 **UNIVERSITATEA “POLITEHNICA” DIN TIMISOARA**

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOAR**

**DEPARTAMENTUL AUTOMATICĂ ŞI INFORMATICĂ APLICATĂ**

**FOTBALIST CU TURTLEBOT3**

**PROIECT SINCRETIC I**

**AUTORI:**

**Pop Darius Sorin**

**Anul III**

**Coordonatori:**

**Conf.dr.ing. Florin DRĂGAN, As. ing. Emil VOIŞAN**

**Conf.dr.ing. Florin DRĂGAN, As. ing. Emil VOIŞAN**

**Anul III AIA – an universitar 2024/2025**

Cuprins

[1. Introducere 5](#_Toc188543781)

[2. Prezentarea temei 6](#_Toc188543782)

[2.1. Obiectivele proiectului 6](#_Toc188543783)

[2.2. Importanța temei 7](#_Toc188543784)

[2.3. Complexitatea tehnologică 8](#_Toc188543785)

[2.4. Rezultate așteptate 8](#_Toc188543786)

[3. Tehnologii utilizate 8](#_Toc188543787)

[3.1. Tehnologii utilizate 8](#_Toc188543788)

[3.2. Sinergia tehnologiilor 10](#_Toc188543789)

[4. Ghidul programatorului 10](#_Toc188543790)

[4.1. Introducere 10](#_Toc188543791)

[4.2. Structura generală a codului 10](#_Toc188543792)

[4.3. Fluxul logic 14](#_Toc188543793)

[4.4. Module folosite 14](#_Toc188543794)

[4.4.1. ROS 2: 14](#_Toc188543795)

[4.4.2. OpenCV: 15](#_Toc188543796)

[4.5. Variabile importante: 15](#_Toc188543797)

[4.5.1. Detecție minge: 15](#_Toc188543798)

[**4.5.2.** Detecție poartă**:** 15](#_Toc188543799)

[4.5.3. **Stare robot**: 15](#_Toc188543800)

[4.6. Cum funcționează detecția? 15](#_Toc188543801)

[4.6.1. Mingea: 15](#_Toc188543802)

[4.6.2. Poarta: 16](#_Toc188543803)

[4.7. Mișcarea robotului 16](#_Toc188543804)

[4.8. Ierarhia claselor 16](#_Toc188543805)

[4.8.1. Atribute: 16](#_Toc188543806)

[4.8.2. Metode: 17](#_Toc188543807)

[4.9. Puncte cheie pentru configurare 17](#_Toc188543808)

[4.10. Recomandări pentru extindere 17](#_Toc188543809)

[5. Ghidul utilizatorului 18](#_Toc188543810)

[5.1. Introducere 18](#_Toc188543811)

[5.2. Cerințe preliminare 18](#_Toc188543812)

[5.3. Instalare și configurare 18](#_Toc188543813)

[6. Concluzii 19](#_Toc188543814)

[Bibliografie 21](#_Toc188543815)

# Introducere

Roboții mobili reprezintă o categorie esențială în domeniul roboticii, fiind caracterizați prin capacitatea de a se deplasa într-un mediu dat pentru a îndeplini diverse sarcini. Această categorie de roboți este utilizată într-o gamă largă de aplicații, de la logistică și agricultură până la intervenții în medii periculoase sau explorare.

Una dintre provocările majore în dezvoltarea roboților mobili este integrarea procesării de imagini pentru a permite interacțiunea cu obiectele din mediu. Aceasta presupune utilizarea senzorilor și a camerelor pentru a identifica și interpreta informațiile vizuale, într-o manieră asemănătoare modului în care oamenii percep mediul înconjurător.

Un alt aspect important al roboticii moderne este conducerea la distanță a roboților, care permite utilizatorilor să controleze și să monitorizeze roboți mobili în timp real, de la locații aflate la distanță semnificativă. Această tehnologie deschide oportunități semnificative în medii periculoase, unde prezența umană nu este posibilă, și în aplicații care necesită intervenții rapide și precise.

Proiectul de față se concentrează pe simularea și controlul unui robot mobil TurtleBot3 în mediul de simulare Gazebo. Obiectivul principal al proiectului este de a dezvolta o aplicație care să permită robotului să identifice o minge folosind procesarea de imagini, să se deplaseze spre aceasta și, ulterior, să identifice o poartă pentru a marca un gol. Acest scenariu presupune integrarea unor tehnologii avansate de simulare, procesare de imagini și algoritmi de navigație.

