

UNIVERSITATEA "POLITEHNICA" DIN TIMIȘOARA
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL AUTOMATICĂ ȘI INFORMATICĂ APLICATĂ

FOTBALIST

PROIECT SINCRETIC I

AUTORI: Dumitrescu Iasmina-Andreea
Jiva Iulia-Maria

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

CUPRINS

I. Introducere.....	1
II.Prezentarea temei.....	2
III.Tehnologii utilizate.....	3
IV.Ghidul programatorului.....	4
V.Ghidul utilizatorului.....	6
VI.Testare și punere în funcțiune.....	9
VII.Prezentarea firmei.....	12
VII.Concluzii.....	13
IX.Bibliografie.....	14

I. Introducere

În ultimele decenii, dezvoltarea tehnologiei a facilitat o expansiune remarcabilă în domeniul roboților mobili și al sistemelor de conducere la distanță. Roboții mobili reprezintă entități mecanice sau virtuale capabile să se deplaseze autonom sau la comandă, cu aplicabilități variate în industrie, cercetare sau chiar în viața de zi cu zi. Aceștia sunt proiectați să opereze fie autonom, fie sub control uman la distanță, în funcție de aplicație și de nivelul de complexitate al sarcinii. Printre exemplele populare se numără vehiculele autonome, dronele, roboții de curățenie și roboții utilizați pentru inspecții în zone periculoase. Conducerea la distanță a roboților mobili se referă la capacitatea de a controla și monitoriza roboții de la distanță, folosind tehnologii de comunicație avansate precum tehnologiile de comunicații fără fir, cum ar fi Wi-Fi, 5G sau rețelele prin satelit.

Dezvoltarea roboților mobili și a tehnologiilor asociate cu conducerea la distanță este impulsionată de necesitatea unor soluții mai eficiente și mai sigure pentru diverse provocări ale societății moderne. Acești roboți sunt folosiți pentru a înlocui oamenii în sarcini repetitive, periculoase sau care necesită precizie ridicată. În același timp, conducerea la distanță aduce un nivel suplimentar de control și adaptabilitate, fiind o punte între capacitatea de operare autonomă a unui robot și deciziile strategice luate de operatorii umani.

Perspectivile viitoare includ integrarea mai avansată a inteligenței artificiale, utilizarea roboților mobili în rețele interconectate (Internet of Things) și creșterea autonomiei prin învățare automată. Conducerea la distanță se va îmbunătăți odată cu dezvoltarea tehnologiilor de comunicație și a interfețelor om-mașină.



II. Prezentarea temei

Tema proiectului este reprezentată de simularea unui robot ce imită mișcarea și comportamentul unui fotbalist. Scopul simulării este de a înscrie mingea într-una dintre cele două porți aflate pe suprafața terenului. Cu ajutorul platformei de dezvoltare Gazebo, se simulează un teren de fotbal în timp real care conține robotul mobil destinat să participe la joc. Porțile de pe suprafața terenului sunt determinate de către 4 cuburi roșii. Mingea reprezintă o sferă de culoare albastră, poziționată aleator pe suprafața terenului.

Pentru o mai bună tehnică a simulării jocului, s-au folosit culori reprezentative și diferite față de suprafața de joc, astfel, procesul fiind mai eficient în detectarea obiectelor și în diferențierea elementelor din mediul virtual. Această abordare a redus erorile de detecție și a accelerat procesul de luare a deciziilor de către robot.

În prima etapă, robotul detectează mingea de culoare albastră și se deplasează spre aceasta. În cazul în care mingea nu a fost găsită, acesta parcurge o rotație de până la 360 de grade cu scopul de a o găsi. Astfel, dacă acțiunea este realizată cu succes, robotul intră în coliziune cu mingea și se deplasează împreună cu ea pentru a înscrie.

În timpul procesului de urmărire a mingii, robotul detectează mijlocul celor mai apropiate 2 porți pentru a dirija mingea în interiorul acestora. Porțile, fiind 2 cuburi de culoare roșie, asigură o performanță mai bună asupra robotului și determină mai ușor distanța dintre elemente, centrând mingea după algoritmul simplificat de acesta.

La final, robotul percepe condiția de oprire datorită unui element de culoare neagră, plasat în dreptul liniei porții, semn că mingea a intrat în poartă, deci robotul a înscris. Astfel, utilizarea culorilor distinctive a fost un factor cheie pentru îmbunătățirea acurateței simulării și a comportamentului autonom al robotului pe teren.



