UNIVERSITATEA "POLITEHNICA" DIN TIMISOARA
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL AUTOMATICĂ ŞI INFORMATICĂ APLICATĂ

HIDE & SEEK

PROIECT SINCRETIC I

AUTORI: PEARJĂ Ion

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

1. Cuprins

2. Introducere

Un robot mobil este o mașină controlată de software care utilizează senzori și alte tehnologii pentru a identifica împrejurimile și a se mișca în jurul său. Roboții mobili folosesc o combinație de inteligență artificială (AI) și elemente robotice fizice, cum ar fi roți, picioare. Roboții mobili devin din ce în ce mai populari în diferite sectoare ale industriei. Aceștia sunt sunt folosiți pentru a ajuta la procesele de lucru și chiar pentru a îndeplini sarcini care sunt imposibile sau periculoase pentru lucrătorii umani.

Calificări și tipuri de roboți mobili

Roboții mobili pot fi clasificați în două moduri: după mediul în care lucrează și după dispozitivul pe care îl folosesc pentru a se deplasa.

Exemple de roboți mobili de mediu diferit includ:

- Roboți polari care sunt proiectați să traverseze medii înghețate și neuniforme.
- Roboți aerieni, cunoscuți și sub numele de vehicule aeriene fără pilot (UAV) sau drone, care zboară prin aer.
- Roboți de teren/casă sau vehicule terestre fără pilot (UGV), care navighează pe uscat sau în interiorul caselor.
- Roboți subacvatici sau vehicule subacvatice autonome (AUV), care se pot direcționa și călători prin apă.
- Roboți mobili de livrare și transport care sunt proiectați pentru a muta materiale într-un mediu de lucru.

Diferitele dispozitive care pot clasifica un robot mobil includ:

- Picioare (picioare asemănătoare omului sau animalului)
- Șine
- Roți

Există, de asemenea, două tipuri principale de roboți mobili: roboți mobili autonomi și neautonomi (ghidați).

Roboții mobili ghidați necesită o anumită formă de instrucție sau sistem de ghidare pentru a se deplasa, în timp ce roboții mobili autonomi (AMR) sunt capabili să se miște și să exploreze împrejurimile lor fără nici un fel de direcție externă.

AMR-urile diferă de vehiculele autonome ghidate (AGV) deoarece AMR-urile au capacitatea de a fi mai independente. AGV-urile necesită de obicei ghidaj extern, cum ar fi benzi de magnet, fire sau senzori instalați în podeaua mediului, creând astfel un sistem inflexibil care este atât costisitor, cât și dificil de ajustat pe măsură ce nevoile se schimbă. AMR-urile urmăresc să depășească aceste obstacole, necesitând puțin sau deloc îndrumări externe. **[www01]**

Caracteristicile unui robot mobil

Fiecare robot mobil va încorpora diferite caracteristici care optimizează sistemul pentru a îndeplini un anumit obiectiv sau pentru a îndeplini o anumită sarcină. Cu toate acestea, sistemele de roboți mobili industriali, probabil cel mai frecvent utilizate astăzi, posedă câteva caracteristici de bază care ar trebui să fie întotdeauna prezente. Aceste caracteristici sunt:

- comunicații fără fir
- siguranta integrata
- software de simulare a flotei
- software de management al flotei
- integrarea cu software-ul de supraveghere al companiei

[www01]

Utilizări ale roboților mobili

Funcțiile de bază ale unui robot mobil includ capacitatea de a se deplasa și de a explora, de a transporta încărcături utile și de a îndeplini sarcini complexe folosind un dispozitiv, cum ar fi brațele robotice. În timp ce utilizarea industrială a roboților mobili este populară, în special în depozite și centre de distribuție, funcțiile sale pot fi aplicate și în medicină, chirurgie, asistență personală și securitate. Explorarea și navigația oceanelor și spațiului sunt, de asemenea, printre cele mai comune utilizări ale roboților mobili.

Roboții mobili sunt folosiți pentru a accesa zone periculoase, cum ar fi centralele nucleare, unde factori, precum radiațiile mari, fac zona prea periculoasă pentru ca oamenii să o poată inspecta și monitoriza. Cu toate acestea, robotica mobilă actuală nu proiectează roboți care pot tolera radiații mari fără ca circuitele lor electronice să fie afectate. În prezent, se fac încercări de a inventa roboți mobili care să facă față în mod specific acestor situații.

