**UNIVERSITATEA “POLITEHNICA” DIN TIMISOARA**

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**DEPARTAMENTUL AUTOMATICĂ ŞI INFORMATICĂ APLICATĂ**

**Conducere automata a unui process**

**-Autobot-**

**PROIECT SINCRETIC**

**AUTORI: Eduard GJURAJ**

**Christian GHINEA**

**Anul III AIA - an universitar 2023 / 2024**

Cuprins

1. Introducere………………………………………………………………………… 3

2. Prezentarea temei – Autobot…..…………………………………………………... 4

3. Tehnologii utilizate………………………………………………………………... 5

4. Ghidul programatorului…………………………………………………………… 7

5. Ghidul utilizatorului………………………………………………………………. 10

7.Bibliografie………………………………………………………………………... 12

**Introducere**

**Notiuni generale**

Conceptul de conducere se referă la procesul dinamic de organizare și coordonare realizat de către un individ sau entitate într-o anumită perioadă de timp, având ca obiectiv îndeplinirea unor sarcini sau atingerea unor scopuri specifice, prin intermediul altor grupuri de membri ai unui sistem. În contextul sistemelor tehnice, conducerea implică o acțiune intenționată asupra unui obiect care trebuie reglat sau controlat. Aplicațiile pentru care au fost, sunt și vor fi dezvoltați roboții mobili sunt extrem de diverse și nu pot fi acoperite în detaliu în această prezentare. O mare parte dintre roboții de dimensiuni mici au găsit aplicații în domeniul medical, unde pot naviga în interiorul vaselor și tuburilor corpului uman, desfășurând investigații, efectuând intervenții chirurgicale sau administrând medicamente. Pe de altă parte, roboții de dimensiuni mari sunt utilizați într-o varietate de domenii, cum ar fi industria (vehicule automate pe roți cu ghidare automată pentru transport și manipulare de piese), agricultura (mașini agricole autonome), silvicultura, militar, securitate, divertisment (roboți-jucării și roboți pentru competiții) și operațiuni de salvare (roboți de salvare - rescue robots).

Un robot este definit ca un operator mecanic sau virtual, creat artificial. Structura unui robot este compusă din multiple elemente, inclusiv mecanică, senzori și actuatori, precum și un mecanism de control. Mecanica determină aspectul și mișcările posibile ale robotului în timpul funcționării. Senzorii și actuatorii sunt folosiți pentru interacțiunea cu mediul înconjurător. Mecanismul de control asigură ca robotul își atinge cu succes obiectivele, evaluând, de exemplu, informațiile provenite de la senzori și planificând mișcările necesare. Acest mecanism reglează motoarele și coordonează mișcările ce trebuie efectuate.

Deși domeniul robotică este vast și nu are o definiție universală, putem simplifica conceptul de manipulator ca fiind un mecanism multifuncțional, cu multiple grade de libertate, care poate fi controlat direct de un operator uman sau printr-un sistem logic.

**Prezentarea temei**

Tema propusa pentru conducerea procesului este **Autobot**.

Proiectul e concentrează pe dezvoltarea unui robot autonom cu capacitatea de a detecta obstacolele din calea sa și de a le ocoli pentru a evita coliziunile. Acest robot este dotat cu senzori pentru a identifica obiectele din fața sa și pentru a lua decizii inteligente privind traseul său. În plus, proiectul include o funcționalitate de analiză a mișcării în timp real, utilizând camera laptopului pentru a detecta orice mișcare din mediul real.

Atata timp cat camera detectează mișcare, robotul merge autonom evitand obstacolele intampinate pe harta. Cand camera nu mai detecteaza miscare functionalitatea de evitare a obstacolelor se opreste.

Diagram

Description automatically generated**Tehnologii utilizate**

* Robot TurtleBot3
* ROS
* Python
* Ubuntu
* GitLab

Diagram, schematic

Description automatically generated **TurtleBot3** este un robot mobil mic, accesibil, programabil, bazat pe **ROS**, pentru utilizare în educație, cercetare, hobby și prototipuri de produse. Tehnologia de la baza TurtleBot3 este **SLAM**, Navigare și Manipulare, ceea ce îl face potrivit pentru roboții de serviciu la domiciliu.