III. Tehnologii utilizate

Pentru a realiza proiectul s-a folosit mediul de simulare Gazebo. Gazebo este un mediu de simulare robotică avansat, utilizat pe scară largă pentru dezvoltarea, testarea și validarea algoritmilor și sistemelor robotice într-un mediu virtual controlat. Este parte a ecosistemului Robot Operating System (ROS) și oferă funcționalități pentru simularea senzorilor și a interacțiunilor robotului cu mediul înconjurător. Robotul folosit pentru simulare face parte din familia TurtleBot. Acesta este dotat cu senzori și un computer care permite utilizatorilor să testeze concepte de robotică, în cazul nostru l-am utilizat pentru a detecta automat mingea, cuburile roșii și de a împinge mingea între ele.

ROS2 (Robot Operating System 2) este o platformă open-source pentru dezvoltarea de aplicații robotice. Funcționează pe multiple platforme, inclusiv Linux, Windows și macOS, și poate fi utilizat pe dispozitive integrate cu resurse limitate. Include mecanisme de securitate integrate, cum ar fi criptarea și autentificarea comunicărilor, esențiale pentru aplicații industriale și autonome. ROS2 este adesea utilizat în robotică mobilă (drone, roboți mobili autonomi), aplicații medicale (roboți chirurgicali), sisteme industriale și automatizări.

ROS2 folosește protocolul DDS (Data Distribution Service), care permite comunicarea distribuită între noduri fără a depinde de un master centralizat. Nodurile sunt elementele de bază ale pentru ROS2, utilizate pentru a împărți funcționalitățile unui sistem robotic în componente mai mici și modulare. Un nod este o unitate executabilă care implementează o anumită funcționalitate, cum ar fi controlul motoarelor, procesarea datelor de la senzori sau navigația. Nodurile comunică între ele printr-o infrastructură distribuită, folosind mecanisme precum topice, servicii și acțiuni.

TurtleBot3 este un robot mobil open-source folosit frecvent în cercetare, educație și dezvoltare de proiecte de robotică. Este una dintre cele mai populare platforme de învățare Robot Operating System (ROS) datorită modularității și suportului larg al comunității.

Visual Studio Code (VS Code) este un editor de cod popular, folosit pe scară largă în dezvoltarea aplicațiilor de robotică, inclusiv pentru lucrul cu Python 3. Este potrivit pentru proiecte de robotică datorită integrării cu diverse extensii și biblioteci specifice domeniului.



IV. Ghidul programatorului

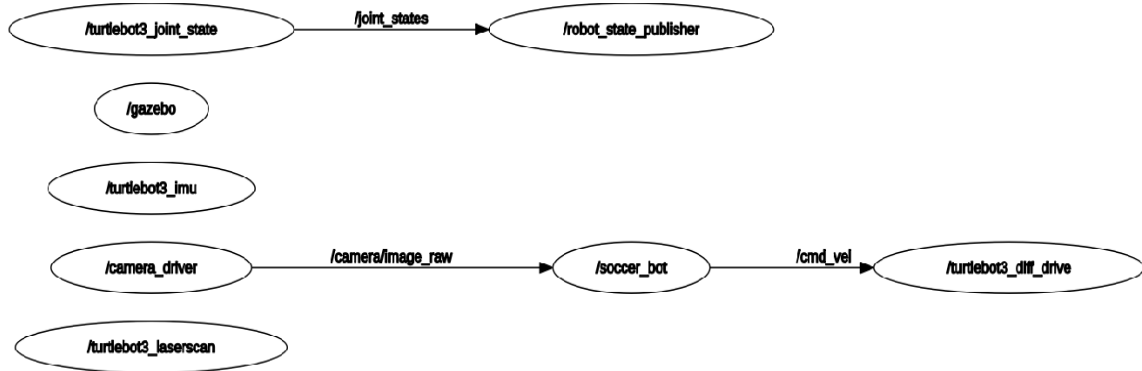
Aplicația conține un script cu un cod python care implementează un robot autonom care detectează și urmărește o minge albastră folosind o cameră, procesând imaginile pentru a calcula mișcările necesare. Robotul se rotește pentru a găsi mingea, o urmează și se oprește gradual odată ce detectează o linie neagră, indicând marcarea unui gol. Acțiunile robotului sunt coordonate prin publicarea de comenzi de mișcare pe topicul `/cmd_vel`.