Alte utilizări ale roboților mobili includ:

- explorarea minelor;
- repararea navelor;
- exoschelet robot pentru a transporta încărcături grele pentru soldații militari;
- mașini de vopsit;
- brațe robotizate pentru a ajuta medicii în intervenții chirurgicale;
- fabricarea de proteze automate care imită funcţiile naturale ale organismului;
- aplicații de patrulare și monitorizare, cum ar fi supravegherea condițiilor termice și a altor condiții de mediu

[www01]

3. Prezentarea temei

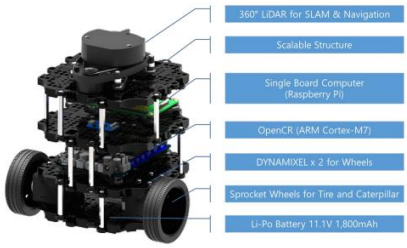
Tema echipei, Hide&Seek, implică setarea/configurarea mediului de simulare a robotului, precum și scrierea codului pentru a determina comportamentul acestuia. Robotul trebuie să simuleze jocul pentru copii 'Hide&Seek', ceea ce determină 2 moduri de funcționare ale acestuia:

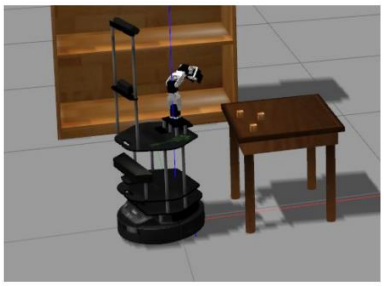
- Modul Hide:
  - Robotul incepe simularea dintr-o pozitie fixă pe hartă.
  - În python se fac proiecții pe hartă astfel încât să se calculeze zonele blocate de obstacole pentru sensorul *lidar*.
  - Se depistează un punct din cele calculate la pasul anterior și se deplasează spre acesta.
- Modul Seek:
  - Robotul incepe simularea dintr-o pozitie fixă pe hartă.
  - În python se fac proiecții pe hartă astfel încât să se calculeze zonele blocate de obstacole pentru sensorul *lidar*.
  - Se depistează un punct din cele calculate la pasul anterior și se deplasează spre acesta.
  - Când ajunge la destinație nodul pentru detectarea culorii este activat.
  - Daca culoarea este depistată atunci se afișează un mesaj în consolă. În caz contrar se reia punctul 2.

4. Tehnologii utilizate

Pentru crearea mediului de lucru am folosit soft-ul *Oracle Virtual Box* în care am creat o mașină virtuală bazată pe *Ubuntu 20.04 LTS*. Tot lucrul dedicat proiectului a fost elaborat în interiorul acestei mașine virtuale.

Am instalat simulatorul *Gazebo* cu versiunea *11.9.0* cu pachetele opționale pentru *ROS Noetic*, care simulează comportamentul robotului și dispune de aceleași caracteristici ca robotul real. Am folosit IDE-ul *VS Code* pentru un proces mai ușor de editare a fișierelor, în special a script-urilor Python. Versiunea de *Python* instalată este *3.8.10*, împreună cu bibliotecile *OpenCV* care permit efectuarea calculelor asupra imaginilor.



