 **UNIVERSITATEA “POLITEHNICA” DIN TIMISOARA**

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**DEPARTAMENTUL AUTOMATICĂ ŞI INFORMATICĂ APLICATĂ**

**Lewanbotski**

**STUDENT : Alexandru-Daniel FLOREA**

**Coordonatori:**

**Conf.dr.ing. Florin DRĂGAN**

**As. ing. Emil VOIŞAN**

Anul III AIA - an universitar 2021 / 2022

**CUPRINS**

Coperta...................................................................................................................1

Cuprins...................................................................................................................2

1. Introducere...................................................................................................3
2. Prezentarea temei.........................................................................................3
3. Tehnologii utilizate......................................................................................4
4. Ghidul programatorului................................................................................8
5. Ghidul utilizatorului.....................................................................................12
6. Testare și punere în funcțiune......................................................................15
7. Concluzii......................................................................................................15
8. Bibliografie...................................................................................................15
9. INTRODUCERE

Se numeşte proces de conducere un proces dinamic de organizare şi coordonare de către cineva (ceva), într-o anumită perioadă de timp, a altor grupuri de membri ai sistemului, în scopul realizării unor sarcini sau scopuri specifice.

Conducerea în sistemele tehnice se numeşte acţiunea intenţionată asupra obiectului reglat.

Robotul mobil este un sistem complex care poate efectua diferite activităţi într-ovarietate de situaţii specifice lumii reale. El este o combinaţie de dispozitive echipate cu servomotoare şi senzori (aflate sub controlul unui sistem ierarhic de calcul) ce operează într-un spaţiu real, marcat de o serie de proprietăţi fizice (de exemplu gravitaţia care influenţează mişcarea tuturor roboţilor care funcţionează pe pământ) şi care trebuie să planifice mişcările astfel încât robotul să poată realiza o sarcină în funcţie de starea iniţială a sistemului şi în funcţie de informaţia apriori existentă, legată de mediul de lucru.

Utilizările pentru care au fost, sunt şi vor fi concepuţi roboţii mobili sunt dintre cele mai diverse şi nu pot fi epuizate într-un spaţiu atât de restrâns. Mulţi roboţi din zona micro îşi găsesc utilizarea în medicină, fiind capabili să se deplaseze de-a lungul vaselor şi tuburilor corpului omenesc, în scopul investigaţiilor, intervenţiilor chirurgicale, dozării şi distribuirii de medicamente etc.

1. PREZENTAREA TEMEI

Tema proiectului este Fotbalist, în care obiectivul principal al robotului TurtleBot3 este de a înscrie gol în poarta. Scopul lui este de a observa mingea, iar mai apoi acesta va trebui sa o împinga spre poarta pentru a marca. Acesta își va relua pozția inițiala după execuția instrucțiunilor.

1. TEHNOLOGII UTILIZATE
2. TurtleBot

TurtleBot3 este un robot mobil ROS de dimensiuni mici, accesibil ca preț și programabil pentru scopuri în educație, cercetare, hobby și creare de prototipuri de produse. Scopul principal urmărit de TurtleBot3 este de a reduce în mod dramatic măsurile platformei și scăderea prețului fără a sacrifica funcționalitatea și calitatea produsului, și totodată oferind și expandabilitate. TurtleBot3 poate fi personalizat în moduri numeroase care depind de modul în care este reconstruită partea mecanică și gestionarea parților opționale cum ar fi senzorul. în plus, TurtleBot3 este dezvoltat cu SBC rentabil și de dimensiuni mici, care este potrivit pentru un sistem încorporat robust, senzor de distanta de 360 de grade și tehnologie de imprimare 3D.

TurtleBot3 este alcătuit din plăci modulare cărora utilizatorii le pot personaliza forma. Este disponibil în trei marimi: Burger mărime mica și Waffle, Waffle Pi mărime medie.

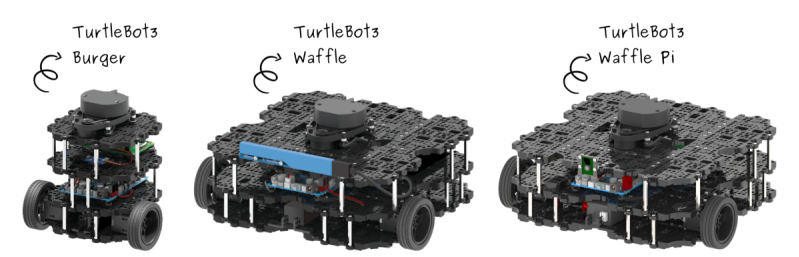


Figura 3.1 Tipuri de roboți TurtleBot3

TurtleBot3 consistă din baza, două motoare Dynamixel, o baterie de 1,800mAh, un LIDAR de 360 de grade, o cameră(+RealSense Camera pentru Waffle kit, +Raspberry Pi Camera pentru Waffle Pi kit), un SBC(single board computer: Raspberry Pi 3 și Intel Joule 570x) și un kit de montare hardware care atașează tot împreună și adaugă mai apoi senzori.

Tehnologia de bază a TurtleBot3 este SLAM, Navigare si Manipulare, ceea ce îl face potrivit pentru roboții de serviciu la domiciliu. TurtleBot poate rula algoritmi SLAM(localizare și mapare simultană) pentru a construi o hartă și mai apoi pentru a naviga prin camera. De asemena, acesta poate fi controlat de la distanță de pe un laptop, joypad sau un smart phone cu Android. TurtleBot poate de asemenea urmări picioarele unei persoane care se plimbă într-o cameră.