TurtleBot3 poate fi personalizat în diferite moduri, în funcție de modul în care reconstruiți părțile mecanice și utilizați piese opționale, cum ar fi computerul și senzorul. În plus, TurtleBot3 a evoluat cu un SBC rentabil și de dimensiuni mici, care este potrivit pentru sisteme încorporate robuste, senzori de distanță de 360 ​​de grade și tehnologie de imprimare 3D.

**ROS**, cunoscut și ca **Robot Operating System**, este un sistem open-source pentru gestionarea operațiilor la nivel înalt ale roboților. Acesta furnizează o serie de servicii esențiale, inclusiv abstractizarea hardware-ului, controlul dispozitivelor la nivel scăzut, implementarea funcționalităților comune, transmiterea mesajelor între procese și administrarea pachetelor software. În plus, ROS pune la dispoziție un set variat de instrumente și biblioteci pentru dezvoltarea, construirea, scrierea și rularea codului pe mai multe platforme.

Obiectivele principale ale ROS sunt următoarele:

* Biblioteci independente de ROS: Acest sistem promovează dezvoltarea de biblioteci independente de ROS, cu interfețe funcționale clare și portabile.
* Flexibilitate în limbaje: ROS poate fi implementat ușor într-o gamă variată de limbaje de programare moderne. Deja există implementări în Python, C++, și Lisp, iar există și biblioteci experimentale pentru Java și Lua.
* Testare simplă: ROS dispune de un cadru integrat de testare de unitate și integrare, numit rostest, care facilitează testarea și depanarea eficientă a componentelor software.
* Scalabilitate: ROS este conceput pentru a fi potrivit atât pentru sisteme de dimensiuni mari, cât și pentru procese de dezvoltare complexe și ample.

**Ghidul programatorului**

**Script folosit:**

#!/usr/bin/env python3

import rospy

import cv2

import numpy as np

from geometry\_msgs.msg import Twist

from cv\_bridge import CvBridge, CvBridgeError

import time

from imutils.video import VideoStream

import imutils

class MovementDetectionTurtleBot:

def \_\_init\_\_(self):

rospy.init\_node('movement\_detection\_turtlebot', anonymous=True)

self.bridge = CvBridge()

self.cmd\_vel\_pub = rospy.Publisher('/cmd\_vel', Twist, queue\_size=10)

self.last\_movement\_time = 0

self.velocity = Twist()

self.vs = VideoStream(src=0).start()

time.sleep(2.0)

self.fgmask = None

def detect\_movement(self):

while not rospy.is\_shutdown():

frame = self.vs.read()

frame = imutils.resize(frame, width=400)

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

gray = cv2.GaussianBlur(gray, (21, 21), 0)

if self.fgmask is None:

self.fgmask = gray.copy().astype("float")

continue

cv2.accumulateWeighted(gray, self.fgmask, 0.5)

frame\_delta = cv2.absdiff(gray, cv2.convertScaleAbs(self.fgmask))

thresh = cv2.threshold(frame\_delta, 25, 255, cv2.THRESH\_BINARY)[1]

if np.any(thresh):

self.last\_movement\_time = time.time()

self.stop\_turtlebot()

elif time.time() - self.last\_movement\_time >= 5:

self.reset\_turtlebot()

cv2.imshow('Camera Feed', frame)

cv2.waitKey(1)

def stop\_turtlebot(self):

self.velocity.linear.x = 0

self.velocity.angular.z = 0

self.cmd\_vel\_pub.publish(self.velocity)

def reset\_turtlebot(self):

# Set the desired velocity and angular velocity here

self.velocity.linear.x = 0.2 # Set the linear velocity

self.velocity.angular.z = 0.5 # Set the angular velocity

self.cmd\_vel\_pub.publish(self.velocity)

def main():

try:

movement\_detection\_turtlebot = MovementDetectionTurtleBot()

movement\_detection\_turtlebot.detect\_movement()

except rospy.ROSInterruptException:

pass

finally:

cv2.destroyAllWindows()