Ca piesă de bază a codului există nodul SoccerBot care are următoarele funcțiuni:

- publică comenzi de mișcare pe topicul `/cmd_vel` folosind un obiect de tip Twist. Aceste comenzi sunt folosite pentru a controla robotul, de exemplu pentru rotirea sau deplasarea înainte.
- se abonează la topicul `/camera/image_raw` pentru a primi imagini brute de la camera robotului. Aceste imagini sunt procesate pentru a detecta mingea și linia neagră
- folosește imaginile pentru detectarea mingii albastre și pentru detectarea unei linii negre pentru a ști ca înscriș, fiind același timp și condiția de stop a robotului.
- în funcție de ce detectează senzorii, nodul decide dacă: se rotește pentru a căuta mingea, urmărește mingea pentru a o aduce spre linia neagră, se oprește dacă a atins linia neagră
- Utilizează `'self.get.logger().info()'` pentru a înregistra mesaje utile în terminal, precum `'Ball detected!'` și `'Goal scored!'`.



Pe lângă nodul SoccerBot, proiectul conține o întreagă ierarhie de noduri:



- /turtlebot3_joint_state este responsabil de gestionarea comunicării sau configurării inițiale între diferitele componente
- /gazebo gestionează simularea mediului 3D, inclusiv fizica, senzori și dinamica robotului.
- /turtlebot3_imu gestionează senzorul IMU (Inertial Measurement Unit). Furnizează date despre accelerație, orientare și viteză unghiulară.
- /camera_driver gestionează camera robotului. Captează imagini și le publică pe un topic
- /turtlebot3_laserscan gestionează senzorul LIDAR (laser) al robotului. Furnizează o scanare a mediului înconjurător pentru detectarea obstacolelor.
- /robot_state_publisher publică starea kinematică și pozițiile articulațiilor robotului pe baza unui model URDF. Ajută la actualizarea poziției robotului în sistemul de coordonate.
- /soccer_bot controlează comportamentul robotului de fotbal așa cum am descris anterior
- turtlebot3_diff_drive controlează motoarele diferențiale ale robotului TurtleBot3. Primește comenzi de viteză și le traduce în mișcări ale roților.



V. Ghidul utilizatorului

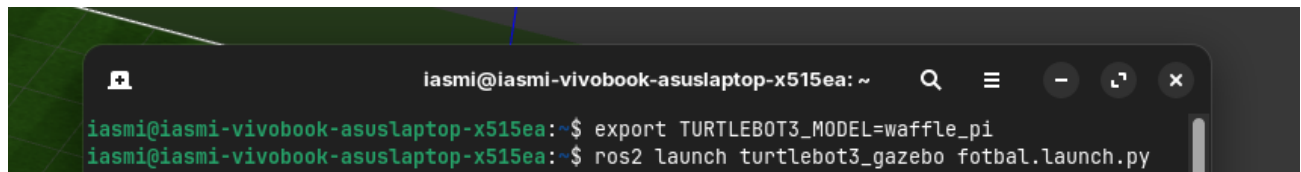
ROS2, în combinație cu platforma de simulare Gazebo, permite simularea și testarea roboților într-un mediu virtual realist.

Pentru instalarea mediului de dezvoltare ROS2, s-a folosit sistemul de operare Linux Ubuntu 22.04 LTS, unde s-au instalat atât pachetele necesare pentru python3, cât și pachetele necesare ROS2, distribuția Humble. Astfel, s-a realizat cu succes instalarea dependențelor care vor rula în momentul rulării codului.

Editorul de cod sursă, Visual Studio Code (VS Code), a fost integrat cu mediile de dezvoltare ROS2 și Gazebo ce facilitează scrierea codului, depanarea și gestionarea proiectului complex de robotică, cu ajutorul extensiilor necesare.