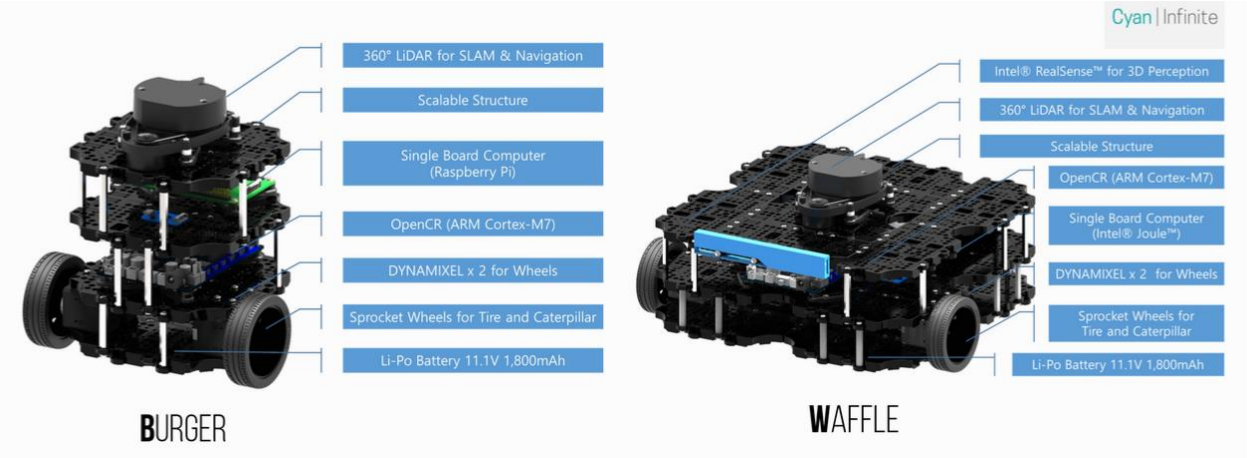


ROS
Robot Operating System

python™

Visual Studio Code

Robotul folosit este un *TurtleBot3 Waffle Pi*. Există și alte opțiuni de roboți (Burger, Waffle), dar din cauza că avem nevoie de a captura imagini folosim Waffle Pi, care dispune de o cameră video.



[www02]

5. Ghidul programatorului

Comportamentul robotului este definit în 2 scripturi python: *search_color.py* și *navigate.py.*
În scriptul *search_color.py* are loc procesarea imaginilor pentru a detecta culoarea roșie.

```
lower = np.array([0, 0, 100], dtype = "uint8")
upper = np.array([0, 0, 255], dtype = "uint8")
mask = cv2.inRange(cv_image, lower, upper)
output = cv2.bitwise_and(cv_image, cv_image, mask = mask)
if np.any((output != 0)):
    pub.publish("found")
```

În această secvență de cod sunt stabilite variabilele lower și upper care determină limitele culorii roșii după care vor fi procesate imaginile. Se creează o masca, care este aplicată pe imaginea primită de la topicul */camera/rgb/image_raw/compressed*, rezultatul fiind stocat în output. Se verifică dacă output-ul, care de fapt este o matrice, conține măcar o valoare diferită de 0. O valoare diferită de 0 înseamnă prezența culorii roșii. În acest caz se publică în topicul *main_state* valoarea 'found'.

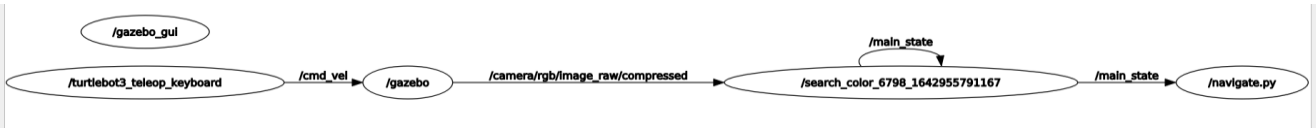
```
pub = rospy.Publisher('main_state', String, queue_size=10)
rospy.init_node('search_color', anonymous=True)
rospy.Subscriber('main_state', String, main_state_callback)
image_sub = rospy.Subscriber("/camera/rgb/image_raw/compressed", CompressedImage,
    image_sub_callback, queue_size=1)
```

În această secvență de cod se creează topicul *main_state* și nodul *search_color*. De asemenea, se 'abonează' la topicul creat în prima linie de cod, precum și la topicul */camera/rgb/image_raw/compressed.*

