



**Politechnika
Śląska**

Projekt
Bezzałogowe Obiekty Autonomiczne

Symulacja robota Unitree Go2 w środowisku ISSAC SIM

Skład sekcji: Filip Bazarnicki, Darian Bonk, Jakub Ligenza, Kacper Wach
Rok akademicki: 2024/25

Kierunek: Automatyka i Robotyka

Specjalizacja: Robotyka

Semestr: 1

1 Cel projektu

Celem projektu była instalacja środowiska ISSAC SIM oraz jego odpowiednia konfiguracja, przygotowanie cyfrowego bliźniaka robota Unitree Go2, przeprowadzenie przykładowych symulacji w ISAAC SIM oraz nawiązanie połączenia między nim, a środowiskiem ROS2.

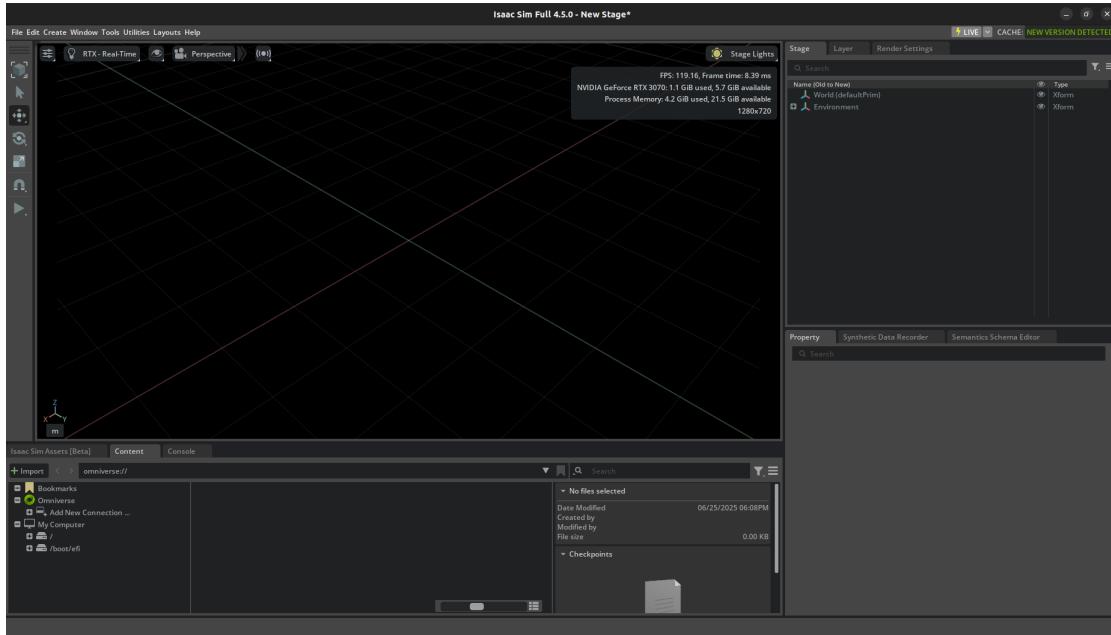
2 Wstęp teoretyczny

2.1 Czym jest środowisko ISAAC SIM

NVIDIA Isaac Sim to zaawansowane środowisko symulacyjne oparte na silniku Omniverse, stworzone z myślą o rozwoju i testowaniu algorytmów dla robotyki. Umożliwia realistyczną symulację fizyki, czujników oraz środowisk 3D, co pozwala na szybkie prototypowanie i walidację systemów autonomicznych bez konieczności fizycznego dostępu do sprzętu.

Symulator ten wspiera modelowanie robotów zgodne z formatem URDF (Unified Robot Description Format) oraz pozwala na integrację z popularnymi bibliotekami takimi jak ROS/ROS2, PyTorch, czy Isaac SDK. Dzięki realistycznemu odwzorowaniu działania sensorów (np. kamer RGB-D, LiDAR, IMU), możliwe jest testowanie algorytmów percepji, lokalizacji, mapowania, planowania ruchu czy sterowania w warunkach zbliżonych do rzeczywistych.

Środowisko to odgrywa kluczową rolę w symulacji „cyfrowych bliźniaków” robotów, co znacząco obniża koszty i ryzyko związane z rzeczywistymi testami. Widok po uruchomieniu środowiska jest przedstawiony na rysunku 1



Rys. 1: Środowisko ISAAC SIM

2.2 Czym jest środowisko ROS2

ROS 2 (Robot Operating System 2) to otwartoźródłowy system operacyjny dla robotów nowej generacji, zaprojektowany jako następca klasycznego ROS 1. Stanowi zaawansowaną platformę programistyczną do tworzenia, testowania i wdrażania oprogramowania dla systemów robotycznych, umożliwiając modularne projektowanie aplikacji z wykorzystaniem tzw. węzłów (nodes), które komunikują się ze sobą poprzez tematy (topics), usługi (services) oraz akcje (actions).