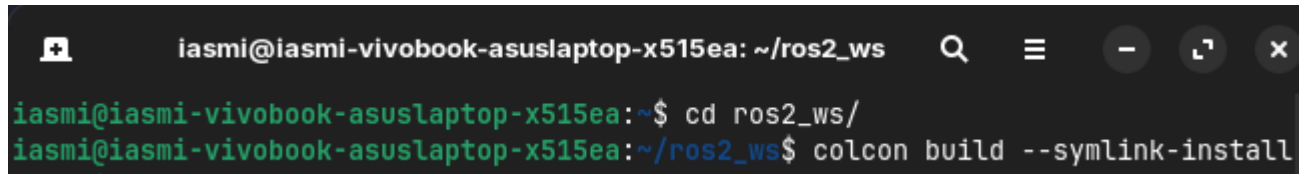
Pentru a porni aplicația și mediul de simulare, sunt necesari pașii de mai jos:

- Configurarea mediului Gazebo:
 - Pornirea simulării:

A terminal window with a dark background. The title bar shows 'iasmi@iasmi-vivobook-asuslaptop-x515ea: ~'. The prompt is 'iasmi@iasmi-vivobook-asuslaptop-x515ea:~\$'. The first command entered is 'export TURTLEBOT3_MODEL=waffle_pi'. The second command is 'ros2 launch turtlebot3_gazebo fotbal.launch.py'.

```
iasmi@iasmi-vivobook-asuslaptop-x515ea:~$ export TURTLEBOT3_MODEL=waffle_pi
iasmi@iasmi-vivobook-asuslaptop-x515ea:~$ ros2 launch turtlebot3_gazebo fotbal.launch.py
```

- Construirea codului (cu ajutorul mediului VS Code) folosind directorul ros2_ws/:

A terminal window with a dark background. The title bar shows 'iasmi@iasmi-vivobook-asuslaptop-x515ea: ~/ros2_ws'. The prompt is 'iasmi@iasmi-vivobook-asuslaptop-x515ea:~/ros2_ws\$'. The first command entered is 'cd ros2_ws/'. The second command is 'colcon build --symlink-install'.

```
iasmi@iasmi-vivobook-asuslaptop-x515ea:~/ros2_ws$ cd ros2_ws/
iasmi@iasmi-vivobook-asuslaptop-x515ea:~/ros2_ws$ colcon build --symlink-install
```