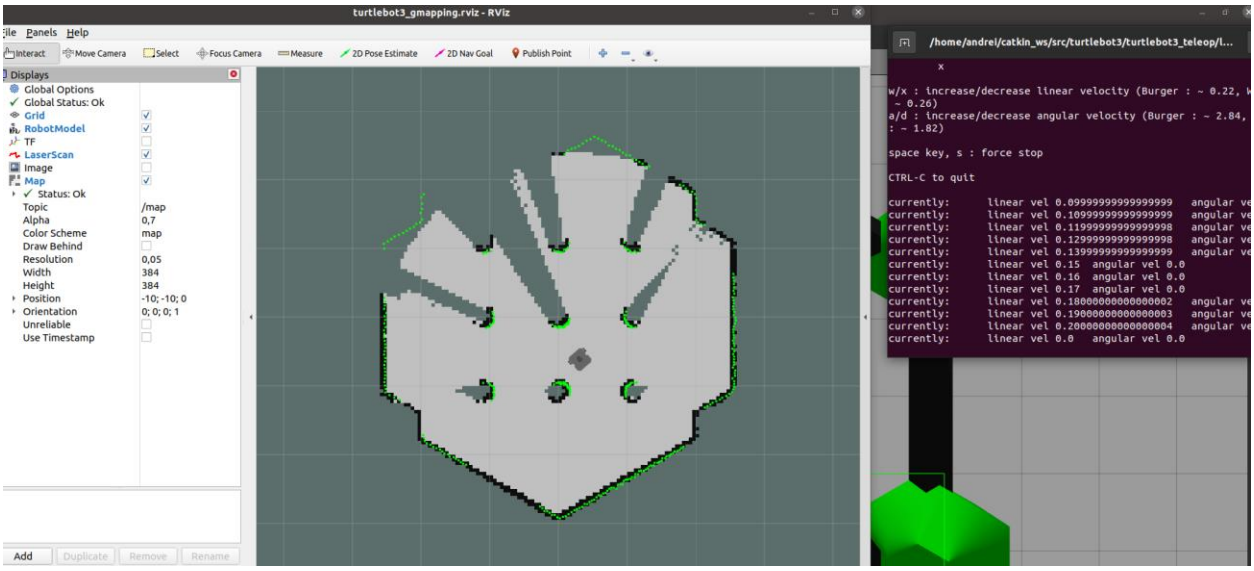
În scriptul navigate.py se 'abonează' la topicul *main_state* și se inițializează nodul *navigate.py.*

```
rospy.Subscriber('main_state', String, main_state_callback)
rospy.init_node('navigate.py')
```

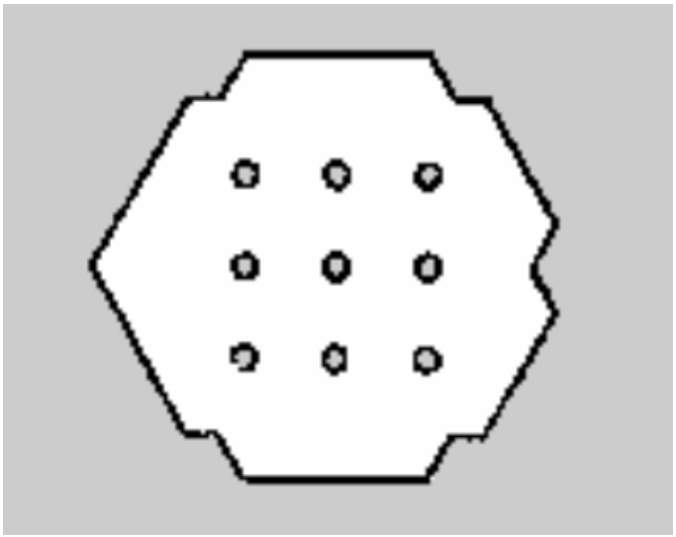
Mai jos este reprezentată grafic structura nodurilor și topicurilor din mediul ROS, care e generată folosind comanda *rqt_graph* în terminal. Se observă că atât nodul *search_color,* cât și *navigate* s-au 'abonat' la topicul *main_state.* Mai mult decât atât, nodul *search_color* este abonat la topicul */camera/rgb/image_raw/compressed,* de unde ia imaginile capturate de camera de pe robot pentru procesarea acestora. Topicul */cmd_vel* e folosit pentru a publica datele primite de la tastatură pentru mișcarea robotului pe hartă.



Pentru partea de navigare am mapat harta cu ajutorul componentei *SLAM*. Am folosit comanda *roslaunch turtlebot3_slam turtlebot3_slam.launch* în terminal și am condus cu robotul pe hartă. Acesta creează harta cu ajutorul sensorului *lidar* de pe bord, măsurând distanța până la obstacole și interpretând rezultatele.



