Projekt

Roboty mobilne

Nawigacja robota Pioneer 3-DX z systemem detekcji kolizji w symulacji

Michał Trela 259312

Jan Masłowski 258962

Prowadzący:
Dr inż. Michał Błędowski

Katedra Cybernetyki i Robotyki
Wydziału Elektroniki, Fotoniki i Mikrosystemów
Politechniki Wrocławskiej

18 maja 2023

1 Cel projektu

Stworzenie symulacji służącej do nawigacji robota, systemu detekcji kolizji oraz mapowania obszaru na podstawie czujników ultradźwiękowych.

2 Założenia projektowe

Projekt polega na stworzeniu symulacji umożliwiającej teleoperację oraz autonomiczną operację robota typu Pioneer 3-DX oraz obsługę LIDAR'u w celu stworzenia systemu detekcji kolizji oraz mapowania obszaru. Całość projektu zrealizowana będzie za pomocą frameworka ROS2. Stworzone oprogramowanie oraz dokumentacja będą udostępniane za pomocą systemu kontroli wersji Git.

3 Podział pracy

- Konfiguracja i uruchomienie symulacji, teleoperacja, obsługa LIDAR'u i mapowanie obszaru - Michał Trela
- Stworzenie modelu, autonomia, system detekcji kolizji- Jan Masłowski

4 Etapy projektu

Etap I:

- 23.03 - Założenia projektowe

Etap II:

- Przegląd literatury oraz zasobów internetowych [1] [2]
- 13.04 - Stworzenie modelu
- 20.04 - Konfiguracja i uruchomienie symulacji
- 11.05 - Teleoperacja

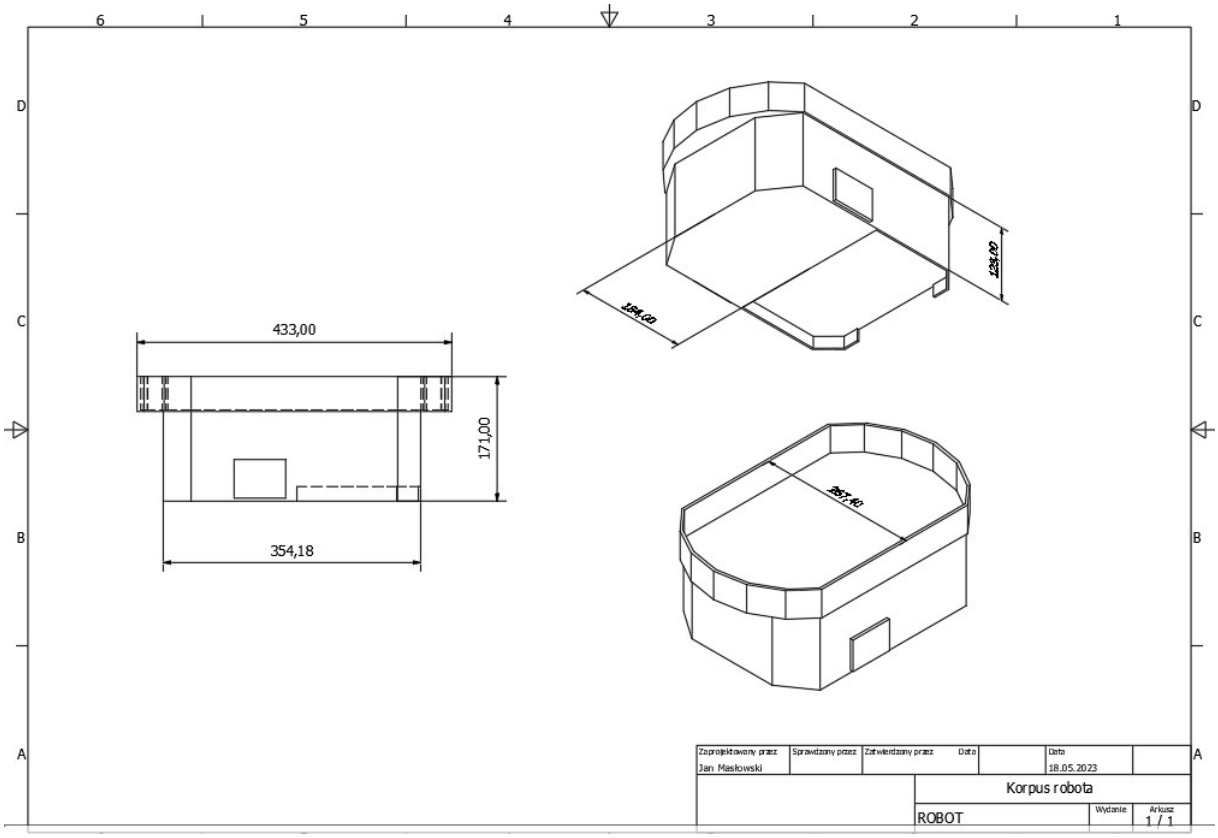
Etap III:

- 01.06 - Autonomia
- 08.06 - Obsługa LIDAR'u i mapowanie obszaru
- 22.06 - System detekcji kolizji

