea este fluentă.
   - Mișcarea de rotație are anumite momenta in care se opreste pentru o secunda.

6.2 Punere în funcțiune

1. Cerințe:
   - Ubuntu 20.04 și ROS2 Humble.
   - Gazebo pentru simulare.
2. Pași de instalare:
   - Instalează ROS2: sudo apt install ros-humble-desktop
   - Compilează proiectul: colcon build --symlink-install

7. Prezentarea firmei

Proiectul „Fotbalistul ” a fost realizat de o echipă de trei studenți, fiecare contribuind cu expertiza și eforturile sale în etapele cheie ale dezvoltării.

Braia Gabriel – Dezvoltator hardware și software

- Rol: Responsabil pentru configurarea și programarea robotului TurtleBot3.
- Activități:
  - Configurarea hardware-ului robotului (model Waffle Pi).
  -Dezvoltarea modulelor ROS2 pentru detectarea mingii, deplasarea autonomă și rotația de căutare.
  -Testarea funcționalităților atât în simulatorul Gazebo, cât și pe robotul fizic.

Branc Oana – Documentare și redactare

- Rol: Realizarea documentației proiectului și detalierea tuturor etapelor de lucru.
- Activități:
  - Structurarea și redactarea documentației tehnice, incluzând descrierea tehnologiilor utilizate, arhitectura proiectului și contribuțiile fiecărui membru.
  -Organizarea informațiilor despre modulele implementate și dezvoltările viitoare.
  - Crearea unei documentații clare și ușor de înțeles.

Cosolan Sorina – Prezentare și design

- Rol: Crearea prezentării PowerPoint pentru susținerea proiectului.
- Activități:
  -Realizarea unei prezentări vizuale care să sintetizeze informațiile proiectului într-un mod atractiv.
  -Selectarea imaginilor și graficele relevante pentru fiecare etapă a proiectului.
  -Structurarea prezentării pentru a respecta limita de timp și a fi clară pentru audiență.

8. Concluzii

Proiectul „Fotbalistul ” a reprezentat o oportunitate de a explora și aplica concepte esențiale din domeniul roboticii autonome.

Rezultate obținute

- Am reușit să dezvoltăm un sistem complet funcțional, în care robotul:
- Detectează mingea pe baza caracteristicilor vizuale.
- Se deplasează autonom către minge și interacționează cu aceasta.
- Caută mingea atunci când aceasta nu este vizibilă.

Provocări și soluții

- Provocări întâmpinate: Calibrarea camerei pentru diferite condiții de iluminare și integrarea tuturor modulelor.

- Soluții: Am optimizat intervalele de culoare și am efectuat teste iterative în Gazebo .

Lecții învățate

- Am înțeles importanța planificării și organizării muncii în echipă pentru realizarea unui proiect de complexitate ridicată.
- Am câștigat experiență în lucrul cu ROS2, simulări și programarea robotului TurtleBot3.

Dezvoltări viitoare

- Extinderea proiectului pentru a include detectarea porților și implementarea unui algoritm de planificare pentru înscrierea mingii în poartă.
- Adăugarea unor strategii complexe pentru a ocoli obstacolele și pentru a îmbunătăți comportamentul autonom al robotului.

În concluzie, proiectul nostru demonstrează aplicabilitatea tehnologiilor open-source în robotică și reprezintă un pas important în înțelegerea funcționării sistemelor autonome.

9. Bibliografie

[www 01] https://releases.ubuntu.com/jammy/
[www 02] https://emanual.robotis.com/docs/en/platform/turtlebot3/appendix_lds_01/
[www 03] https://emanual.robotis.com/docs/en/platform/turtlebot3/features/
[www 04] https://www.youtube.com/watch?v=Gg25GfA456o
[www 05] https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/#gazebo-simulation
[www 06] https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debs.html
[www 07] https://emanual.robotis.com/docs/en/platform/turtlebot3/quick-start/