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Ghidul utilizatorului**

1 -> export TURTLEBOT3\_MODEL=burger

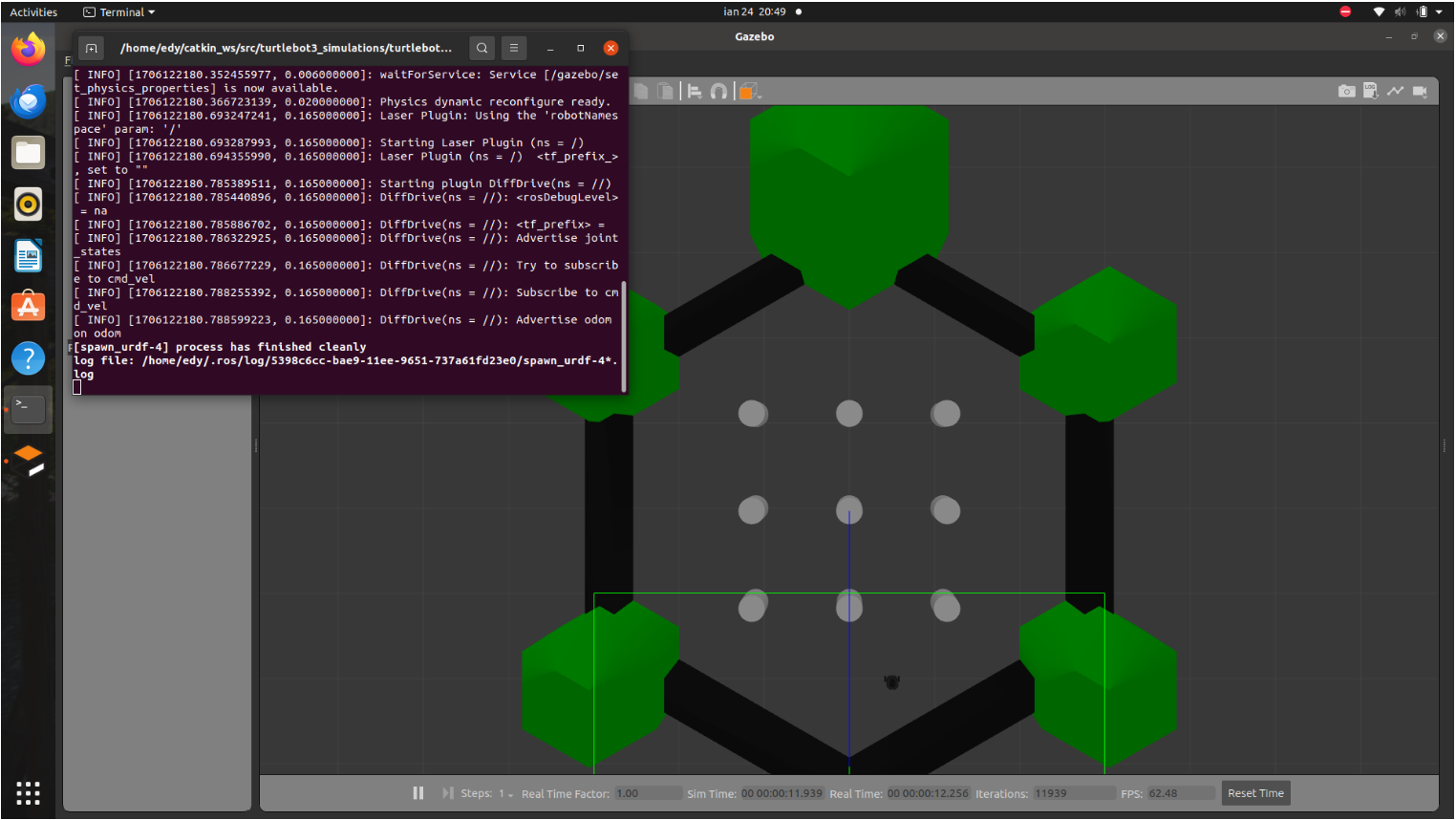
roslaunch turtlebot3\_gazebo turtlebot3\_world.launch