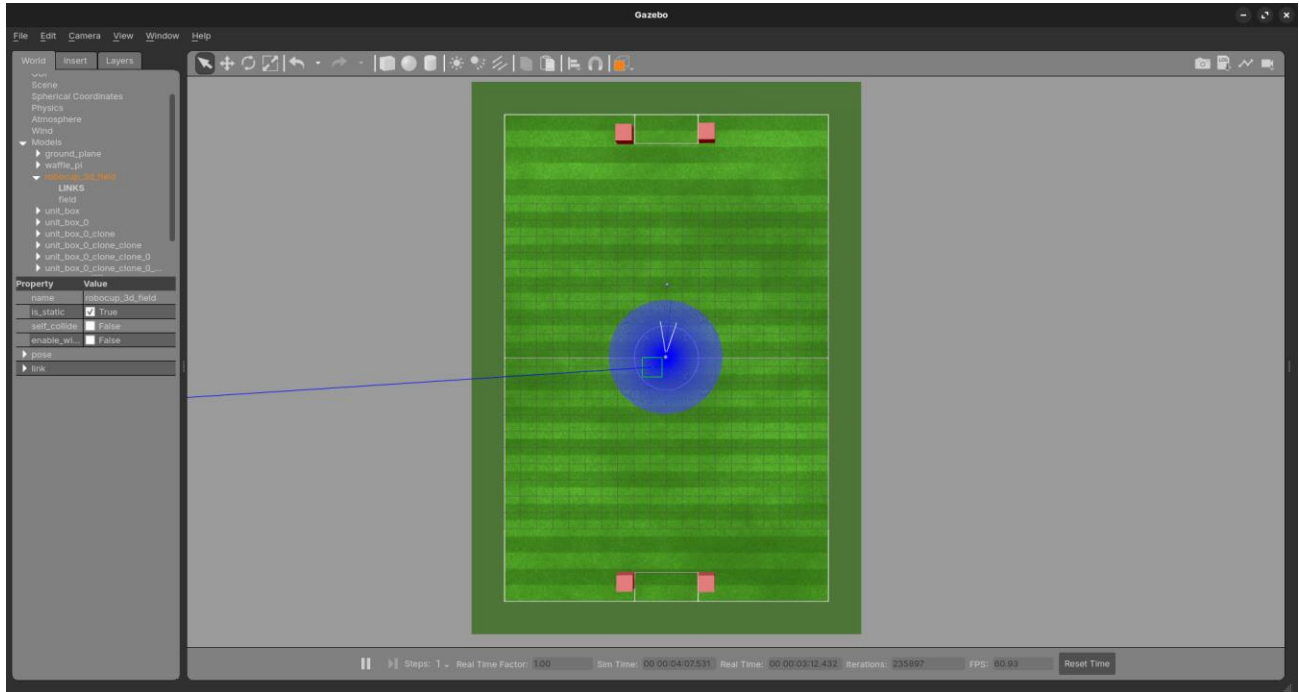


- Rularea codului cu ajutorul bash și a fișierului cod sursă:

```
Finished <<< proiectfotbalist [1.44s]

Summary: 5 packages finished [1.69s]
iasmi@iasmi-vivobook-asuslaptop-x515ea:~/ros2_ws$ . install/setup.bash
iasmi@iasmi-vivobook-asuslaptop-x515ea:~/ros2_ws$ ros2 run proiectfotbalist node
```

Astfel, simularea ar trebui să fie prezentă pe ecran în felul următor:



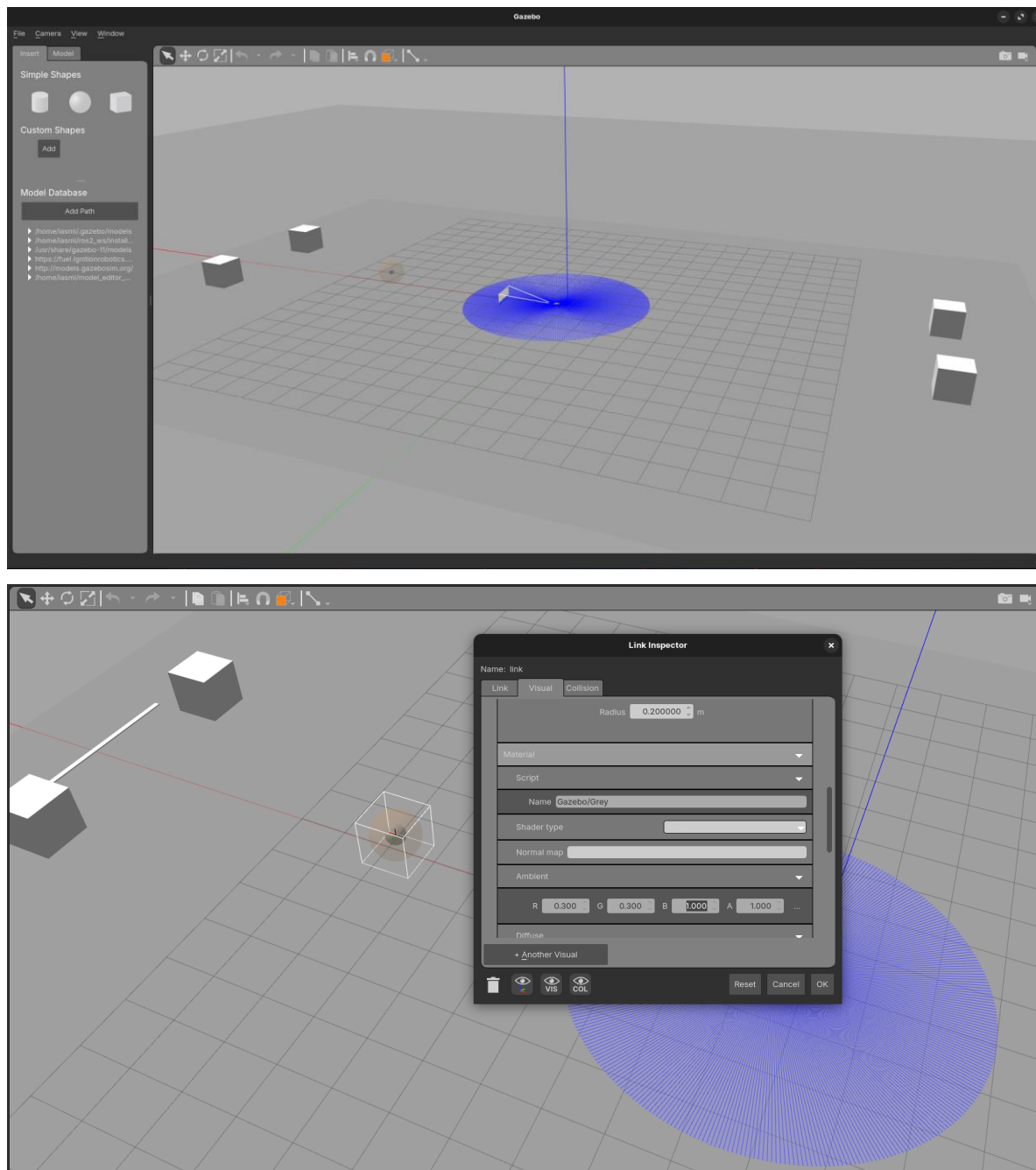
După cum se observă, simularea s-a pornit cu robotul plasat în mijlocul terenului, iar mingea, într-un loc aleator. Porțile sunt conturate cu culoarea roșie, fapt ce sugerează perimetrul de înscriere a robotului.