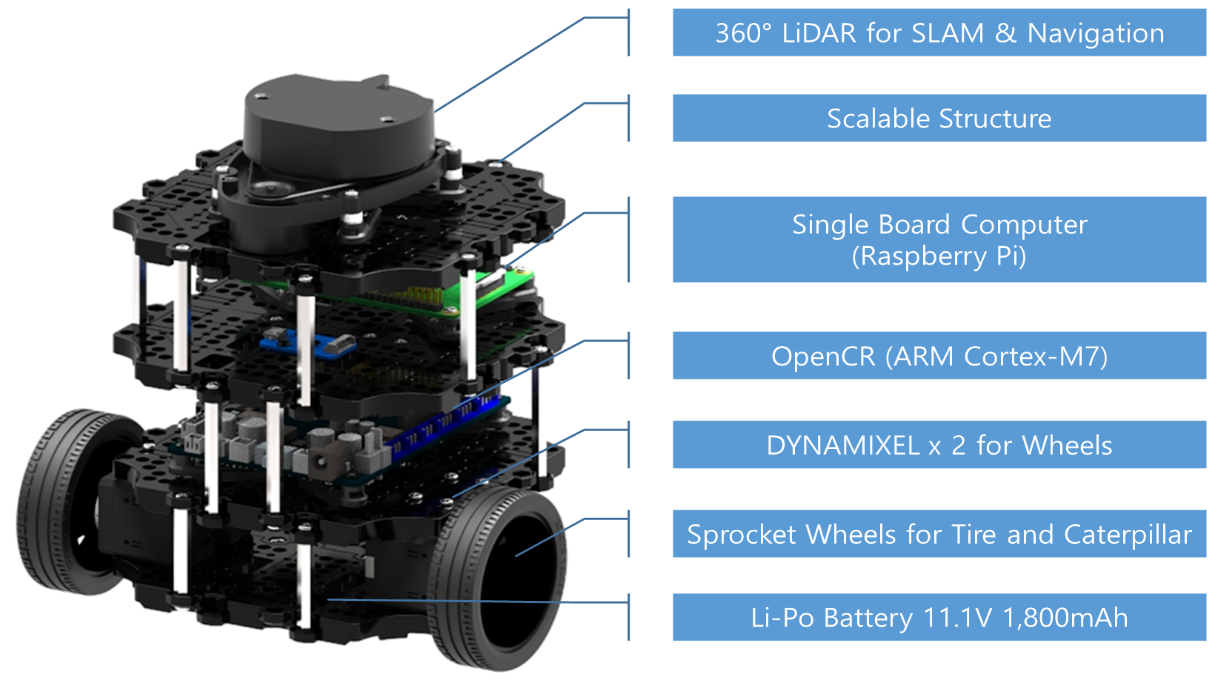


Figura 3.2 Alcătuirea unui robot TurtleBot

1. ROS

ROS(Robot Operating System) faciliteaza librării și unelte pentru a ajuta programatorii software să creeze aplicații cu roboți. Acesta oferă abstracție hardware, librării, vizualizare, transmitere de mesaje, management de pachete și multe altele.



Figura 3.3 Logo ROS

ROS Master oferă servicii de denumire și înregistrare pentru restul nodurilor din sistemul ROS. Acesta urmărește publisher și subscriberii la topic-uri și servicii. Rolul Masterului este de a permite nodurilor individuale ROS să se localizeze unul pe celălalt. Odată ce aceste noduri s-au localizat unul pe celălalt, ele comunică între ele peer-to-peer.

Un nod este fișier executabil într-un pachet ROS, care folosește o bibliotecă client ROS pentru a comunica cu alte noduri. Nodurile pot publica sau se pot abona la un topic. De asemenea, nodurile pot furniza sau utiliza un serviciu.

Topic-urile facilitează comunicarea între noduri. În general, nodurile nu știu cu cine comunică, dar cele care sunt interesate de date fac subscribe la topic-ul interesat. Un topic poate avea multiplii pulisheri și subscriberi.