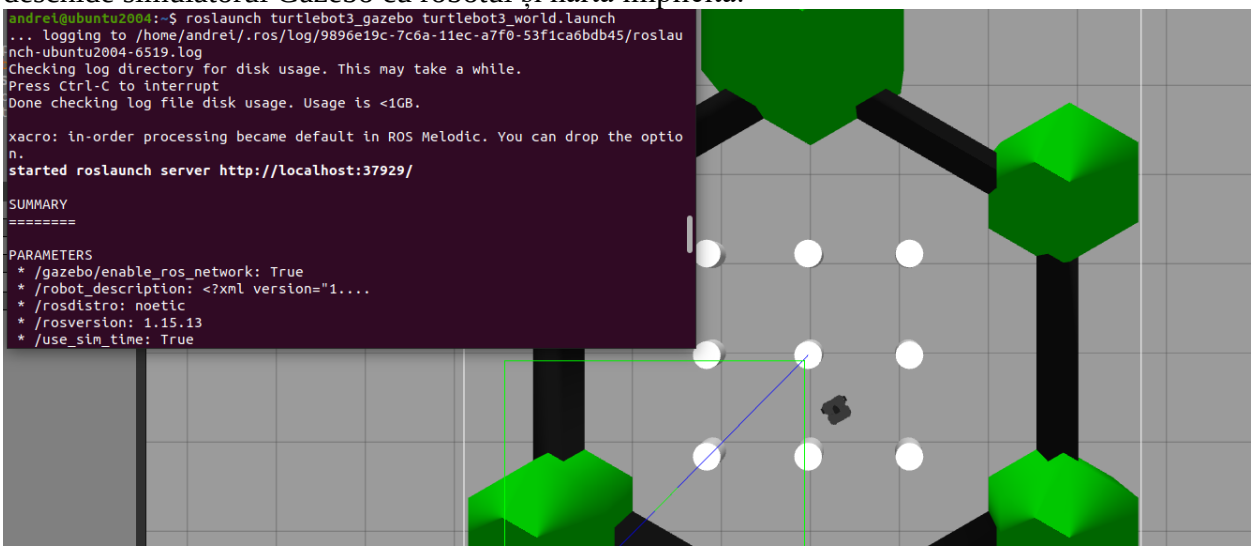
Am salvat mapa folosind comanda *rosrun map_server map_saver -f ~/world.* Aceasta arată în modul următor:



6. Ghidul utilizatorului

Pentru a rula simularea proiectului se vor urma pașii următori:

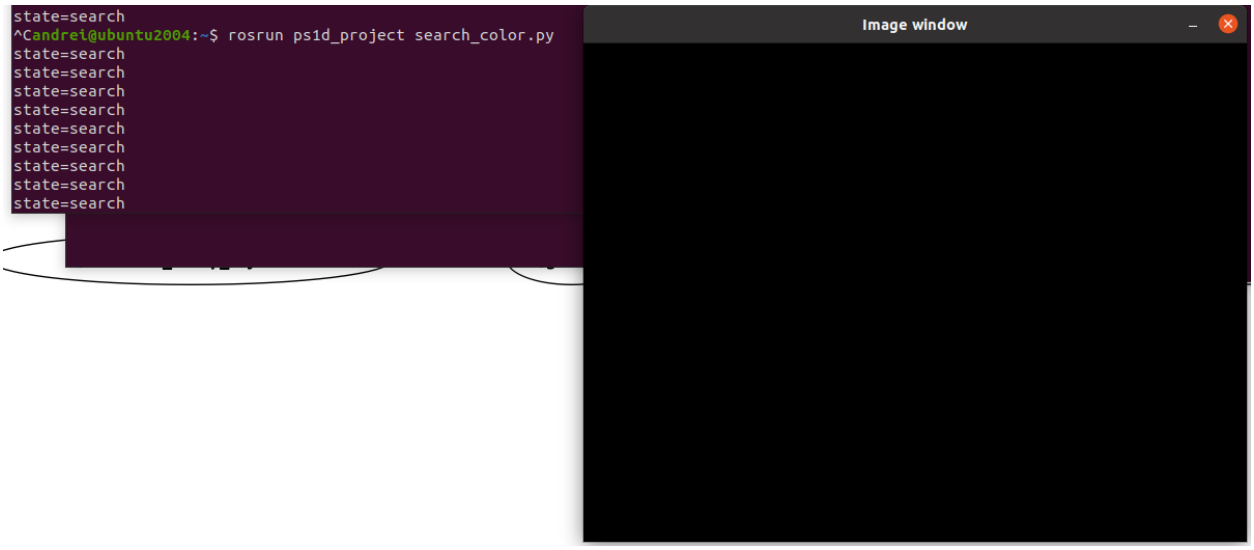
1. Se rulează comanda *roslaunch turtlebot3_gazebo turtlebot3_world.launch* pentru a deschide simulatorul Gazebo cu robotul și harta implicită:



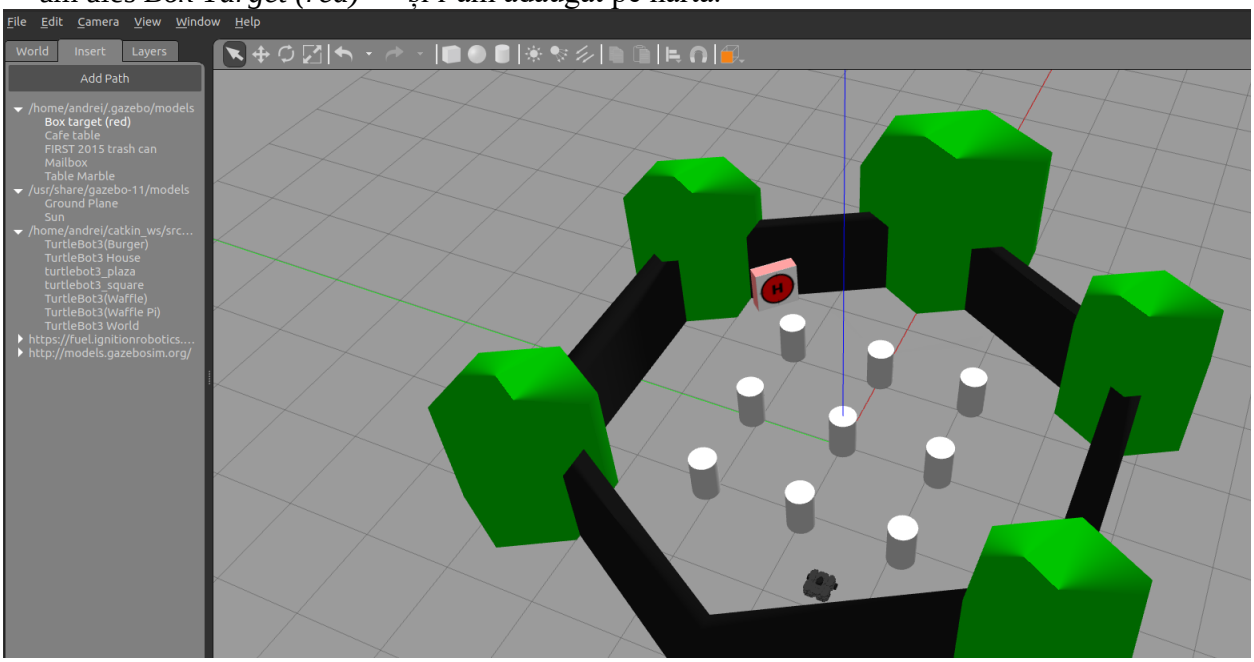
2. Se rulează comanda *roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch* pentru a porni nodul *turtlebot3_teleop_keyboard* care ne permite să controlăm robotul cu ajutorul tastaturii.

```
andrei@ubuntu2004:~$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
... logging to /home/andrei/.ros/log/9896e19c-7c6a-11ec-a7f0-53f1ca6bdb45/roslaunch-ubuntu2004-7114.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://localhost:39395/

SUMMARY
========

PARAMETERS
 * /model: waffle_pi
 * /rosdistro: noetic
 * /rosversion: 1.15.13

NODES
  /
    turtlebot3_teleop_keyboard (turtlebot3_teleop/turtlebot3_teleop_key)

ROS_MASTER_URI=http://localhost:11311
```