Rularea simulării se determină și cu ajutorul butonului de începere, plasat pe interfața mediului de dezvoltare Gazebo. Astfel, robotul își începe parcursul determinării mingii. Odată detectată mingea, robotul continuă să simuleze mișcarea de fotbalist și încearcă să centreze mingea între cele mai apropiate 2 cuburi roșii din poziția în care se află. În final, simularea se oprește în momentul de înscriere.



**Proiect sincretic I – Fotbalist
an universitar 2024 / 2025**

Pentru configurarea grafică a mediului de dezvoltare Gazebo, s-au folosit diferite structuri pe interfața simulării. Modelarea lor, cum ar fi elementele de culoare, de suprafață sau chiar de poziționare, s-a efectuat local, în mediul Gazebo, astfel:



VI. Testare și punere în funcțiune

Înainte de toate, pașii următori reprezintă instalarea tuturor configurațiilor și pachetelor necesare dezvoltării proiectului. Instalarea constă în rularea următoarelor comenzi de terminal:

- Pachetele pentru Python3

```
sudo apt update && sudo apt install -y \  
python3-flake8-docstrings \  
python3-pip \  
python3-pytest-cov \  
ros-dev-tools
```

```
sudo apt install -y \  
python3-flake8-blind-except \  
python3-flake8-builtins \  
python3-flake8-class-newline \  
python3-flake8-comprehensions \  
python3-flake8-deprecated \  
python3-flake8-import-order \  
python3-flake8-quotes \  
python3-pytest-repeat \  
python3-pytest-rerunfailures
```

- ROS2

```
sudo apt install ros-humble-desktop
```

```
# Replace ".bash" with your shell if you're not using bash  
# Possible values are: setup.bash, setup.sh, setup.zsh  
source /opt/ros/humble/setup.bash
```



- Mediul de dezvoltare Gazebo:

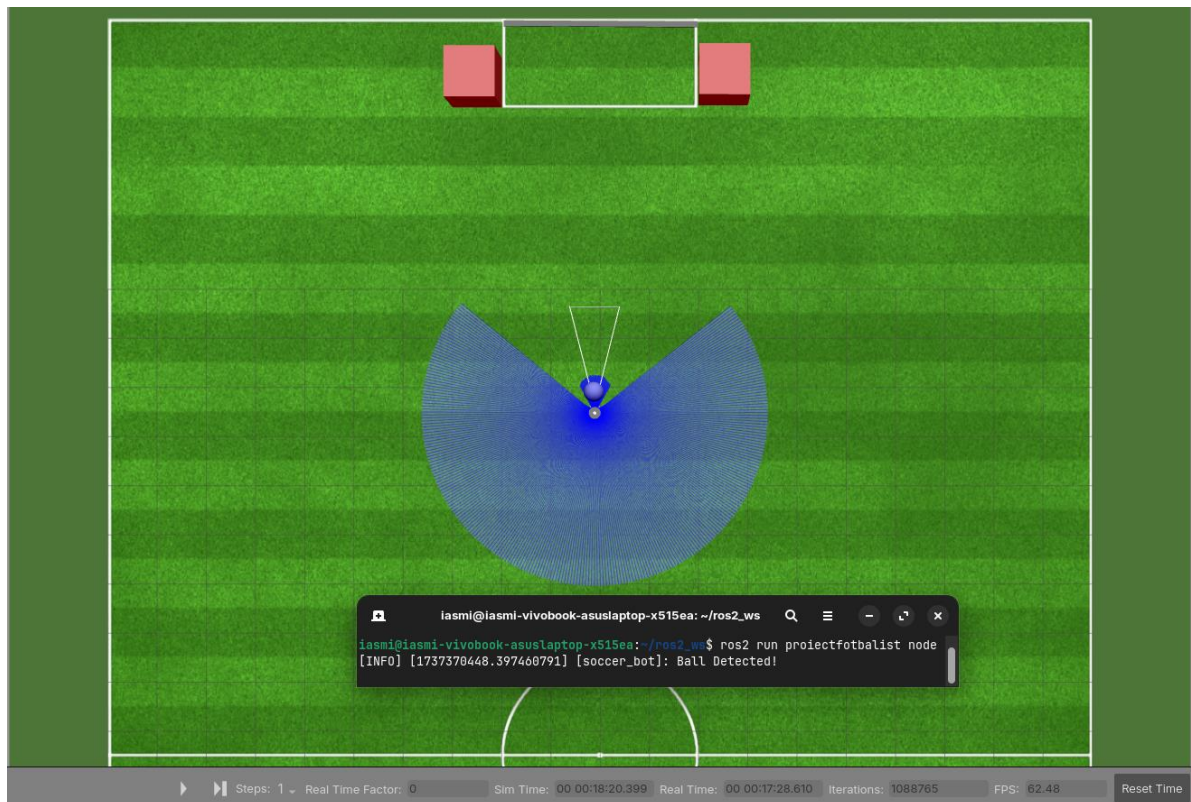
```
$ sudo apt install ros-humble-gazebo-*
```