Dezvoltarea aplicației a implicat o serie de provocări tehnice, inclusiv integrarea senzorilor virtuali, configurarea mediului de simulare și optimizarea algoritmilor pentru a îmbunătăți performanța robotului. Cu toate că aplicația nu este perfectă, aceasta reprezintă un pas important în explorarea posibilităților oferite de roboții mobili și inteligența artificială.

Introducerea proiectului oferă o bază teoretică pentru înțelegerea domeniului și a provocărilor abordate, punând accent pe relevanța dezvoltării unor astfel de soluții în contextul roboticii moderne.

# Prezentarea temei

Tema acestui proiect constă într-un scenariu dinamic și complex, în care un robot mobil TurtleBot3 trebuie să interacționeze cu un mediu virtual simulat în Gazebo, în scopul de a îndeplini un set de obiective bine definite. Această temă nu doar că testează capacitatea de integrare a tehnologiilor moderne, dar și înseamnă explorarea diverselor abordări utilizate în dezvoltarea roboților mobili autonomi. (Video, How to build a soccer robot using TurtleBot3, n.d.)

## Obiectivele proiectului

Proiectul se concentrează pe implementarea unor funcționalități cheie care pun în evidență capacitatea robotului de a percepe mediul înconjurător, de a lua decizii și de a executa sarcini autonome. Scopurile specifice includ:

1. **Identificarea mingii:**

* Folosirea algoritmilor de procesare a imaginilor pentru detectarea unei mingi, un obiect cheie în scenariu.
* Implementarea unui sistem de recunoaștere bazat pe caracteristici vizuale precum culoare, dimensiune sau formă, utilizând fluxurile video capturate de senzorii virtuali ai robotului.
* Ajustarea parametrilor algoritmilor pentru a îmbunătăți rata de succes în identificare, chiar și în condiții variate de iluminare sau unghiuri de vizualizare.

1. **Deplasarea autonomă către minge:**

* Dezvoltarea unei strategii eficiente de navigație, care implică mai întâi detecția mingii, apoi a porții prin state-uri.
* Utilizarea datelor primite de la cameră pentru a naviga spre minge și pentru a centra mingea
* Integrarea unor mecanisme de control proporțional-derivativ pentru mișcări precise, cu reglaj de viteză pentru rotația sa

1. **Identificarea și localizarea porții:**

* Duplicarea procesului de recunoaștere vizuală pentru a localiza poarta din mediul de simulare, prin formă și culoare
* Utilizarea datelor primite de la cameră pentru a naviga spre poartă cu mingea

1. **Marcarea unui gol:**

* Implementarea unei strategii de deplasare care asigură transportul eficient al mingii către poartă.
* Monitorizarea constantă a mingii și ajustarea poziției robotului pentru a compensa orice abatere sau pierdere a mingii.

## Importanța temei

Tema este semnificativă în contextul roboticii moderne, deoarece integrează mai multe aspecte tehnologice esențiale, cum ar fi:

* Procesarea de imagini: O tehnologie centrală în robotică, folosită pentru a oferi roboților capacitatea de a percepe și interacționa cu mediul.
* Navigația autonomă: Un domeniu critic pentru orice robot mobil, esențial pentru desfășurarea sarcinilor în mod eficient și sigur.
* Controlul la distanță: Facilitarea utilizării robotului în medii simulate pentru dezvoltare, cu posibilitatea extinderii acestor capacitați în medii reale.

Tema abordată oferă un cadru didactic și practic pentru înțelegerea problemelor reale și a soluțiilor utilizate în robotică. Prin explorarea și dezvoltarea acestui proiect, se pune accent pe interacțiunea dintre modulele software și hardware, utilizarea simulării pentru testare și optimizarea deciziilor autonome ale robotului.

## Complexitatea tehnologică

Această temă presupune o abordare inter-disciplinară, implicând cunoștințe din:

* **Robotică:** Pentru configurarea TurtleBot3 și gestionarea motorului de mișcare.
* **Simulare:** Utilizarea Gazebo pentru a crea un mediu virtual realist și adaptabil.
* **Programare:** Dezvoltarea algoritmilor și implementarea logicii aplicației folosind Python și ROS2.

Prin aceste componente, proiectul oferă o oportunitate excelentă de a pune în practică teorii și concepte învățate, aplicându-le într-un context practic și captivant.