Jedną z kluczowych zmian w ROS 2 względem poprzednika jest zastosowanie standardu DDS (Data Distribution Service) jako warstwy komunikacyjnej, co zapewnia deterministyczne i wydajne przesyłanie danych między węzłami – również w sieciach rozproszonych.

ROS 2 współpracuje z popularnymi narzędziami robotycznymi i symulatorami (np. Gazebo, Isaac Sim, MoveIt 2), wspiera wiele języków programowania (C++, Python) i znajduje zastosowanie w robotyce przemysłowej, mobilnej, medycznej i badawczej.

2.3 Czym jest robot Unitree Go2

Unitree Go2 EDU to zaawansowany robot mobilny o czterokończynowej konstrukcji, zaprojektowany przez firmę Unitree Robotics – jednego z liderów w dziedzinie robotyki kroczącej. Model ten stanowi wersję edukacyjną (EDU), oferując rozszerzone możliwości programistyczne i integracyjne względem wersji konsumenckiej. Dzięki otwartemu środowisku programowania, rozbudowanemu układowi sensorycznemu oraz dużej mobilności, znajduje zastosowanie w edukacji wyższej, pracach badawczo-rozwojowych, a także w zadaniach inspecyjnych i eksperymentalnych.

Korpus robota wykonano z lekkich i wytrzymałynych materiałów – głównie stopów aluminium i włókien węglowych. Robot porusza się na czterech kończynach o trzech stopniach swobody ka da (12 DOF), napędzanych przez wydajne, bezszczotkowe silniki z czujnikami momentu. Taka konfiguracja zapewnia płynny, dynamiczny chód z możliwością dostosowania kroku do zmiennych warunków terenowych, a także wykonywanie złożonych manewrów, takich jak omijanie przeszkód, wchodzenie po schodach czy skoki.

W wersji EDU robot wyposażony jest w szereg czujników, umożliwiających percepcję otoczenia i autonomiczne działanie. W skład systemu sensorycznego wchodzą:

- Kamera głębokości (np. Intel RealSense)
- 9-osiowy IMU (żyroskop + akcelerometr + magnetometr)
- Enkodery położenia i momentu
- LIDAR 2D/3D (opcjonalnie)
- Mikrofony i czujniki dźwięku
- Opcjonalna kamera RGB i termowizyjna

Sercem systemu sterowania jest minikomputer klasy NVIDIA Jetson Orin lub komputer przemysłowy z systemem Linux Ubuntu, z preinstalowanym środowiskiem ROS 2 (Robot Operating System). Umożliwia to rozwój własnych algorytmów z zakresu SLAM, autonomicznej nawigacji, rozpoznawania obiektów, oraz komunikacji człowiek–robot.

Robot może być sterowany na trzy sposoby:

- Zdalnie, za pomocą dedykowanego kontrolera lub aplikacji mobilnej.
- Z poziomu komputera, przez połączenie Wi-Fi, Bluetooth lub Ethernet.

- Autonomicznie, poprzez zaprogramowane algorytmy w ROS 2.

Do dyspozycji użytkownika oddano interfejsy API w językach Python i C++, umożliwiające pełne sterowanie ruchem, przetwarzaniem danych z czujników i integrację z dodatkowymi modułami.

Unitree Go2 EDU znajduje zastosowanie w szerokim zakresie dziedzin technicznych i naukowych, w tym:

- Robotyka mobilna i biomechanika – modelowanie ruchu, optymalizacja lokomocji, analiza chodu
- Nawigacja autonomiczna i SLAM – mapowanie środowiska i eksploracja przestrzeni
- Widzenie maszynowe i sztuczna inteligencja – rozpoznawanie gestów, twarzy, przeszkód
- Przemysł i inspekcja – patrolowanie zakładów przemysłowych, kontrola infrastruktury technicznej
- Działania ratunkowe i wojskowe – misje w środowiskach nieprzyjaznych człowiekowi (gruzowiska, strefy skażone)
- Edukacja i dydaktyka – nauczanie sterowania, analizy danych sensorycznych, projektowania systemów embedded

Specyfikacja techniczna:

- **Wymiary (wysokość x szerokość x długość):** 70x31x40 cm (pozycja stojąca), 76x31x20 cm (pozycja kucająca)
- **Waga (z baterią):** Około 15 kg
- **Materiał:** Stop aluminium + wysoka wytrzymałość plastiku inżynierskiego
- **Napięcie robocze:** 28V $\hat{3}$ 3.6V
- **Moc szczytowa:** Około 3000W
- **Udźwig:** Około 8 kg (maks. 12 kg)
- **Prędkość:** 0 ~ 3.7 m/s (maks. 5 m/s)
- **Maksymalna wysokość podjazdu:** Około 16 cm
- **Kąt wspinania:** 40°
- **Maksymalny moment obrotowy:** Około 45 N·m
- **Liczba stawów motorycznych:** 12 zestawów
- **Zakres ruchu:** Ciało: -48° ~ 48°, Udo: -200° ~ 90°, Goleni: -156° ~ -48°
- **Bateria:** Standardowa 8000 mAh, długowieczna 15000 mAh
- **Czas pracy:** Około 2–4 godziny
- **Ładowanie:** Szybkie ładowanie (33.6V 9A)
- **Procesor:** Nvidia Jetson Orin (opcjonalny)

- **Łączność:** Wi-Fi 6 (Dual-band), Bluetooth 5.2
- **Bateria:** Inteligentna bateria, zabezpieczona przed przegrzaniem, przeciążeniem oraz zwarciem
- **Inne cechy:** Głośnik do odtwarzania muzyki

Funkcje i urządzenia robota:

- Moduł śledzenia: Zdalne sterowanie lub automatyczne śledzenie obiektów
- Mikrofon interkomowy: Umożliwia komunikację z robotem bez ograniczeń scenariuszowych
- Kamery i sensory:
 - Kamera przednia: Transmisja obrazu w rozdzielcości 1280x720, pole widzenia 120°, szeroki kąt widzenia
 - LIDAR 4D L1: Skanning 360° w zakresie 90°, omijanie przeszkód z minimalnymi martwymi strefami
- Motory precyzyjnych stawów: 12 silników z aluminium zapewnia mocne, precyzyjne ruchy
- Czujniki siły w stopach: Monitorowanie sił działających na nogi robota w czasie rzeczywistym, co pozwala na dynamiczną adaptację do terenu
- Przenośny pasek samonawijający: Umożliwia łatwe przenoszenie robota oraz załadunek
- Głośnik do odtwarzania muzyki: Odtwarzanie muzyki lub dźwięków w trakcie pracy robota

Unitree Go2 EDU to wyjątkowe narzędzie dydaktyczne i badawcze, umożliwiające eksperymentowanie z nowoczesnymi technikami robotyki mobilnej, widzenia komputerowego oraz sztucznej inteligencji. Dzięki swojej modułowości, wszechstronności i kompatybilności z ROS 2, robot ten jest szczególnie ceniony przez uczelnie techniczne, laboratoria R&D oraz przemysł innowacyjny.



Rys. 2: Przedstawia robota Unitree Go2.

3 Instalacja i konfiguracja środowiska ISSAC SIM oraz ROS2

W projekcie korzystaliśmy z systemu Ubuntu 22.04, wykorzystaliśmy system Linux ze względu na potrzebę wykorzystania środowiska ROS2 w projekcie, co jest prostsze w systemach linuxowych.

3.1 Instalacja środowiska ISAAC SIM

Ponieważ nie da się już pobrać ISAAC SIM bezpośrednio przy użyciu Omniverse Launcher, pobraliśmy ręcznie archiwum ze strony NVIDIA:

- Strona: <https://developer.nvidia.com/isaac-sim>
- Wersja: Isaac Sim 4.5.0

Po pobraniu rozpakowaliśmy plik i możliwe było już uruchomienie środowiska ISAAC SIM. Można to zrobić, wchodząc do katalogu, w którym umieszczone zostały rozpakowane pliki i wpisując ./isaac-sim.sh lub alternatywnie isaac-sim.selector.sh.