- TurtleBot3 Simulation Package:

```
$ cd ~/turtlebot3_ws/src/
$ git clone -b humble https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
$ cd ~/turtlebot3_ws && colcon build --symlink-install
```

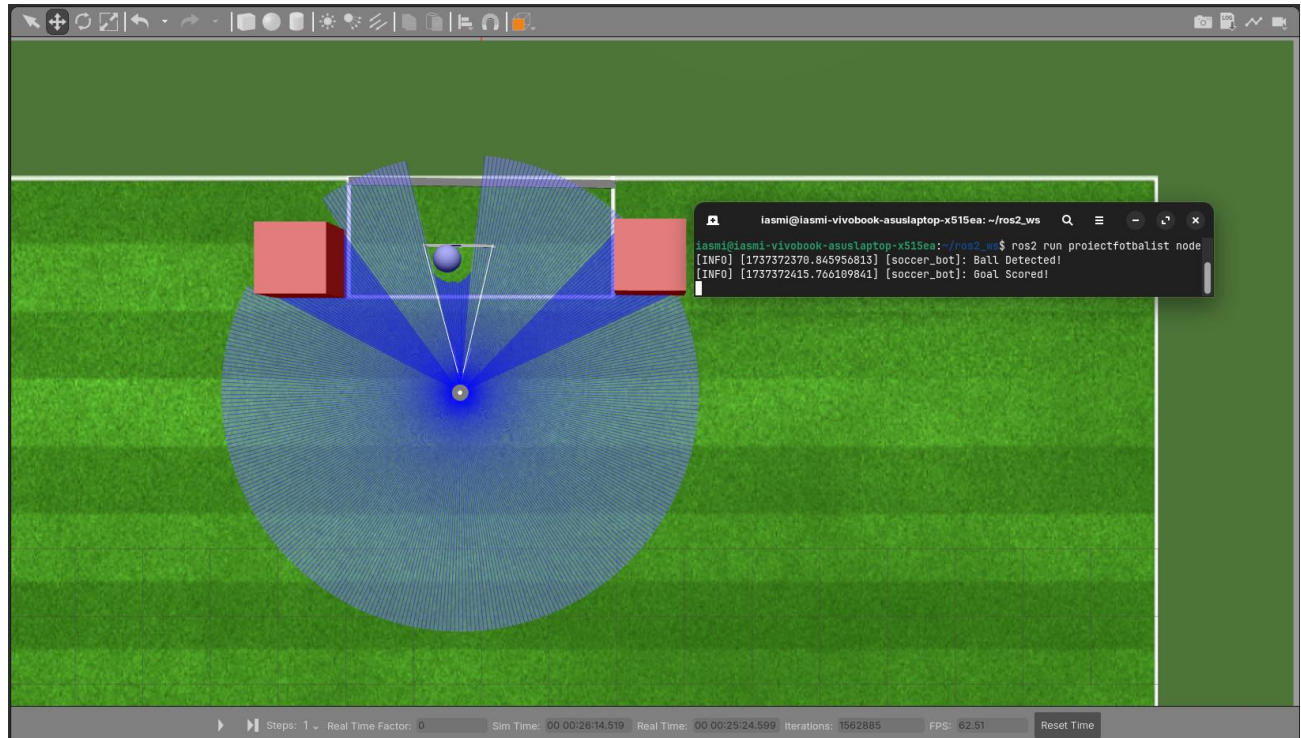
Pe partea de testare a aplicației s-a poziționat robotul în diferite unghiuri ale mingii, acesta împingând-o între cuburi doar dacă este aliniat cu ele. De asemenea, are funcționalitatea de a rămâne pe teren. Dificultățile întâmpinate se referă la direcționarea robotului într-un unghi astfel încât să împingă mingea corect din orice parte a terenului, acest lucru putând fi îmbunătățit la proiect.