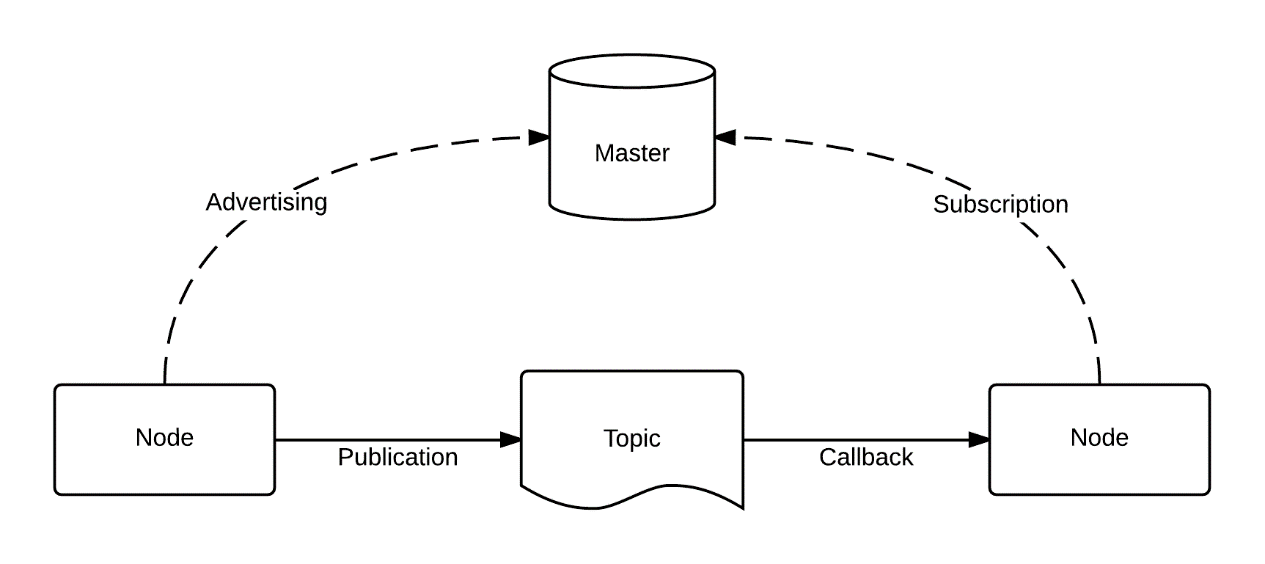


Figura 3.4 Schema de comunicare

1. Gazebo

Gazebo este un simulator de robotică 3D open-source. Gazebo a fost o componentă a Proiectului Player din 2004 pana in 2011. Gazebo a integrat motorul fizic ODE, redarea OpenGL și codul de suport pentru simularea senzorilor și controlul actuatorului. În 2011, Gazebo a devenit un proiect independent susținut de Willow Garage.

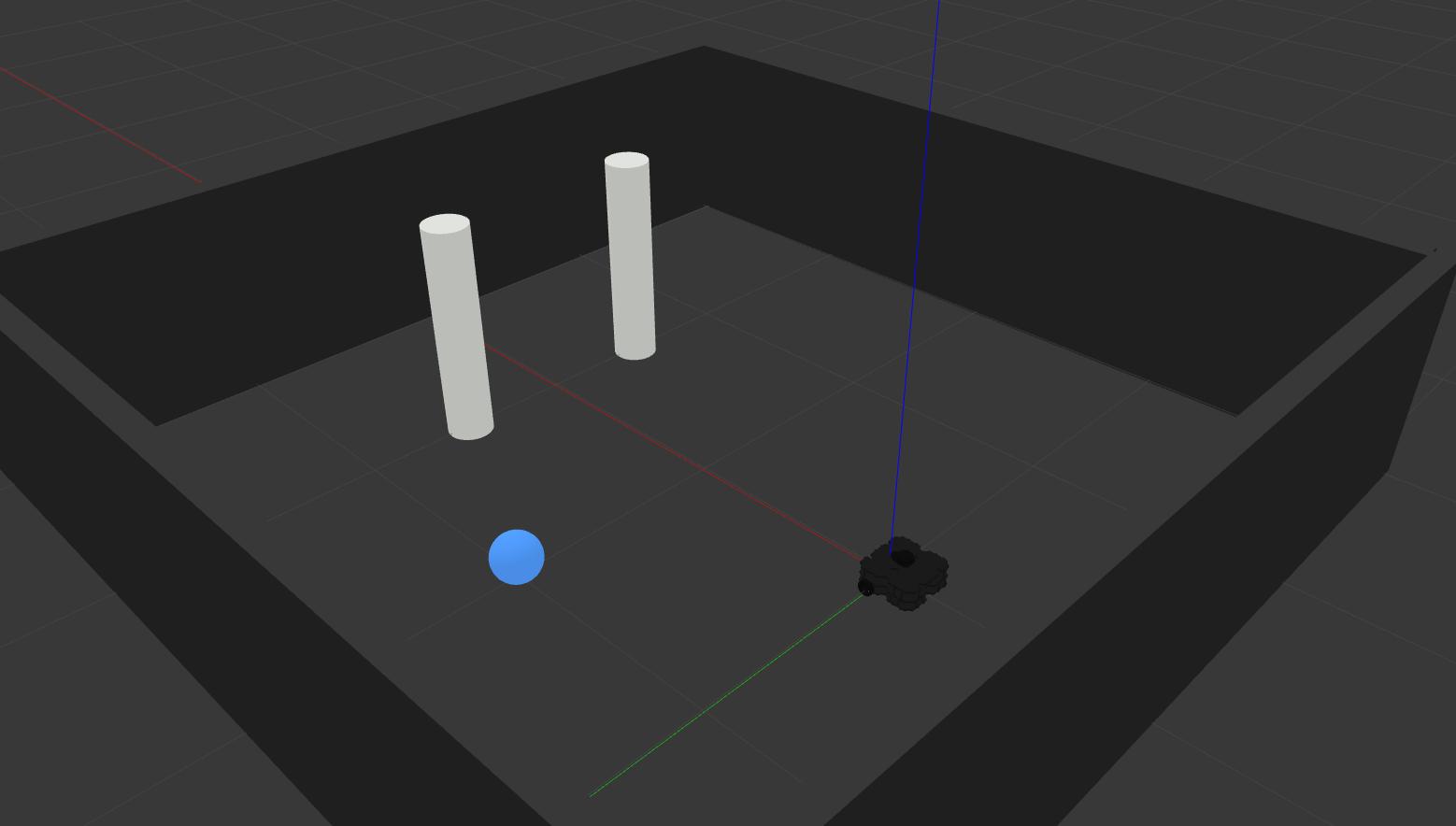
Gazebo poate folosi mai multe motoare fizice de înaltă performanța, cum ar fi ODE, Bullet (default fiind ODE). Oferă o redare realistă a mediului, inclusiv iluminare de înalta calitate, umbre și texturi. Poate modela senzori care pot "vedea" mediul simulat, cum ar fi telemetrie laser, camere(inclusiv cu wide angle), senzori în stil Kinect.

