Zadanie 3

The `mode.sh` command is used to set the mode of a system. The command takes a single argument, `X`, which represents the desired mode.

- OFF: 0
- MANUAL: 1
- AUTO: 2

```
bash mode.sh 0
```

```
bash mode.sh 1
```

```
bash mode.sh 2
```

Startup

```
ros2 run control_package main
```

```
ros2 run control_package auto
```

```
ros2 run control_package manual
```

```
ros2 launch control_package robot_launch.py
```

Dokumentácia pre riadiaci systém robota Kobuki

Prehľad Projekt zahŕňa vývoj riadiaceho systému pre robot Kobuki, ktorý môže fungovať v simulovanom prostredí s použitím ROS2. Riadiaci systém je navrhnutý tak, aby umožnil robotu prepínať medzi troma hlavnými režimami: vypnutý režim, manuálny režim a automatický režim.

Systémové režimy

0. Vypnutý režim

- Počiatočný stav programu.
- Robot neobdrží žiadne riadiace príkazy.
- Pri prechode z akéhokoľvek režimu do vypnutého režimu sa robot zastaví a jeho stav sa overí, aby sa zabezpečilo, že je bezpečné ho vypnúť.

1. Manuálny režim

- Robot je možné ovládať pomocou manuálnych vstupov na úpravu jeho lineárnych a uhlových rýchlostí.
- Poskytuje sa možnosť okamžitého zastavenia.
- Pri prechode z automatického do manuálneho režimu sa existujúce rýchlosti upravujú hladko. Napríklad, ak je aktuálna lineárna rýchlosť 0,5 m/s a znižuje sa o 0,1 m/s, nová rýchlosť bude 0,4 m/s.

2. Automatický režim

- Robot autonómne naviguje tak, aby sa vyhýbal prekážkam pri zachovaní prednastavenej vzdialenosti od stien pomocou LIDARu.
- Robot dokáže zvládnuť steny nepravidelných tvarov a rohy bez potreby konkrétneho smeru otáčania.
- Je nevyhnutné, aby robot nenarazil do žiadnych stien.

Implementačné detaily

ROS2 topics:

- `/mode` Výber režimu je riadený cez špecifický ROS2 topic, ktorý možno aktualizovať počas behu simulácie. `ros2 topic pub /mode std_msgs/msg/Int32 "data: 0" --once`

- Vzdialenosť, ktorú robot prešiel, je neustále publikovaná do ROS2 topicu a zobrazovaná pomocou Matplotlibu pre monitorovanie v reálnom čase.

Senzory:

Na detekciu stien a meranie vzdialenosti sa využíva LIDAR.

Logika riadenia

Zdrojový kód je rozdelený do viacerých súborov zodpovedajúcich rôznym funkcionalitám:

- `MotionActions.hpp`: Definuje akcie súvisiace s riadením pohybu a ich kódy.
- `AutoController.cpp`: Implementuje logiku pre automatickú navigáciu.
- `ManualController.cpp`: Spravuje manuálne vstupy na ovládanie.
- `KobukiController.cpp`: Centrálny kontrolór pre interakciu s ROS2.
- `MotionController.cpp`: Koordinuje rôzne pohybové ovládania a režimy.

`isInSafeDistance()`

- Skontroluje, či niektoré z meraní vzdialenosti z LIDARu sú menšie alebo rovné zadanému bezpečnému limitu (0.7 metra ako prednastavená hodnota).
- Vráti true, ak aspoň jedna z vzdialeností je v nebezpečnej blízkosti, inak false.

`isInSafeDistanceInRange()`

- Overí, či niektoré z meraní vzdialenosti v zadanom vektore sú menšie alebo rovné vnútornej bezpečnej vzdialenosti.
- Vráti true, ak aspoň jedna vzdialenosť je príliš blízko, inak false.

`getMiddleValue()`

- Získa strednú hodnotu zo zoznamu hodnôt.
- Vráti hodnotu v strede zadaného vektora.

`selectMiddlePointsInRange()`

- Vyberie hodnoty okolo stredového bodu zoznamu, pridáva zadaný počet bodov na každú stranu stredového bodu (rangePadding = 4).
- Vráti vektor týchto vybraných bodov.

`getAverage()`

- Vypočíta priemernú hodnotu zo zoznamu čísel.
- Vráti priemernú hodnotu všetkých čísel v zadanom vektore.

`logLastRanges()`

- Ukladá informácie o posledných meraniach vzdialenosti z LIDARu pre ľavú a pravú stranu.
- Udržiava záznamy iba posledných 50 hodnôt.

`isMoreThan80PercentTrue()`

- Skontroluje, či viac ako 60% hodnôt v zadanom vektore booleovských hodnôt je true.
- Vráti true, ak je viac ako 60% hodnôt true, inak false.

Vizualizácia - neimplementované

Graf zobrazujúci prejdenú vzdialenosť robotom sa generuje a aktualizuje v reálnom čase, čím poskytuje vizuálnu spätnú väzbu o pohybe robota počas simulácie.