Ingineria Sistemelor
2023-2024

PROIECT SINCRETIC
-Fotbalist-

Autori: Codină Maria-Beatrice
Fiat Vanessa Ariana Ionela
Email: maria.codina@student.upt.ro
vanessa.fiat@student.upt.ro

31 Noiembrie 2022

1. Cuprins

1 INTRODUCERE

În ultimele decenii, domeniul roboților mobili a devenit o ramură esențială a roboticilor, aducând inovații semnificative în diverse domenii precum educația, cercetarea și industria. Roboții mobili sunt dispozitive autonome sau semiautonome, capabile să se deplaseze și să interacționeze cu mediul înconjurător, deschizând noi perspective în ceea ce privește automatizarea și eficiența proceselor.

Un element crucial în evoluția acestui domeniu este reprezentat de conducerea la distanță a roboților mobili, o tehnologie care a evoluat rapid și care oferă posibilitatea de a controla și monitoriza roboții de la distanță, utilizând diverse mijloace de comunicație. Această abordare deschide noi orizonturi în domeniul intervenției în medii periculoase sau inaccesibile pentru oameni, precum și în utilizările educaționale și de cercetare.

Proiectul nostru se concentrează pe integrarea unui robot mobil în cadrul platformei TurtleBot3, o soluție open-source populară, versatilă și accesibilă. Ne propunem să explorăm posibilitățile oferite de această platformă în ceea ce privește cartografierea, navigația și detecția obiectelor, aducând în prim-plan avantajele aduse de utilizarea roboților mobili în diverse aplicații practice.

Prin intermediul acestui proiect, vom explora și implementa conducerea la distanță a robotului nostru, folosind tehnologii moderne și interfețe intuitive. Vom analiza implicațiile acestei tehnologii în ceea ce privește eficiența operațională și adaptabilitatea la diferite scenarii de utilizare.

În final, acest proiect nu reprezintă doar o experiență practică în manipularea și controlul roboților mobili, ci și o incursiune în lumea inovației tehnologice și a posibilităților nelimitate pe care o astfel de tehnologie o poate oferi în viitorul roboticelor mobile.

2 PREZENTAREA TEMEI

Tema proiectului este Fotbalistul și constă în programarea robotului TurtleBot3 Waffle să găsească poarta adversarului și să marcheze gol.

Pentru început, plasăm Fotbalistul pe teren. Acesta va executa o rotire în cazul în care nu a găsit deja mingea, iar dacă a fost găsită acesta se va deplasa drept spre ea. Pentru a realiza acest lucru am procesat imaginile primite de la camera robotului (dacă detectează culoarea verde a mingii, acesta se deplasează spre ea).

Senzorul pe care îl folosește robotul este un senzor “360 Laser Distance Sensor LDS-01” care ajută la determinarea poziției robotului nostru, cât și la identificarea imaginilor.

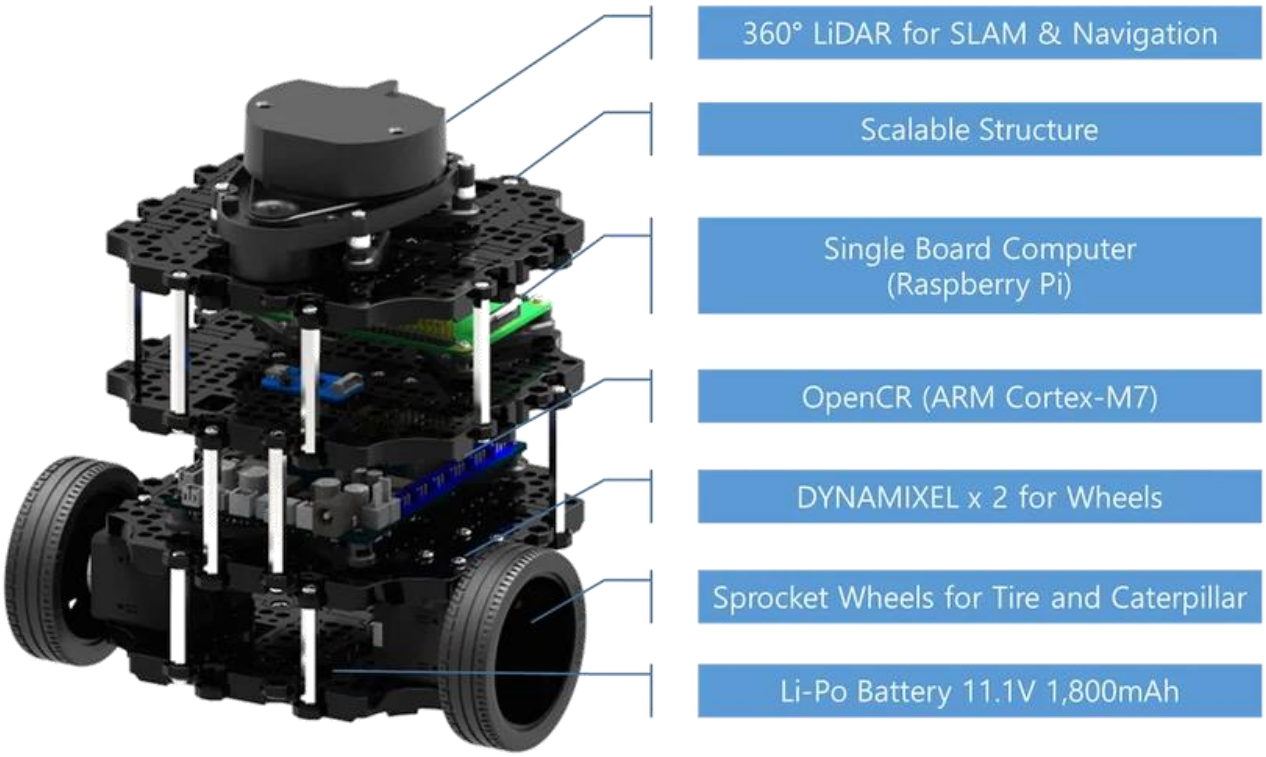
Fotbalistul știe să dea gol după preluarea mingii în poarta adversarului.

