**Documentație pentru Proiectul Turtlebot3 Waffle**

**1. Introducere**

Proiectul **Automated Guided Vehicle** implică implementarea unui sistem autonom pentru Turtlebot3 Waffle, care se deplasează pe o bandă marcată, detectează semnele STOP și evită obstacole. Acest proiect folosește ROS2 și OpenCV pentru procesarea imaginilor, precum și senzori LiDAR pentru detectarea obstacolelor.

Scopul acestui proiect este de a dezvolta un vehicul ghidat autonom care să poată fi utilizat în scenarii educaționale, de testare și cercetare.

**2. Hardware Utilizat**

1. **Turtlebot3 Waffle**:
   * Dimensiuni compacte pentru medii interioare.
   * Senzor LiDAR 360° pentru detectarea mediului.
   * Motorizare cu motoare DC pentru mobilitate lină.
2. **Senzori**:
   * Cameră (compatibilă cu ROS2) pentru procesarea vizuală.
   * Senzor LiDAR pentru detectarea obstacolelor.
3. **Alte componente**:
   * Calculator de bord (Raspberry Pi/NVIDIA Jetson Nano).
   * Conectivitate Wi-Fi pentru control și procesare distribuită.

**3. Software Utilizat**

1. **Sistem de operare**: Ubuntu 20.04 LTS.
2. **ROS2**: Framework pentru comunicații distribuite.
3. **Pachete ROS**:
   * turtlebot3 pentru controlul robotului.
   * gazebo\_ros pentru simulări.
4. **Biblioteci adiționale**:
   * OpenCV pentru procesarea imaginilor.
   * NumPy pentru calcule numerice.
   * Scikit-learn (daca este cazul) pentru funcții de machine learning.

**4. Descrierea Funcționalității**

**4.1 Urmărirea Benzii**

* Camera captează imaginea drumului.
* Algoritmul de procesare detectează marcajele benzii utilizând transformări binare și detectarea contururilor.
* Un controller PID ajustează traiectoria robotului pentru a menține banda centrală.

**4.2 Detectarea Semnului STOP**

* Procesarea imaginilor cu OpenCV identifică semnul STOP pe baza culorii și formei.
* La detectarea semnului, robotul trimite o comandă de oprire pentru o durată prestabilită.

**4.3 Evitarea Obstacolelor**

* Datele LiDAR sunt procesate pentru a detecta obiectele din calea robotului.
* Algoritmul ajustează direcția pentru a ocoli obstacolele păstrând banda detectată.

**5. Algoritmi și Implementare**

**5.1 Detecția Liniei**

* Imaginea este convertită într-o scală de gri.
* Se aplică o transformare binară pentru a evidenția marcajele benzii.
* Algoritmul Hough detectează liniile principale.

**5.2 Detecția Semnului STOP**

* Se folosește detecția culorii pentru a identifica zonele roșii.
* Un filtru de formă validează prezența semnului.

**5.3 Controlul Mișcării**

* Controller-ul PID reglează viteza și direcția bazându-se pe erorile de aliniere a robotului.
* Comenzile de oprire și schimbare de direcție sunt integrate din fluxurile de date ale camerelor și LiDAR.

**6. Configurație ROS**

1. **Pachete ROS**:
   * group8: Noduri pentru simulare.
   * group8\_real: Noduri pentru hardware real.
2. **Fișiere Launch**:
   * turtlebot3\_world.launch.py: Lansarea simulării.
   * bringup.launch.py: Inițializarea nodurilor pentru robotul real.
3. **Topicuri**:
   * /camera/image\_raw: Flux video de la cameră.
   * /scan: Datele LiDAR pentru obstacole.
   * /cmd\_vel: Comenzi pentru mișcarea robotului.

**7. Testare și Calibrare**

**Testare:**

* **Urmărirea benzii**:
  + Testarea pe drumuri drepte și curbe.
* **Detectarea semnului STOP**:
  + Teste cu semne STOP de dimensiuni diferite.
* **Evitarea obstacolelor**:
  + Teste cu obstacole staționare și dinamice.

**Calibrare:**

* Reglarea sensibilității camerei pentru iluminare diferită.
* Ajustarea parametrilor PID pentru stabilitate.

**8. Rezultate**

* Robotul urmează banda încadrată corect pe traseu.
* Detectarea semnului STOP este realizată cu o acuratețe de 95%.
* Obstacolele sunt evitate cu succes, menținând siguranța deplasării.

**9. Concluzii și Îmbunătățiri**

**Concluzii:**

* Sistemul este robust în medii simulate și reale.

**Îmbunătățiri:**

* Optimizarea timpului de procesare.
* Integrarea cu semaforizare inteligentă.