3.2 Instalacja środowiska ROS2

Źródłem wykonanej instalacji była oficjalna dokumentacja ROS 2 dla wersji Humble (Ubuntu Development Setup):

<https://docs.ros.org/en/humble/Installation/Alternatives/Ubuntu-Development-Setup.html>

3.2.1 Krok 1: Konfiguracja lokalizacji UTF-8

Na początku upewniliśmy się, że system ma skonfigurowane środowisko lokalizacyjne obsługujące UTF-8.

```
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8
```

Sprawdziliśmy, czy ustawienia lokalne są prawidłowe:

```
locale
```

3.2.2 Krok 2: Instalacja zależności systemowych

Zainstalowaliśmy wymagane pakiety narzędziowe i komplilacyjne:

```
sudo apt update && sudo apt install -y
build-essential cmake git wget curl
gnupg lsb-release python3-colcon-common-extensions
python3-pip python3-vcstool python3-rosdep
```

Zainicjowaliśmy narzędzie rosdep:

```
sudo rosdep init
rosdep update
```

3.2.3 Krok 3: Utworzenie przestrzeni roboczej i pobranie kodu ROS 2

Stworzyliśmy katalog roboczy i pobraliśmy plik repozytoriów:

```
mkdir -p ~/ros2_humble/src  
cd ~/ros2_humble  
wget https://raw.githubusercontent.com/ros2/ros2/humble/ros2.repos  
vcs import src < ros2.repos
```

3.2.4 Krok 4: Instalacja zależności pakietów ROS 2

Zainstalowaliśmy wszystkie zależności wymagane do komplikacji (pomijając rti-connext, którego nie potrzebujemy).

```
rosdep install -from-paths src -ignore-src -r -y  
-skip-keys "fastcdr rti-connext-dds-6.0.1 urdfdom_headers"
```

3.2.5 Krok 5: Kompilacja ROS 2

Rozpoczęliśmy komplikację całego środowiska z wykorzystaniem colcon:

```
cd ~/ros2_humble  
colcon build -symlink-install
```

Ten proces trwał kilkanaście minut.

3.2.6 Krok 6: Konfiguracja środowiska

Po zakończeniu komplikacji dodaliśmy ROS 2 do zmiennych środowiskowych:

```
echo "source ~/ros2_humble/install/setup.bash" >> ~/.bashrc  
source ~/.bashrc
```

Dzięki temu komenda ros2 była dostępna z każdego terminala.

3.2.7 Weryfikacja działania ros2

Przeprowadziliśmy podstawowy test komunikacji między dwoma węzłami:

Terminal 1:

```
ros2 run demo_nodes_cpp talker
```

Terminal 2:

```
ros2 run demo_nodes_cpp listener
```

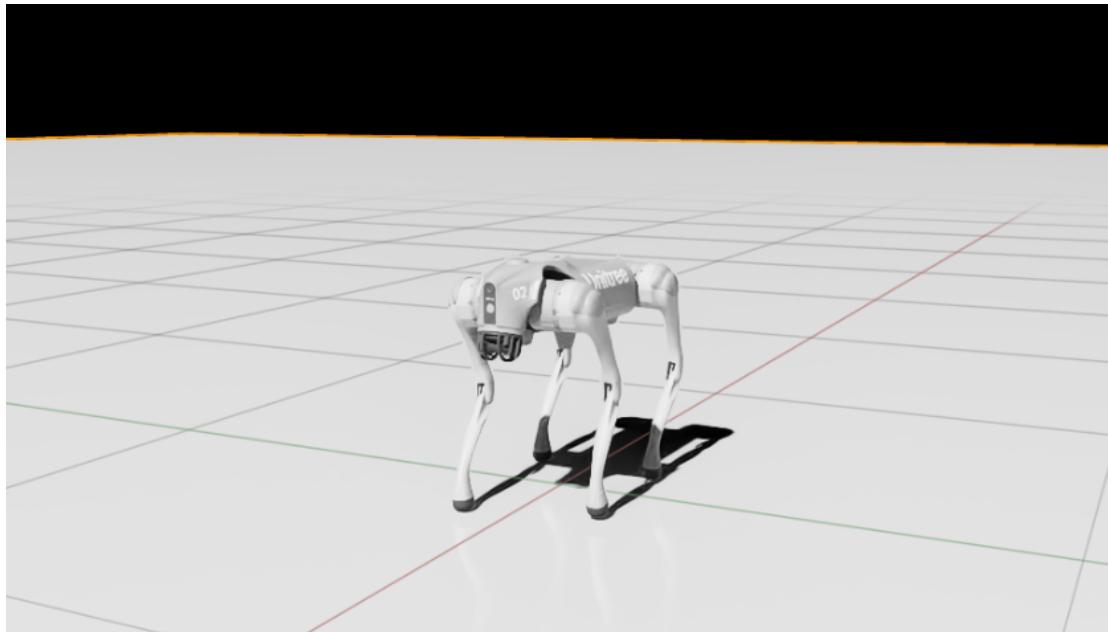
Terminal listener poprawnie odbierał wiadomości wysyłane przez talker, co potwierdziło, że środowisko działa.