3 DESCRIEREA ROBOTULUI

Robotul TurtleBot3 este rezultatul unor eforturi majore de a dezvolta un robot puternic și rapid, dar care să fie în același timp ușor și mobil.

Este considerat un robot de o nouă generație, fiind modular, compact și personalizabil, folosit ăn aplicații interesante pentru educație, cercetare și dezvoltare de produse.

Acest robot cântărește doar 1 kg, dar poate cară o încărcătură adițională de 15 kg. Are o înălțime de 192 mm și un diametru de 160mm.



4 TEHNOLOGII UTILIZATE

Am folosit Visual Studio Code ca mediu de dezvoltare, iar limbajul de programare folosit este Python.

Visual Studio Code este un editor sursă ușor, dar puternic, care rulează pe desktop și este disponibil pentru Windows, macOS și Linux. Vine cu suport integrat pentru

JavaScript, TypeScript și Node.js și are un ecosistem bogat de extensii pentru alte limbi (cum ar fi C++, C#, Java, Python, PHP, Go) și runtime (cum ar fi .NET și Unity).

Python este un limbaj de programare dinamic multi-paradigmă, concentrându-se asupra programării imperative, orientate pe obiecte și funcționale, ceea ce permite o flexibilitate mai mare. Python pune accentul pe curățenia și simplitatea codului, iar sintaxa să le permită dezvoltatorilor să exprime unele idei programatice într-o manieră mai clară și mai concisă decât în alte limbaje de programare ca C.

ROS (Sistem de operare Robot) oferă biblioteci şi instrumente pentru a ajuta dezvoltatorii de software să creeze aplicații robot. Oferă abstractizarea hardware, driverele de dispozitiv, bibliotecile, vizualizatoarele, transmiterea mesajelor, gestionarea pachetelor şi multe altele. ROS este licențiat sub licenţa BSD, open source.

Procesele ROS sunt reprezentate ca noduri într-o structură grafică, conectate prin margini numite topicuri.

Noduri

Un nod reprezintă un singur proces care rulează graficul ROS. Fiecare nod are un nume, pe care îl înregistrează cu masterul ROS înainte de a putea întreprinde alte acțiuni. Mai multe noduri cu nume diferite pot exista sub spatii de nume diferite, sau un nod poate fi definit ca anonim, caz în care va genera la întâmplare un identificator suplimentar pentru a se adăuga la numele său dat. Nodurile sunt în centrul programării ROS, întrucât majoritatea codului client ROS se prezintă sub forma unui nod ROS care întreprinde acțiuni bazate pe informațiile primite de la alte noduri, trimite informații către alte noduri sau trimite şi primeşte cereri de acțiuni către şi de la alte noduri.

Topicuri

Topicurile sunt denumite autobuze peste care nodurile trimit şi primesc mesaje.Numele de topicuri trebuie să fie unice şi în spațiul lor de nume. Pentru a trimite mesaje la un topic, un nod trebuie să publice pe topicul respectiv, în timp ce pentru a primi mesaje trebuie să se aboneze. Modelul de publicare/ abonare este anonim: niciun nod nu ştie ce noduri trimit sau primesc pe un topic, ci doar că acesta trimite/ primeşte pe acel topic. Tipurile de mesaje transmise pe un topic variază foarte mult şi pot fi definite de utilizator. Conținutul acestor mesaje poate fi: date despre senzori, comenzi de control ale motorului, informații de stare, comenzi de actionare sau orice altceva.