## Rezultate așteptate

Finalizarea proiectului va demonstra capacitatea TurtleBot3 de a funcționa autonom într-un scenariu complex. De asemenea, se va evidenția importanța procesării eficiente a datelor senzoriale și a luării deciziilor precise pentru atingerea obiectivelor.

Pe lângă aceste rezultate, proiectul va contribui la dezvoltarea competențelor tehnice ale dezvoltatorului, cum ar fi programarea, integrarea tehnologiilor și analiza performanței.

# Tehnologii utilizate

Pentru implementarea proiectului, s-au utilizat mai multe tehnologii cheie, fiecare având un rol specific în dezvoltarea și funcționarea aplicației. Alegerea acestora a fost determinată de cerințele proiectului și de compatibilitatea lor cu sistemele utilizate.

## Tehnologii utilizate

1. **Gazebo**

Gazebo este un simulator avansat pentru roboți, care permite dezvoltarea și testarea algoritmilor într-un mediu virtual realist. Acesta a fost utilizat pentru:

* **Crearea mediului de simulare:** Generarea unui mediu virtual care include terenul, mingea și porțile.
* **Testarea algoritmilor:** Permite evaluarea funcționalității robotului într-un mod sigur și repetabil, fără riscul de a deteriora hardware-ul real. (Documentation G. , n.d.)
* **Integrarea cu ROS2:** Simulatorul Gazebo poate comunica cu ROS2, facilitând dezvoltarea rapidă a aplicației. (TurtleBot3 Soccer Actions, n.d.)

1. **ROS2 (Robot Operating System 2)**

ROS2 este un sistem de operare destinat roboților, utilizat pentru gestionarea comunicării dintre componentele robotului. Acesta a fost utilizat pentru:

* **Managementul nodurilor:** Divizarea funcționalității în module independente (noduri), cum ar fi procesarea imaginilor, navigația și controlul mișcării.
* **Publicare și abonare la topicuri:** Facilitarea comunicării între senzori, actuatori și componentele software prin topicuri definite.
* **Simulare în timp real:** Integrarea cu Gazebo pentru a transmite date și comenzi între mediul de simulare și aplicația de control. (Peyman, n.d.) (Documentation R. , n.d.)

1. **Python**

Python a fost limbajul principal utilizat pentru dezvoltarea algoritmilor. Acesta a fost ales datorită:

* **Bibliotecilor:** OpenCV pentru procesarea imaginilor.
* **Flexibilității:** Posibilitatea de a integra rapid diverse funcționalități.
* **Compatibilității cu ROS2:** Suport extins pentru dezvoltarea de noduri ROS2.

1. **Visual Studio Code**

Visual Studio Code a fost mediul de dezvoltare utilizat pentru scrierea, testarea și depanarea codului. Acesta a fost preferat datorită:

* **Suportului pentru extensii ROS:** Integrarea facilităților specifice ROS2.
* **Depanării eficiente:** Posibilitatea de a urmări și analiza comportamentul codului în timp real.
* **Interfeței intuitive:** Un mediu de dezvoltare prietenos și configurabil.

1. **OpenCV**

OpenCV a fost utilizat pentru procesarea imaginilor capturate de senzorii virtuali ai robotului. Funcționalitățile cheie includ:

* **Detectarea obiectelor:** Algoritmi pentru recunoașterea mingii și a porții.
* **Filtrare și preprocesare:** Reducerea zgomotului din imagini și creșterea preciziei detectării. (Documentation O. , n.d.)

1. **Biblioteci ROS2**: Proiectul a implicat utilizarea mai multor biblioteci ROS2 esențiale pentru funcționarea aplicației:

* rclpy: Bibliotecă pentru gestionarea nodurilor și comunicarea între ele.
* geometry\_msgs: Utilizat pentru mesaje geometrice, precum Twist, pentru controlul mișcării.
* sensor\_msgs: Pentru gestionarea mesajelor de la senzori, cum ar fi imaginile.
* cv\_bridge: Pentru conversia imaginilor între formatele ROS (sensor\_msgs/Image) și OpenCV.

1. **Senzori virtuali**

Simularea include utilizarea unor senzori virtuali precum:

* **Camera RGB:** Pentru capturarea imaginilor utilizate în procesarea vizuală.
* **LiDAR:** Pentru detectarea obstacolelor și navigație.