Figura 3.5 Exemplu din mediul de simulare Gazebo

1. Python

Python este un limbaj util de învățat, deoarece este utilizat pe scară largă în informatică și machine learning. Python este un limbaj folosit cu Raspberry Pi. Acest lucru îl face extrem de relevant pentru robotică, deoarece Raspberry Pi poate fi folosit pentru a controla un robot.

1. Visual Studio Code

Visual Studio Code este un editor de cod care oferă suport pentru numeroase limbaje de programare printre care Python, Java, C++, JavaScript și multe altele.

1. GHIDUL PROGRAMATORULUI

Aplicația s-a dezvoltat cu ajutorul mediilor de programare: Visual Studio Code, Gazebo Simulator, TurtleBot WafflePi, ROS. Mediul de simulare este alcătuit din 4 pereți cu latura de 4m care alcătuiesc un pătrat, 2 cilindrii paraleli situați la o distanță de 1m unul față de celălalt care alcătuiesc poarta și o sferă de culoarea albastră cu o masa setată care poate fi poziționată oriunde înaintea porții.

Implementarea și pașii parcurși de robot sunt următorii:

1. Robotul pornește tot timpul din punctul de coordonate (0,0).
2. La startul aplicatiei robotul incepe sa se roteasca la stanga pana cand va aparea in imagine culoarea albastra, moment in care acesta incearca sa se centreze pe minge.
3. Cand e pozitionat pe minge, atunci acesta calculeaza si retine pozitia mingii pe baza distantei si unghiului facut de robot fata de axa ox.
4. Dupa ce calculeaza coordonatele mingii, va calcula un punct in spatele ei cu 0.5 m pe directia portii. Centrul portii este punctul de coordonate (2,0)
5. Dupa ce robotul ajunge in spatele mingii, se reia primul pas, acela de a gasi mingea si de a se centra pe ea. De aici, robotul poate sa dribleze mingea pana in poarta sau sa lanseze un sut.
6. In functie de ce alegem, robotul se va intoarce in punctul de plecare dupa ce duce mingea in poarta sau dupa ce da sutul.

Start

Cautare minge

Calculare coordonate minge

Poziționare robot in spatele mingii

Revenire la poziția inițială

Loveste mingea spre poarta

Duce mingea in poarta

Probleme întâmpinate și modul lor de rezolvare:

1. Gestionarea unui program cu mai mulți subscriberi

* soluție în link-ul [www 05]

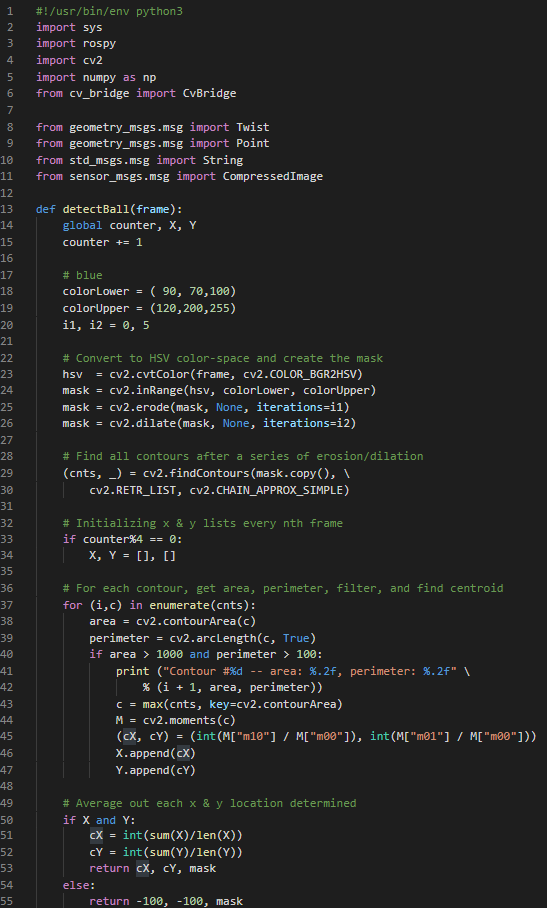
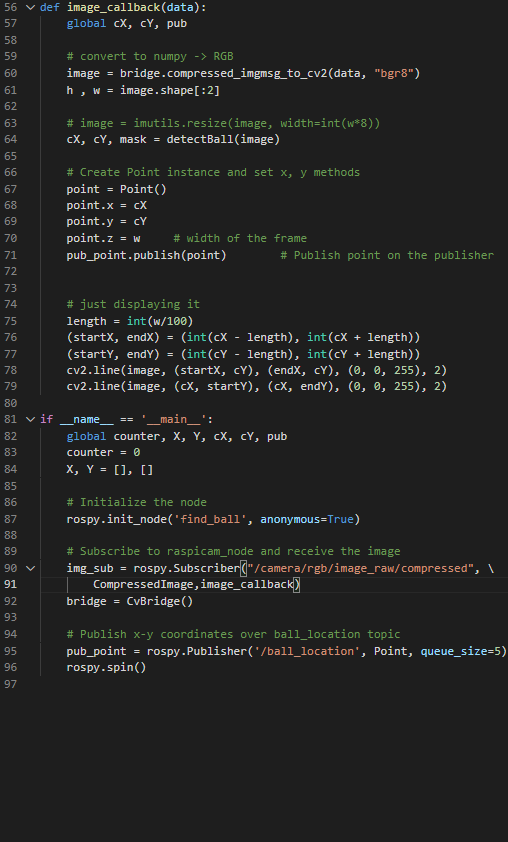
1. Calcularea poziției mingii