Pachete

ROS contine multe implementări open source de funcționalitate și algoritmi de robotică comună. Aceste implementări open source sunt organizate în ,,pachete". Multe pachete sunt incluse ca parte a distribuțiilor ROS, în timp ce altele pot fi dezvoltate de către persoane şi distribuite prin site-uri de distribuire a codurilor, cum ar fi github.

5 GHIDUL PROGRAMATORULUI

a. Capturarea imaginii

Acest proces se bazează pe capturarea unor imagini cu ajutorul camerei frontale.

```
def image_callback(self, data):
    try:
        cv_image = self.bridge.imgmsg_to_cv2(data, 'bgr8')
    except CvBridgeError as e:
        print(e)
        return

    # Convert BGR to HSV
    hsv = cv2.cvtColor(cv_image, cv2.COLOR_BGR2HSV)

    # Define the range for all colors in HSV
    lower_color = np.array([0, 50, 50])
    upper_color = np.array([179, 255, 255])

    # Threshold the HSV image to get all colors
    mask = cv2.inRange(hsv, lower_color, upper_color)

    # Find contours in the mask
    contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

b. Detectarea mingii

Acest proces are ca scop detectarea mingii cu ajutorul camerei robotului printr-o rotație continuă a robotului.

```
if self.detecting_ball:
    # Rotește robotul dacă nu s-a detectat nicio minge
    if not contours:
        self.twist.angular.z = 0.5  # Rotație la dreapta
        self.cmd_vel_pub.publish(self.twist)
    else:
        # Filtru pentru conturile mici
        valid_contours = [cnt for cnt in contours if cv2.contourArea(cnt) > self.threshold_area]

        if valid_contours:
            # Obține cel mai mare contur
            largest_contour = max(valid_contours, key=cv2.contourArea)

            # Calculează centroidul
            M = cv2.moments(largest_contour)
            cx = int(M['m10'] / M['m00'])
            cy = int(M['m01'] / M['m00'])

            # Oprește rotația și se deplasează înainte către minge
            self.twist.angular.z = 0
            self.twist.linear.x = 0.2
            self.cmd_vel_pub.publish(self.twist)
            self.detecting_ball = False  # Schimbă starea la detectarea mingei
```

c. Lovirea mingii

Odată ce robotul a detectat mingea, se deplasează către centrul ei până când o lovește. După ce acesta lovește mingea, se oprește o secundă, iar apoi reia din nou procesul de detectare a mingii.

```
else:
    # Loveşte mingea
    self.twist.linear.x = 0.5  # Se deplasează înainte
    self.cmd_vel_pub.publish(self.twist)
    rospy.sleep(1)  # Aşteaptă un moment pentru a lovi mingea
    self.stop_robot()  # Opreşte robotul după 1 secundă
    self.detecting_ball = True  # Reia detectarea mingii
```

d. Oprirea robotului după impactul cu mingea

```
def stop_robot(self):
    # Opreşte complet robotul
    self.twist.linear.x = 0
    self.twist.angular.z = 0
    self.cmd_vel_pub.publish(self.twist)
    rospy.sleep(0.1)  # Aşteaptă 0.1 secunde după impact
```

6 GHIDUL UTILIZATORULUI

Pentru punerea în funcțiune a robotului este necesar să parcurgem următoarele etape:

1. Se deschide un terminal nou unde vom executa următoarea comandă:

   $ export TURTLEBOT3_MODEL=waffle

   $roslaunch turtlebot3_gazebo turtlebot3_world.launch

2. Se introduce un obiect de tip sferă în mapa utilizată, cu ajutorul butonului Insert din Gazebo, se ajustează dimensiunea și culoarea acesteia.

3. Se deschide un al doilea terminal unde vom executa următoarea comandă:

   $ python Code.py

7 CONCLUZII

În acest laborator, am avut ocazia să folosim pentru prima dată un sistem de operare ubuntu și să ne familiarizăm cu terminalele din ROS (Robot Operating System), platforma pe care s-a bazat tema laboratorului. A fost o experiență nouă și provocatoare pentru noi, deoarece am descoperit lucruri noi și interesante, precum limbajul de programare python, terminalele, nodurile, simulatorul și altele. Acest proiect ne-a dezvoltat abilitățile de lucru în echipă, deoarece am învățat să comunicăm, să ne distribuim sarcinile în mod echitabil, să venim cu idei noi și să rezolvăm problemele apărute. În concluzie, proiectul a fost foarte complex și util, pregătindu-ne pentru proiectele viitoare.

8 BIBLIOGRAFIE

[1] https://copilot.microsoft.com/
[2] https://www.google.com/imghp?hl=en
[3] https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/#gazebo-simulation
[4] https://chat.openai.com/