Odată cu începerea simulării, robotul este poziționat pe suprafața terenului, astfel:



Robotul începe activitatea prin detectarea mingii de culoare albastră. În caz contrar, acesta se va roti în același loc până la 360 de grade, fapt ce sugerează ca mingea va fi cu siguranță găsită. În următoarea etapă, robotul se deplasează către minge și va încerca o aliniere cu aceasta, calculând în același timp și mijlocul distanței dintre cele 2 cuburi roșii ce sugerează perimetrul porții.

Odată găsit mijlocul, robotul preia mingea și o împinge către poartă, astfel, simularea se va opri în momentul înscrierii mingii.



Pentru condiția de încheiere a simulării, s-a folosit un obiect de culoare neagră ce determină faptul că mingea a ajuns cu succes în poartă, afișându-se în consolă mesajul ‘Goal scored!’.



VII. Prezentarea firmei

Echipa OpenGoal Robotics, echipa care s-a ocupat de realizarea proiectului *Fotbalist* este compusă din doi membri: Dumitrescu Iasmina-Andreea și Jiva Iulia-Maria. Fiecare membru a avut sarcini bine definite de îndeplinit în cadrul proiectului.

Iasmina s-a ocupat de partea tehnică, ea având deja sistemul de operare Linux instalat a putut ușor și eficient să instaleze pachetul ROS2 și simulatorul Gazebo pentru dezvoltarea proiectului. Ambii membri au contribuit cu idei pentru structuri și design-ul a lumii din simulator, idei ce implica așezarea porții, robotului, câmpului de joc. Pentru partea de logică a codului s-a contribuit la comun, dar partea de scriere și rulare efectivă în simulator a fost realizată de Iasmina.

La documentație s-a lucrat pe un document Word în Google Drive, astfel putând lucra ambii membri în același timp, iar modificările se pot vedea în timp real. Prezentarea Powerpoint a proiectului, un rezumat al acestuia, folosită la susținerea finală, a fost realizată de Iasmina.



VIII. Concluzii

Per ansamblu, acest proiect reprezintă un exemplu al aplicabilității tehnologiilor avansate de robotică mobilă și control la distanță. Prin integrarea ROS2 (Robot Operating System 2), senzorilor vizuali și unui algoritm simplu de luare a deciziilor, robotul *Fotbalist* demonstrează cum sistemele autonome pot fi programate să interpreteze mediul înconjurător și să acționeze în mod logic pentru atingerea unui scop bine definit – în acest caz, detectarea, urmărirea unei mingi și marcarea unui gol. Cu îmbunătățiri, cum ar fi integrarea unor algoritmi de inteligență artificială pentru recunoașterea avansată a obiectelor sau implementarea hărților 3D pentru navigație, acest proiect poate fi extins pentru a aborda scenarii mai complexe.

Robotul este construit pentru a reacționa la stimuli vizuali și pentru a interacționa cu mediul său. Prin utilizarea camerei și procesării imaginii, robotul identifică obiecte-cheie (mingea albastră și linia neagră) și își ajustează mișcările în funcție de poziția acestora. Acest tip de autonomie este un exemplu pentru aplicațiile în care interacțiunea cu mediul este dinamică și imprevizibilă. Dezvoltarea unor astfel de roboți are aplicații largi, inclusiv în domenii precum logistica, agricultura și intervențiile de salvare.

Proiectul "*Fotbalist*" este un exemplu relevant al roboților mobili și controlului la distanță, demonstrând potențialul acestor tehnologii în diverse aplicații. Dezvoltarea continuă în acest domeniu are capacitatea de a revoluționa industriile moderne și de a oferi soluții inovatoare pentru provocările globale.



IX. Bibliografie

[www 01] <https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/>

[www 02] <https://docs.ros.org/en/humble/Installation.html>