* soluție în link-rile [www 07] si [www 08]

1. Stabilirea unor proprietăți optime ale mingii

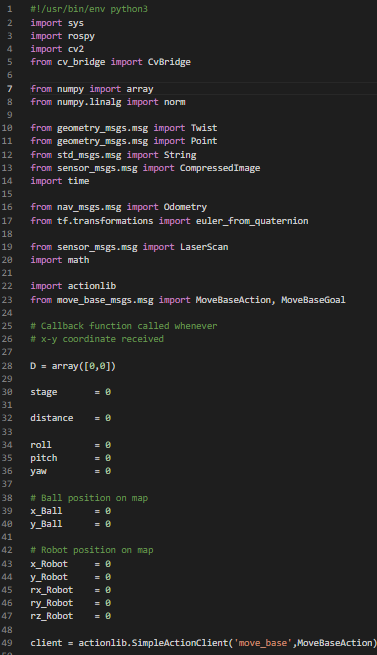
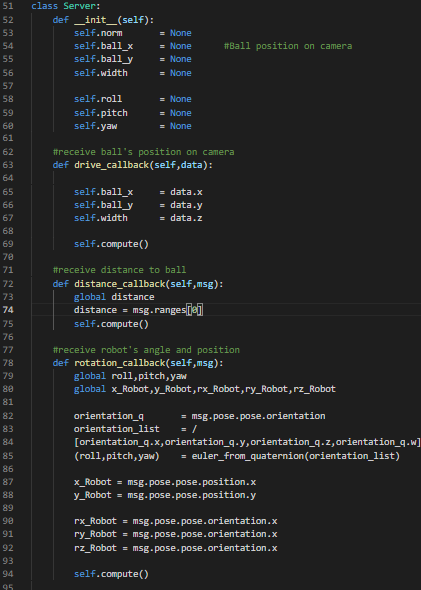
Codul de mai jos este cel din fisierul ball.py .

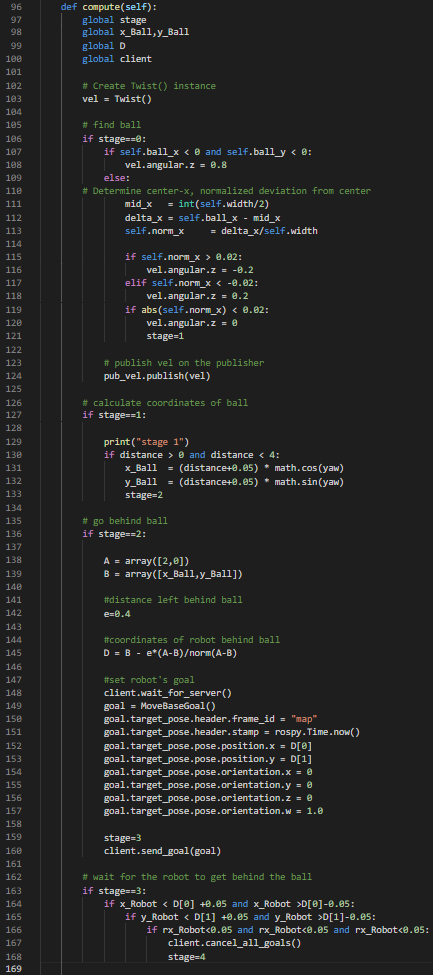
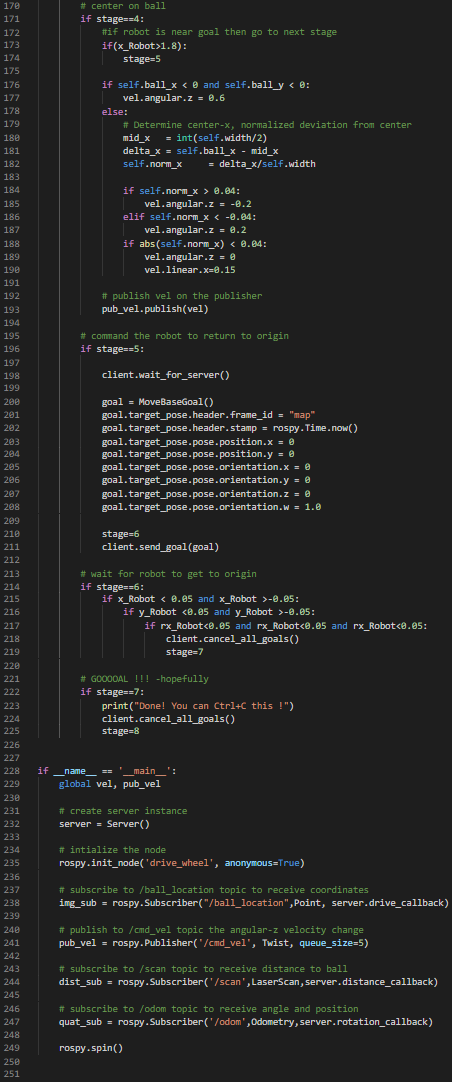
In acesta se face procesarea imaginii primate de la camera Raspberry Pi si publicarea pozitiei mingii in imagine. La acest publisher ne vom subscrie din fisierele dribble.py sau shoot.py .