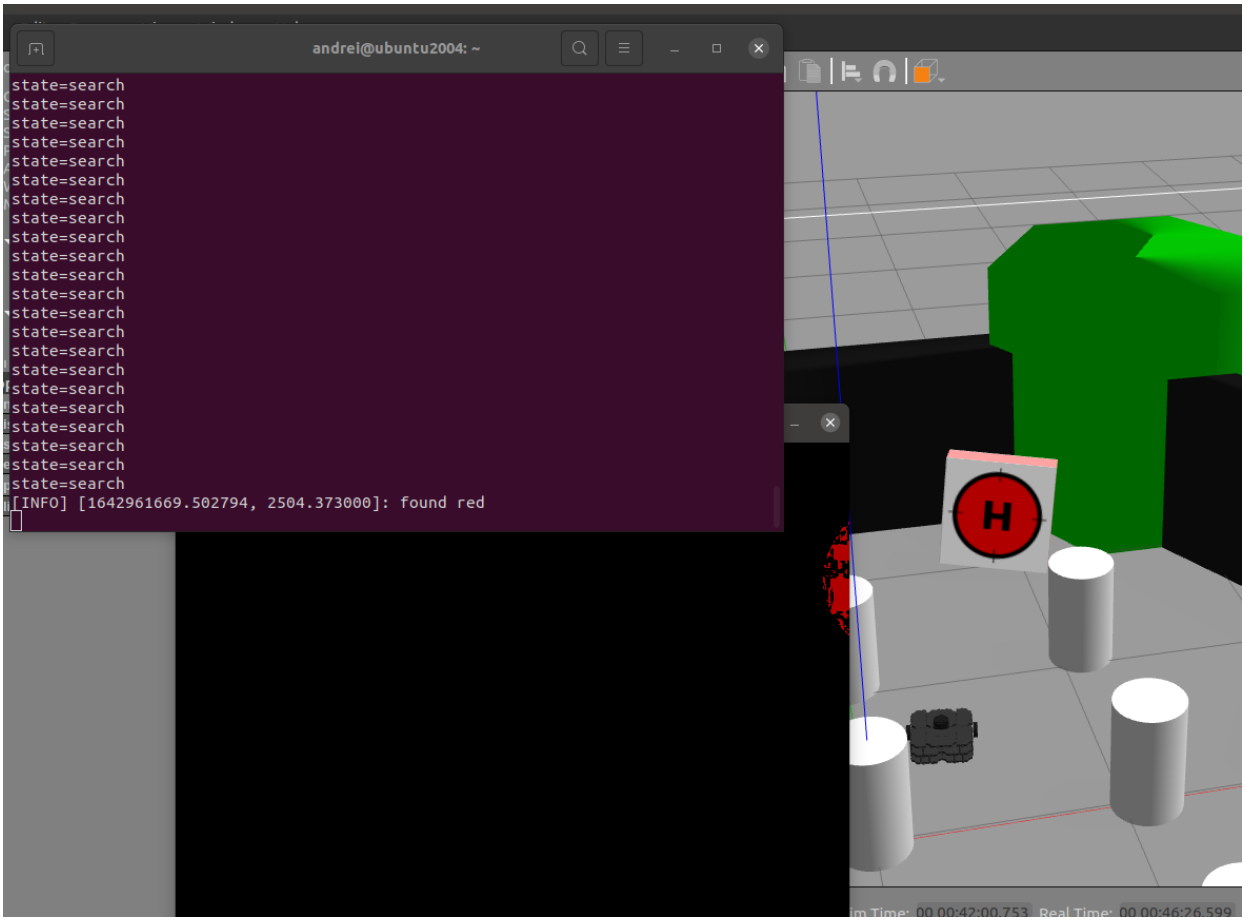
3. Se rulează scriptul search_color.py folosind comanda *rosrun ps1d_project search_color.py* care realizează procesarea imaginilor în timp real.



4. Se adaugă un obiect de culoare roșie pe care robotul urmează să-l caute. În meniul *Insert* -> am ales *Box Target (red)* -> și l-am adăugat pe hartă.