## Sinergia tehnologiilor

Integrarea acestor tehnologii a permis dezvoltarea unei aplicații funcționale care combină simularea, procesarea imaginilor și navigația autonomă. Alegerea acestor instrumente reflectă o balanță între performanță, compatibilitate și ușurință de utilizare, contribuind la realizarea obiectivelor proiectului într-un mod eficient.

# Ghidul programatorului

## Introducere

Codul implementează un robot mobil care detectează o minge și o poartă, se aliniază cu acestea și împinge mingea spre poartă. Logica este bazată pe o mașină de stări finite, utilizând ROS 2 pentru comunicație și OpenCV pentru procesarea imaginilor.

## Structura generală a codului

* **Clasa Node:**
  + Bază a nodurilor ROS 2, utilizată pentru crearea nodului principal al robotului.
* **Clasa CaptureSub:**
  + Nod principal care implementează logica de detectare a mingii și a porții, precum și controlul mișcării robotului.
  + Utilizează:
    - Publisher (/cmd\_vel): Controlează viteza și direcția robotului.
    - Subscriber (/camera/image\_raw): Primește imagini de la camera robotului pentru procesare.
  + Funcții principale:
    - callback: Procesează imaginile pentru a detecta mingea și poarta.
    - control\_robot: Controlează mișcarea robotului în funcție de starea curentă.

A diagram of a ball

Description automatically generated

A diagram of a ball

Description automatically generated

## Fluxul logic

Codul utilizează o mașină de stări finite pentru gestionarea comportamentului robotului.

1. **SEARCH\_BALL:**

* Robotul scanează mediul pentru a localiza mingea prin rotație.
* Dacă mingea este detectată, trece la APPROACH\_BALL.

1. **APPROACH\_BALL:**

* Robotul se aliniază cu mingea folosind coordonata cx și se apropie de aceasta.
* Odată ce mingea este centrată și aproape, trece la SEARCH\_GOAL.

1. **SEARCH\_GOAL:**

* Robotul scanează mediul pentru a localiza poarta.
* Dacă poarta este detectată, trece la ALIGN\_GOAL.

1. **ALIGN\_GOAL:**

* Robotul ajustează poziția mingii și se aliniază cu poarta.
* Dacă poziția este corectă, trece la PUSH\_BALL.

1. **PUSH\_BALL:**

* Robotul împinge mingea spre poartă.
* Dacă mingea dispare (este introdusă în poartă sau pierdută), revine la SEARCH\_BALL.

## Module folosite

### ROS 2:

rclpy: Biblioteca principală pentru nodurile ROS.

* **Mesaje:**

Twist: Controlează viteza robotului.

Image: Transmite imagini de la cameră.

### OpenCV:

* **Prelucrare imagini:**

Convertire la HSV (cv2.cvtColor).

Creare mască pentru culori (cv2.inRange).

Detectare contururi (cv2.findContours).

Dilatare și operațiuni morfologice (cv2.dilate). (Video, TurtleBot3 Navigation and OpenCV Tutorial, n.d.)

## Variabile importante:

### **Detecție minge**:

* self.ball\_detected: Indică dacă mingea este vizibilă.
* self.cx: Coordonata X a centrului mingii în imagine.
* self.area: Aria mingii detectate, folosită pentru a aproxima distanța. (Mingleeee, n.d.)

### **Detecție poartă:**

* self.goal\_detected: Indică dacă poarta este vizibilă.
* self.goal\_center\_x: Coordonata X a centrului porții.

### **Stare robot**:

* self.state: Starea curentă a robotului.

## Cum funcționează detecția?

### Mingea:

* Imaginea este convertită la HSV.
* Se creează o mască pe baza intervalului HSV pentru roșu.
* Se dilatează masca pentru a elimina zgomotul și a mări mingea.
* Contururile sunt analizate pentru a găsi mingea.

### Poarta:

* Două măști sunt create: una pentru albastru și alta pentru galben.
* Măștile sunt procesate pentru a elimina zgomotul.
* Contururile sunt analizate pentru a găsi poarta bazată pe dimensiune și raportul lățime/înălțime.

## Mișcarea robotului

* **Viteza angulară**: Controlată prin twist.angular.z pentru a ajusta orientarea robotului.
* **Viteza liniară:** Controlată prin twist.linear.x pentru deplasarea în linie dreaptă.
* Robotul reglează viteza în funcție de distanța față de minge și poartă.