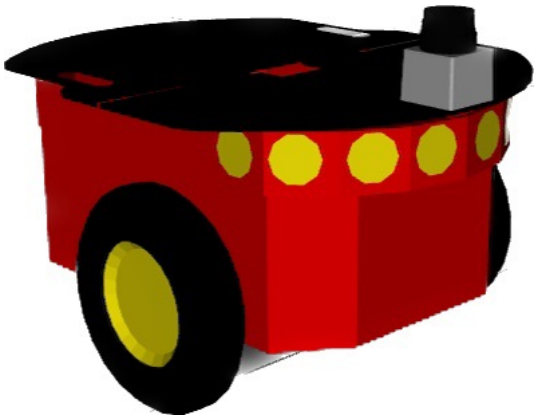
5 Osiągniecia – Etap II

5.1 Model Robota

Model robota Pioneer 3-DX został odwzorowany na podstawie modelu znalezionego w internecie. Wymiary robota zostały pobrane z ogólnodostępnych siatek, a części stworzone w programie CAD – Inventor.



Rysunek 1: Rysunek techniczny korpusu robota.



Rysunek 2: Wygląd robota w środowisku symulacyjnym Gazebo.

Model został złożony w formacie URDF (Unified Robotics Description Format) umożliwiając odpowiednią konfigurację w ROS2.

5.2 Konfiguracja i uruchomienie symulacji

Robot Pioneer jest robotem o napędzie różnicowym co wymagało skonfigurowania odpowiednich narzędzi w celu poprawnego sterowania robotem. Wykorzystana do tego została paczka ros2_control. Napisany został plik konfiguracyjny przekazujący odpowiednie parametry. na bazie odpowiednich parametrów uruchamiany jest nadajnik stanu przegubów oraz kontroler napędu różnicowego

```yaml
controller_manager:
  ros__parameters:
    update_rate: 50  # Hz
    use_sim_time: true

    joint_state_broadcaster:
      type: joint_state_broadcaster/JointStateBroadcaster

    diff_drive_controller:
      type: diff_drive_controller/DiffDriveController

diff_drive_controller:
  ros__parameters:
    left_wheel_names: ['left_wheel_joint']
    right_wheel_names: ['right_wheel_joint']

    wheel_separation: 0.35
    wheel_radius: 0.1

    wheel_separation_multiplier: 1.0
    left_wheel_radius_multiplier: 1.0
    right_wheel_radius_multiplier: 1.0

    open_loop: false
    enable_odom_tf: true

    cmd_vel_timeout: 0.6
    use_stamped_vel: false

    publish_rate: 50.0
```

Rysunek 3: Część pliku konfiguracyjnego

Następnie środowisko symulacji Gazebo wymagało odpowiedniego skonfigurowania pluginów. Należało ustawić odpowiednie interfejsy poleceń dla kół jezdnych oraz interfejsy stanu dla kół jezdnych oraz kół obrotowych.

```
<xacro:macro name="differential_drive" params="name">
  <ros2_control name="GazeboSystem" type="system">
      <hardware>
          <plugin>gazebo_ros2_control/GazeboSystem</plugin>
      </hardware>
      <joint name="left_wheel_joint">
          <command_interface name="velocity">
              <param name="min">-10</param>
              <param name="max">10</param>
          </command_interface>
          <state_interface name="velocity"/>
          <state_interface name="position"/>
      </joint>
      <joint name="right_wheel_joint">
          <command_interface name="velocity">
              <param name="min">-10</param>
              <param name="max">10</param>
          </command_interface>
          <state_interface name="velocity"/>
          <state_interface name="position"/>
      </joint>
      <joint name="swivel_joint">
          <state_interface name="velocity"/>
          <state_interface name="position"/>
      </joint>
      <joint name="swivel_wheel_joint">
          <state_interface name="velocity"/>
          <state_interface name="position"/>
      </joint>
  </ros2_control>

  <gazebo>
      <plugin name="gazebo_ros2_control" filename="libgazebo_ros2_control.so">
          <parameters>$(find pioneer3dx_description)/config/pioneer_control.yaml</parameters>
      </plugin>
  </gazebo>
</xacro:macro>

<xacro:macro name="joint_state_publisher" params="name">
  <gazebo>
  <plugin name="gazebo_ros_joint_state_publisher"
      filename="libgazebo_ros_joint_state_publisher.so">
```

Rysunek 4: Konfiguracja pluginu

5.3 Teleoparacja

Do teleoperacji wykorzystywana jest paczka teleop_twist publikująca odpowiednie dane na topic zadając prędkości na koła.

```
linear:
  x: 0.5
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
linear:
  x: 0.5
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
```

Rysunek 5: Wiadomości publikowane na topic \cmd_vel

Publikowane wiadomości są odbierane przez kontroler napędu różnicowego, który następnie zadaje prędkości na koła

Literatura

[1] Marco Matteo Bassa. *A very informal journey through ROS 2*. Marco Matteo Bassa, 2023.

[2] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022.