Codul de mai jos este cel din dribble.py . Diferenta dintre acesta si shoot.py este destul de minca. In shoot.py avem un stage nou (4.5) pentru a comanda robotului sa mearga inainte pana la pozitia mingii, lucru care in dribble.py se facea in stage 4. In dribble.py stage 4 dureaza pana cand mingea ajunge in poarta pe cand in shoot.py se iese din stage 4 odata cu centrarea pe minge.

In dribble.py/shoot.py avem clasa Server. Aceasta contine functii care preiau si interpreteaza datele de la subscriberii din main. O parte importanta din clasa Server este functia compute, unde comandam prin mai multe stage-uri modul de functionare al robotului. Aceasta se executa la finalul fiecarei functii \_callback din clasa.





1. GHIDUL UTILIZATORULUI

Pașii parcurși pentru a rula aplicația sunt următorii:

1. Adăugăm fisierele ball.py, dribble.py și shoot.py intr-un catkin package
2. Adăugăm fotbal1.world si fotbal1.launch în catkin\_ws/src/turtlebot3\_simulations/ turtlebot3\_gazebo/worlds respectiv /launch
3. Adaugam map3.yaml si map3.png in /home
4. Deschidem un terminal și lansăm simulatorul Gazebo cu instrucțiunea roslaunch turtlebot3\_gazebo fotbal1.launch și poziționăm mingea (la cel putin jumatate de metru de vreun perete).
5. Lansăm harta într-un terminal nou folosind instructiunea roslaunch turtlebot3\_navigation turtlebot3\_navigation.launch map\_file:=$HOME/map3.yaml.
6. Navigăm la locul unde am salvat codul, deschidem 2 terminale noi, în primul vom rula instrucțiunea python3 ball.py și in al doilea python3 dribble.py sau python3 shoot.py după care se va rula aplicația.

În Figura 6.1 putem observa simulatorul Gazebo lansat cu instrucțiunea aferentă de la pasul 1 și de asemenea, harta cu instrucțiunea corespunzătoare în terminalul deasupra ei.

În Figura 6.2 sunt parcurse instrucțiunile de la pasul 3 în care este dată calea către codul aplicației.

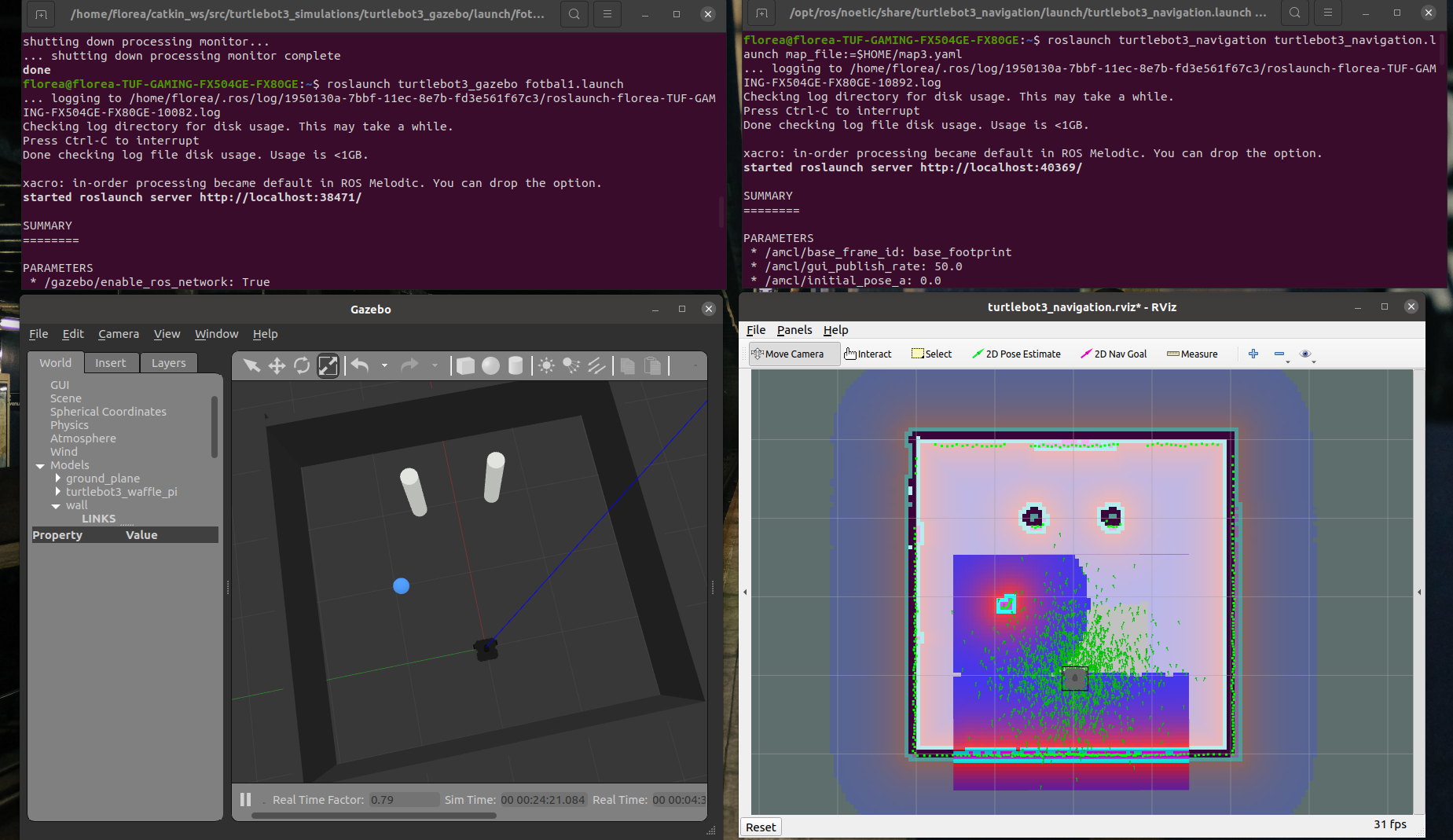
Iar în Figura 6.3 se poate vedea aplicația în rulare, robotul își părăsește punctul inițial și se îndreaptă către minge, iar pe hartă este evidențiata traiectoria pe care robotul va urma să o parcurgă pentru a ajunge la minge.