## Ierarhia claselor

Clasa principală CaptureSub utilizează structura următoare:

### Atribute:

* Publisher și Subscriber (ROS):
* self.cmd\_pub: Publică comenzi de mișcare.
* self.subscription: Primește imagini de la cameră.
* Stări:
  + - self.state: Gestionează starea curentă a robotului.
    - self.ball\_detected: Confirmă detectarea mingii.
    - self.goal\_detected: Confirmă detectarea porții.
* Variabile OpenCV:
  + - self.cx: Coordonata centrului mingii.
    - self.area: Aria mingii detectate.
    - self.goal\_center\_x: Coordonata centrului porții.
    - self.goal\_area: Dimensiunea porții detectate.

### Metode:

* **\_\_init\_\_:** Inițializează nodul ROS și variabilele.
* **callback**: Procesează imaginile pentru detecții.
* **control\_robot:** Controlează mișcarea robotului.

## Puncte cheie pentru configurare

* **Intervale HSV:**
* Ajustate pentru condiții de iluminare diferite.
* Culorile pentru minge (roșu) și poartă (albastru și galben) sunt testate și calibrate manual.
* **Praguri pentru dimensiuni:**
  + Dimensiunea minimă a contururilor detectate este definită pentru a elimina zgomotul.

## Recomandări pentru extindere

* Optimizarea detectării:
  + Utilizarea unor modele de inteligență artificială pentru clasificarea obiectelor.
* Simulare avansată:
  + Extinderea mediului din Gazebo pentru teste mai complexe.
* Integrare hardware:
  + Adaptarea aplicației pentru a controla un robot fizic.

Acest ghid oferă o imagine detaliată asupra structurii și funcționării codului proiectului, facilitând înțelegerea și extinderea ulterioară a aplicației. (Construct, 2019)

# Ghidul utilizatorului

## Introducere

Acest ghid oferă instrucțiuni clare despre utilizarea aplicației care controlează un robot mobil TurtleBot3, proiectat pentru a detecta o minge, a se deplasa spre ea și a marca un gol într-o poartă. Aplicația este destinată utilizatorilor familiarizați cu mediul ROS2 și simularea Gazebo.

## Cerințe preliminare

Pentru a utiliza aplicația, sunt necesare următoarele componente:

**Software:**

* ROS2 instalat și configurat (distribuția Humble sau Foxy este recomandată).
* Gazebo pentru simularea robotului și mediului.
* Python instalat, cu pachetele necesare (OpenCV, rclpy).

**Hardware:**

* Robot TurtleBot3 (opțional, pentru testare în mediu real).
* Un PC compatibil cu ROS2 și Gazebo. (Mouhyemen, n.d.)

## Instalare și configurare

Pentru pornire, se rulează următoarele comenzi astfel:   
În primul terminal, se dau comenzile

export TURTLEBOT3 MODEL=waffle\_pi

ros2 launch turtlebot3\_gazebo empty\_world.launch.py

Odată rulat Gazebo, se va deschide simulatorul unde va apărea doar robotul. De la insert se alege terenul de fotbal denumit ROBOCup 2009 Field, apoi se va adăuga o minge tot de la insert. Amplasăm mingea undeva pe teren, apoi deschidel alt terminal.

În alt terminal, se rulează

Cd ~/turtlebot3\_ws ( aici se pune numele directorului unde este salvat ros2 si gazebo.

Colcon build (procesul trebuie sa nu afișeze erori)

Source install/setup.bash

Ros2 run tb3\_tele rotate2

După rulare, mergem în Gazebo și vedem procesul.

A screenshot of a computer

Description automatically generatedA computer screen shot of a blue and white diagram

Description automatically generated

# Concluzii

Proiectul realizat reprezintă o implementare practicată a unui sistem robotic autonom, capabil să execute sarcini complexe precum detectarea obiectelor, navigația autonomă și interacțiunea cu mediul. Utilizând tehnologii avansate precum ROS2, Gazebo și OpenCV, aplicația demonstrează integrarea eficientă a componentelor software și hardware pentru a rezolva un scenariu de tip real-world. Această experiență a fost nu doar un proces tehnic, ci și o oportunitate de a explora creativitatea și soluționarea problemelor în robotică.