2 -> export TURTLEBOT3\_MODEL=burger

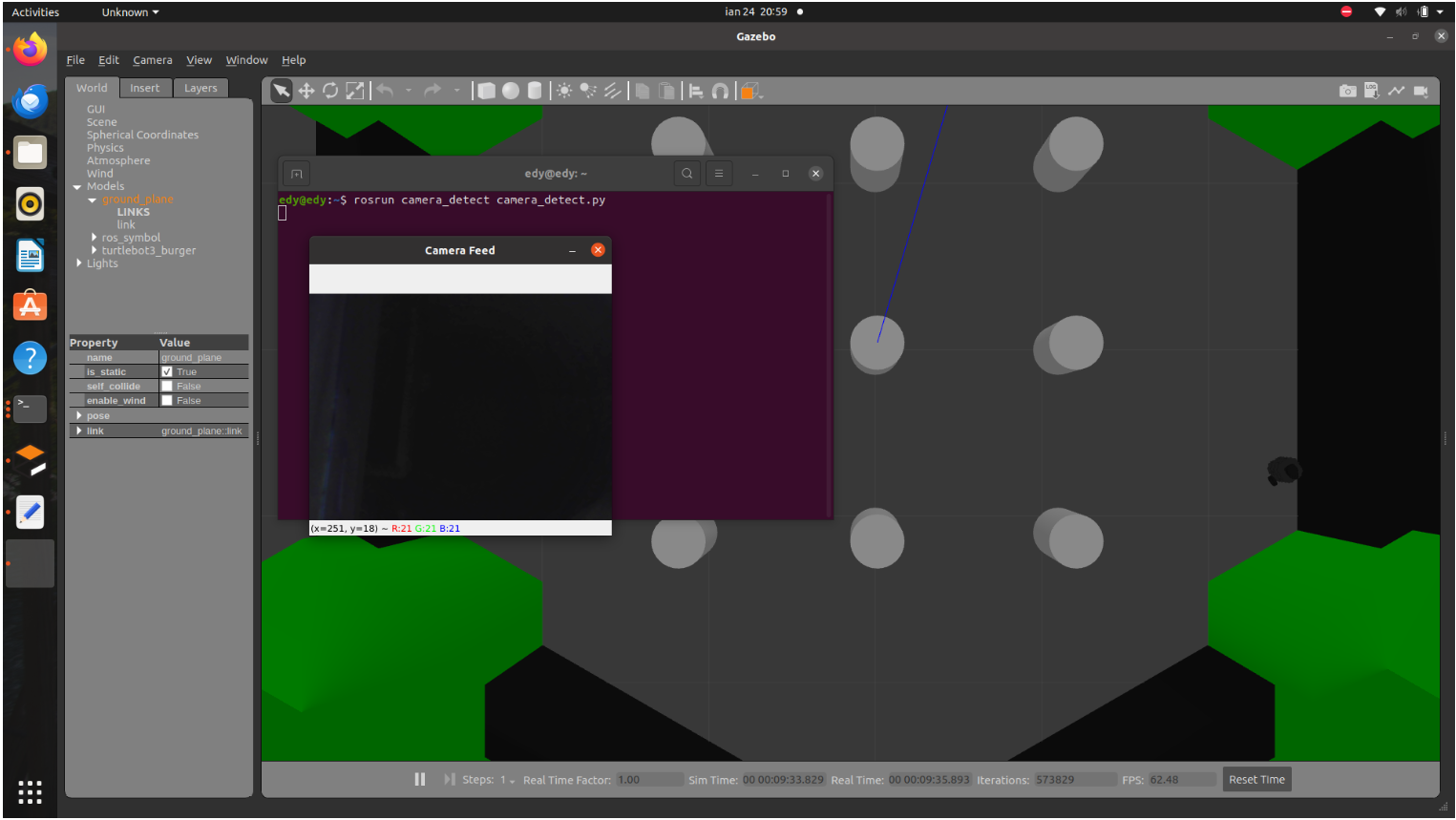
roslaunch turtlebot3\_gazebo turtlebot3\_simulation.launch