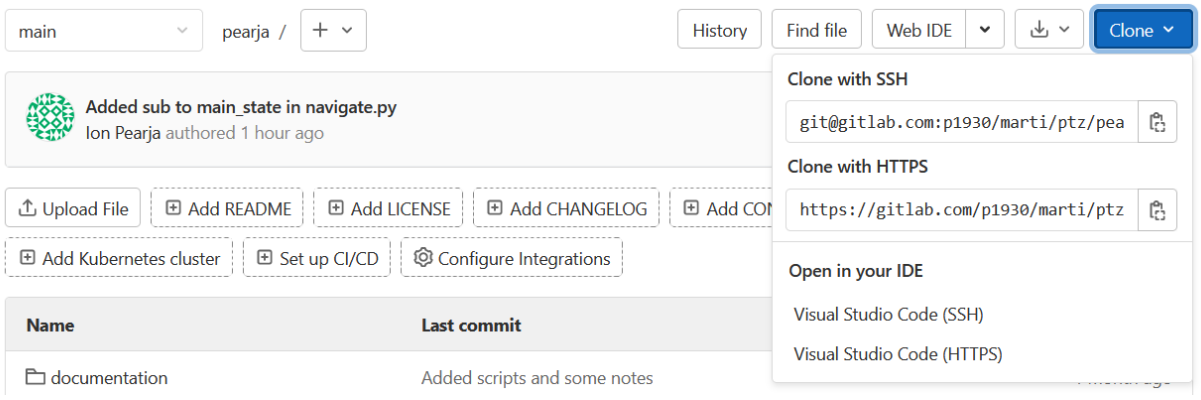
5. Se controlează robotul utilizând terminalul creat la punctul 2 pentru a naviga pe hartă și ca robotul să caute culoarea roșie.



7. Testare și punere în funcțiune

Pentru a testa aplicația trebuie:

1. Descărcat codul de pe Git, folosind comanda *git clone* urmată de adresa web a proiectului pe GitLab în terminal. Astfel toate fișierele sursă vor fi descarcate în spațiul de lucru. Adresa proiectului poate fi luată din interfața platformei GitLab



2. Navigăm în directoriul *catkin_ws*, care implicit se află în *~/catkin_ws*.
3. Rulăm comanda *catkin_make* pentru a compila proiectul.
4. Pașii urmatori pentru testare sunt descriși la punctul anterior (Ghidul utilizatorului)

8. Prezentarea firmei

Firma este compusă din 3 membri: Pearjă Ion, Zugravu Claudiu-Cristian, Teodorescu Raul-Andrei. Împreună am elaborat planul de lucru, etapele și logica proiectului. Pearjă Ion a realizat partea răspunzătoare de procesarea imaginilor pentru a detecta culoarea roșie, precum și structura de topicuri în script-urile necesare.

9. Concluzii

În procesul complex de elaborare a proiectului am învățat mai multe competențe. M-am cunoscut cu mai multe tehnologii:

- Am setat o mașină virtuală în mediul Oracle Virtual Box.
- Am instalat pachete software (ROS, Gazebo, python) direct din terminal pe SO-ul Ubuntu 20.04 LTS.
- Am configurat un repository de git local în Ubuntu și l-am sincronizat cu cel remote de pe GitLab. De asemenea, am învățat cum se setează credențialele de pe git, diferența dintre *commit, fetch, pull, push, amend.*
- Am studiat sintaxa unui nou limbaj de programare, Python, precum și vasta bibliotecă OpenCV.
- Am învățat cum să lucrez cu IDE-ul VS Code și să folosesc beneficiile acestuia, precum lucrul integrat cu Git.
- Am folosit interfața simulatorului Gazebo pentru a adăuga și configura obiecte pe hartă.
- Am aflat ce reprezintă un sensor *lidar.*

De asemenea, am avut posibilitatea de a crea un plan de elaborarea pașilor unui proiect într-o echipă constituită din mai mulți membri.

10. Bibliografie

[www01] https://internetofthingsagenda.techtarget.com/definition/mobile-robot-mobile-robotics

[www02] https://cyaninfinite.com/installing-turtlebot-3-simulator-in-ubuntu-14-04/