Figura 6.1 Cerințele de la Pasul 1 și 2

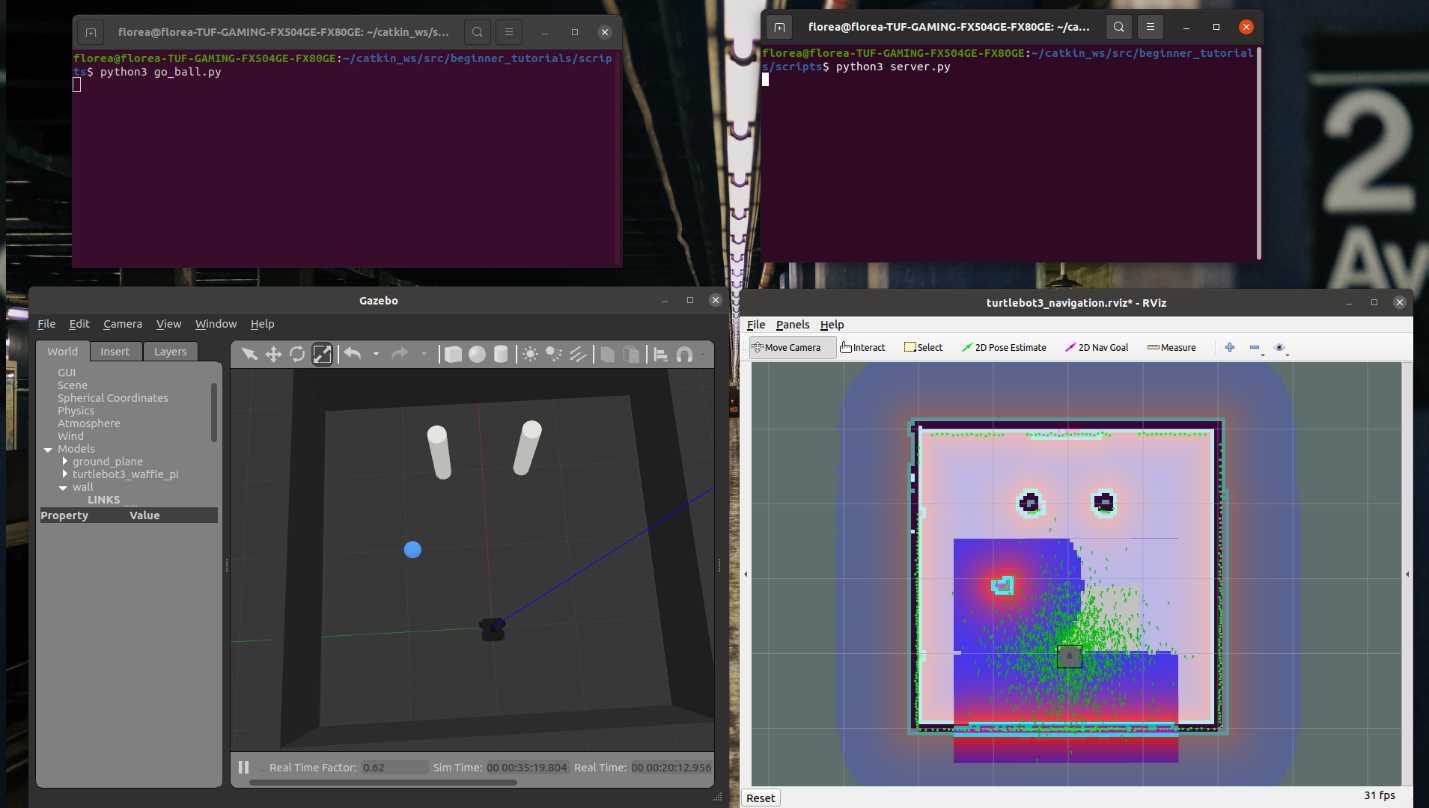
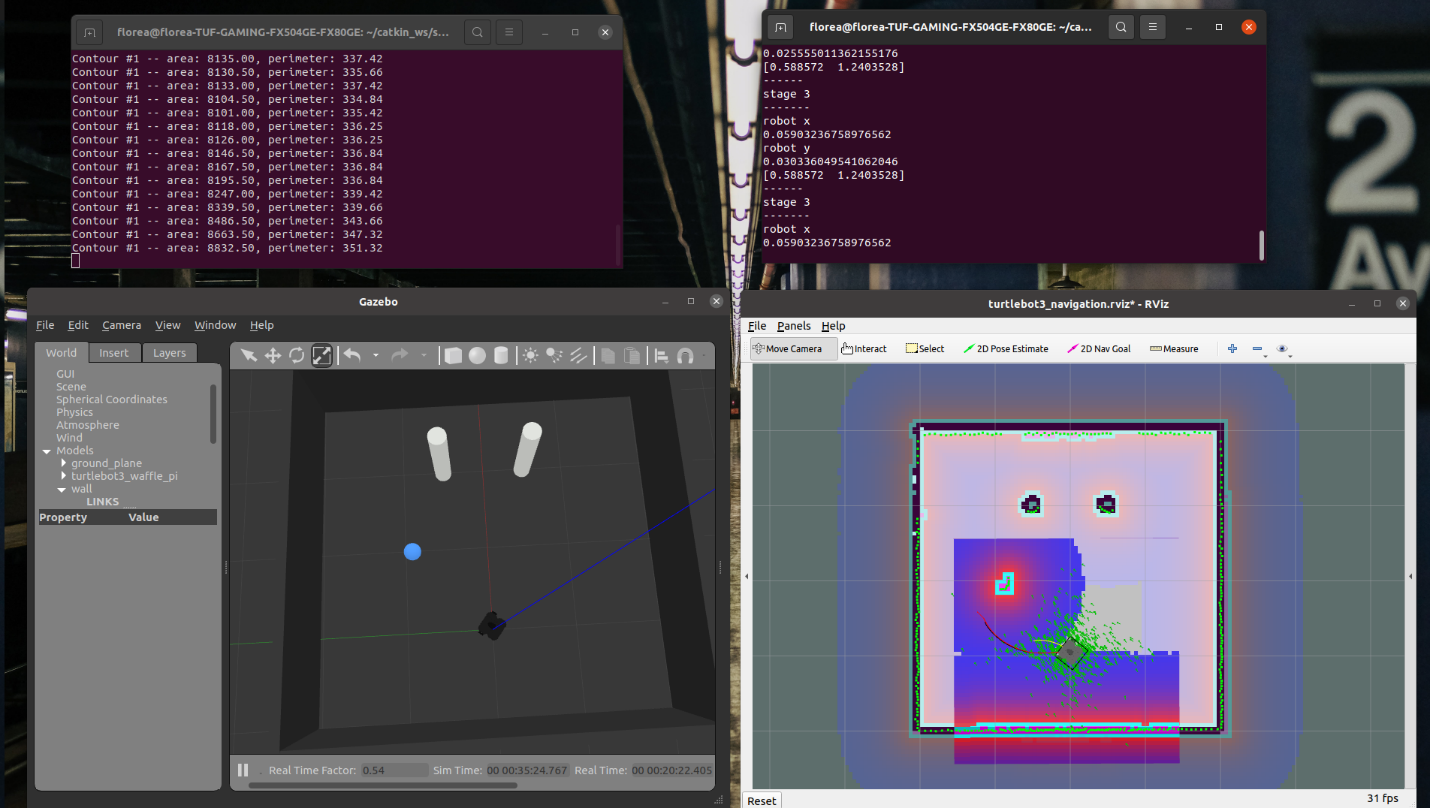
Figura 6.2 Cerințele de la Pasul 3