**Rezultate obținute**

1. **Detectarea obiectelor:** Robotul a fost capabil să identifice mingea și poarta în medii simulate, folosind algoritmi de procesare a imaginilor. Aceasta reprezintă un pas esențial în înțelegerea modului în care roboții pot percepe mediul înconjurător.
2. **Navigația autonomă:** S-a demonstrat că robotul poate urmări și ajusta traiectoria pentru a se deplasa cu mingea către obiectiv, arătând potențialul utilizării algoritmilor de navigație în medii dinamice.
3. **Execuția sarcinilor:** Sistemul a reușit să completeze scenariul propus, de la detectarea mingii până la marcarea unui gol, demonstrând că abordarea adoptată este viabilă și eficientă.

**Provocări și lecții învățate**

* **Calibrarea parametrilor HSV:** Detectarea culorilor a necesitat ajustări fine pentru a se adapta la diverse condiții de iluminare și zgomot vizual. Această etapă a evidențiat importanța detaliilor în procesarea imaginilor.
* **Optimizarea navigației:** Procesul de navigație a implicat multe iterații pentru a găsi combinația optimă între viteză și precizie. Această experiență a subliniat importanța testărilor extensive.
* **Integrarea componentelor:** Arhitectura modulară adoptată a permis identificarea rapidă și rezolvarea problemelor, arătând că o planificare bine structurată face diferența între succes și eșec.

**Contribuții și perspective viitoare**

Proiectul contribuie la domeniul roboticii mobile prin implementarea unui sistem accesibil, scalabil și ușor de adaptat pentru scenarii similare. El reprezintă o demonstrație practicată a modului în care tehnologiile moderne pot fi utilizate pentru a dezvolta soluții reale.

Perspectivele viitoare includ:

1. Extinderea funcționalității pentru medii reale prin integrarea de senzori suplimentari precum LiDAR, care ar putea îmbunătăți semnificativ percepția mediului.
2. Optimizarea algoritmilor pentru o performanță mai mare în condiții dinamice, cu scenarii mai complexe și variate.
3. Implementarea de rețele neuronale pentru detecția mai avansată a obiectelor, oferind robotului o capacitate de percepție mai aproape de cea umană.

**Concluzie finală**

Pe parcursul acestui proiect, am descoperit că robotica nu este doar despre tehnologie, ci și despre învățarea de a privi o problemă din mai multe perspective. Fiecare provocare a adus o lecție valoroasă, iar rezultatele obținute reflectă munca, pasiunea și perseverența implicată. În primă instanță, pare un domeniul abstract și nu neapărat plăcut, dar după ce ajungi să înțelegi cum funcționează ecosistemul de programare al unui robot, lucrurile devin plăcute și merită fiecare minut investit în crearea astfel de lucruri.

Acest proiect demonstrează potențialul incredibil al tehnologiilor moderne în domeniul roboticii pentru a dezvolta soluții autonome eficiente. Chiar dacă există împrejurări care pot fi optimizate, proiectul se constituie ca un exemplu clar de progres tehnologic, cu aplicații tangibile în viața reală și cu promisiunea unor posibilități nelimitate pe viitor.

# Bibliografie

Construct, T. (2019). *ROS2 in 5 Days.* https://www.theconstruct.ai/wp-content/uploads/2019/03/ROS2-IN-5-DAYS-e-book.pdf.

Documentation, G. (fără an). *Gazebo Tutorials*. Preluat de pe https://gazebosim.org/

Documentation, O. (fără an). *OpenCV Python Tutorials*. Preluat de pe https://docs.opencv.org/

Documentation, R. (fără an). *ROS2 Basics*. Preluat de pe https://docs.ros.org/en/

Mingleeee. (fără an). *Ball Tracking Script*. Preluat de pe https://github.com/mingleeeeee/turtlebot3-soccer/blob/main/ball\_tracking.py

Mingleeeeee. (fără an). *TurtleBot3 Soccer Actions*. Preluat de pe https://github.com/mingleeeeee/turtlebot3-soccer/actions

Mouhyemen. (fără an). *ECE7785 Lab2*. Preluat de pe https://github.com/mouhyemen/ECE7785\_Lab2

Peyman, M. (fără an). *ROS TurtleBot3 OpenCV*. Preluat de pe https://github.com/moeinpeyman/ROS\_turtlebot3\_OpenCV

Video, O. (fără an). *How to build a soccer robot using TurtleBot3*. Preluat de pe https://www.youtube.com/watch?v=GSZ6z3GQk7Y

Video, O. (fără an). *TurtleBot3 Navigation and OpenCV Tutorial*. Preluat de pe https://www.youtube.com/watch?v=Rw6ATkORRG8