3 -> rosrun camera\_detect camera\_detect.py

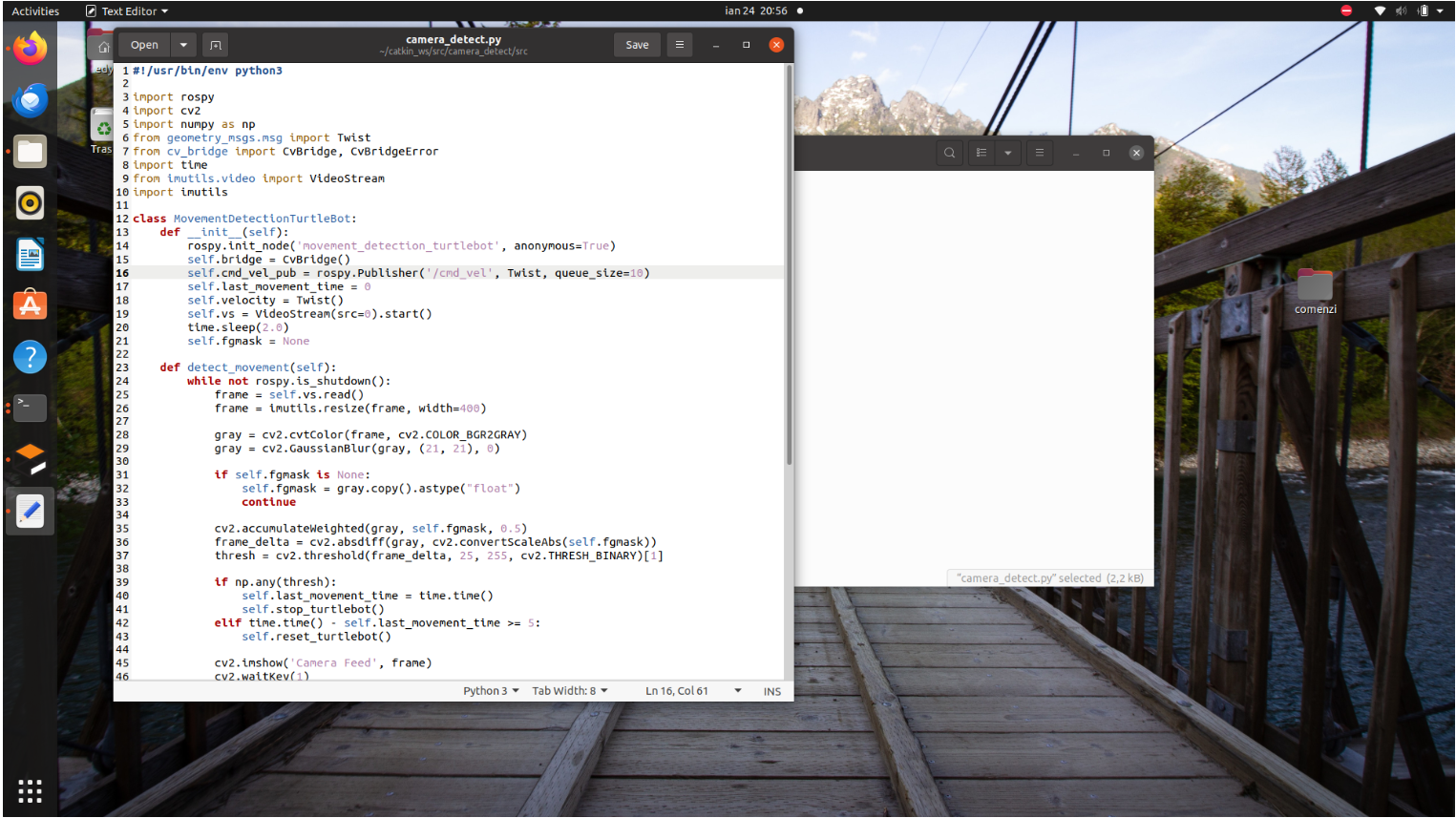
**1**. Exportam modelul de Turtlebot3 pe care il vom folosi in simulare, apoi vom deschide lumea in care il vom simula in programul Gazebo.



**2**. Introducem comanda care va face ca Turtlebot-ul nostru sa poata sa evite obstacolele.



**3.** Introducem comanda care ne va deschide camera prin care putem comunica cu Turtlebot-ul ca sa ii putem specifica cand vrem ca acesta sa evite obstacolele pe care le intampina in calea lui.



**Bibliografie**

* Tutorial TurtleBot3 ROS:

<https://automaticaddison.com/how-to-launch-the-turtlebot3-simulation-with-ros>

* Videoclipuri ROS educaționale: <https://www.youtube.com/watch?v=5fZmuPxMZz0&list=PLRG6WP3c31_V3SVb6UHCnbEgqEy7etwFI>
* Documentație navigație TurtleBot3: <https://emanual.robotis.com/docs/en/platform/turtlebot3/navigation/#ros-1-navigation>
* Sistem de operare Ubuntu:   
  <https://ubuntu.com>
* Site oficial ROS (Robot Operating System):

<https://www.ros.org>

* Camera movement detection:  
  <https://opencv.org>