Figura 6.3 Demonstrație

1. TESTARE ŞI PUNERE ÎN FUNCŢIUNE

Punerea în funcțiune a aplicației s-a făcut pe baza instrucțiunilor de la punctul anterior, iar testarea s-a făcut pe diferite mingi, așezate la diferite poziții față de poartă. De asemena, am testat diferite poziții pentru poartă și pereți.

1. CONCLUZII

În concluzie, realizarea acestui proiect a fost o experiență placută dar în același timp și challenging din multe puncte de vedere deoarece a fost o materie nouă și complexă. De asemenea, am acumulat cunoștințe noi despre roboți, mediul de simulare și am avut ocazia să lucrez cu Linux și Python.

1. BIBLIOGRAFIE

[www 01] <https://edu.gaitech.hk/ria_e100/map-navigation.html>

[www 02] <https://stackoverflow.com/questions/48830056/use-data-from-multiple-topics-in-ros-python>

[www 03] <https://stackoverflow.com/questions/48830056/use-data-from-multiple-topics-in-ros-python>

[www 04] <https://edu.gaitech.hk/ria_e100/map-navigation.html>

[www 05] <https://stackoverflow.com/questions/48830056/use-data-from-multiple-topics-in-ros-python>

[www 06] <https://iq.opengenus.org/norm-method-of-numpy-in-python/>

[www 07] <https://www.youtube.com/watch?v=mFpH9KK7GvI>

[www 08] <https://www.youtube.com/watch?v=kPsrOQrzkaM>

[www 09] <https://answers.ros.org/question/69754/quaternion-transformations-in-python/>

[www 10] <https://stackoverflow.com/questions/47989508/will-actionlib-simpleactionclient-cancel-all-goals-block-until-completion>

[www 11] <http://wiki.ros.org/>

[www 12] <https://emanual.robotis.com/>

[www 13] <https://stackoverflow.com/questions/48830056/use-data-from-multiple-topics-in-ros-python>

[www 14] <https://iq.opengenus.org/norm-method-of-numpy-in-python/>

[www 15] <https://www.youtube.com/watch?v=mFpH9KK7GvI>

[www 16] <https://www.youtube.com/watch?v=kPsrOQrzkaM>

[www 17] <http://repository.utm.md/bitstream/handle/5014/1219/Conf_UTM_2012_I_pg174-175.pdf?sequence=1>

[www 18] <https://www.turtlebot.com/about/>

[www 19] <https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/>

[www 20] <https://dabit.industries/products/turtlebot-3-burger>

[www 21] <https://robots.ros.org/turtlebot3/>

[www 22] <http://wiki.ros.org/ROS/Introduction>

[www 22] <https://en.wikipedia.org/wiki/Gazebo_simulator>