4 Cyfrowy bliźniak robota Unitree Go2

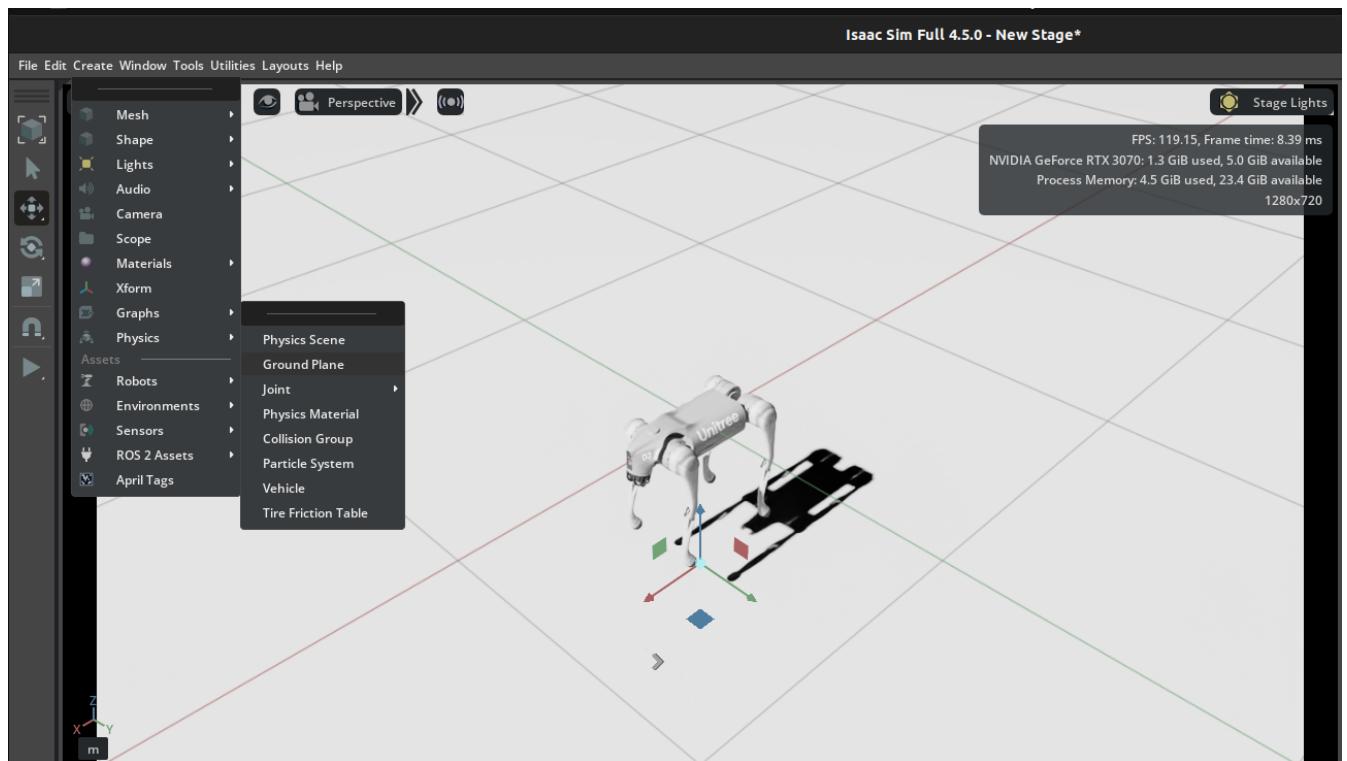
W środowisku ISSAC SIM znajduje się gotowy model cyfrowego bliźniaka robota Unitree Go2, w celu umieszczenia go na scenie otwartego programu należy odnaleźć go w oknie Isaac Sim Assets i dodać go, wciskając przycisk Load as Reference lub przeciągając jego ikonę na scenę. Widok okna Isaac Sim Assets przedstawiony jest na rysunku 3. Widok umieszczonego w symulacji robota przedstawiony jest na rysunku 4. Na rysunku 5 pokazane jest, jak dodać podłogę w scenie, należy wybrać w górnym pasku Create, następnie Physics, a na końcu wcisnąć Ground Plane.



Rys. 3: Biblioteka dostępnych robotów Isaac Sim Assets.



Rys. 4: Cyfrowy bliźniak robota Unitree Go2 w środowisku ISAAC SIM.



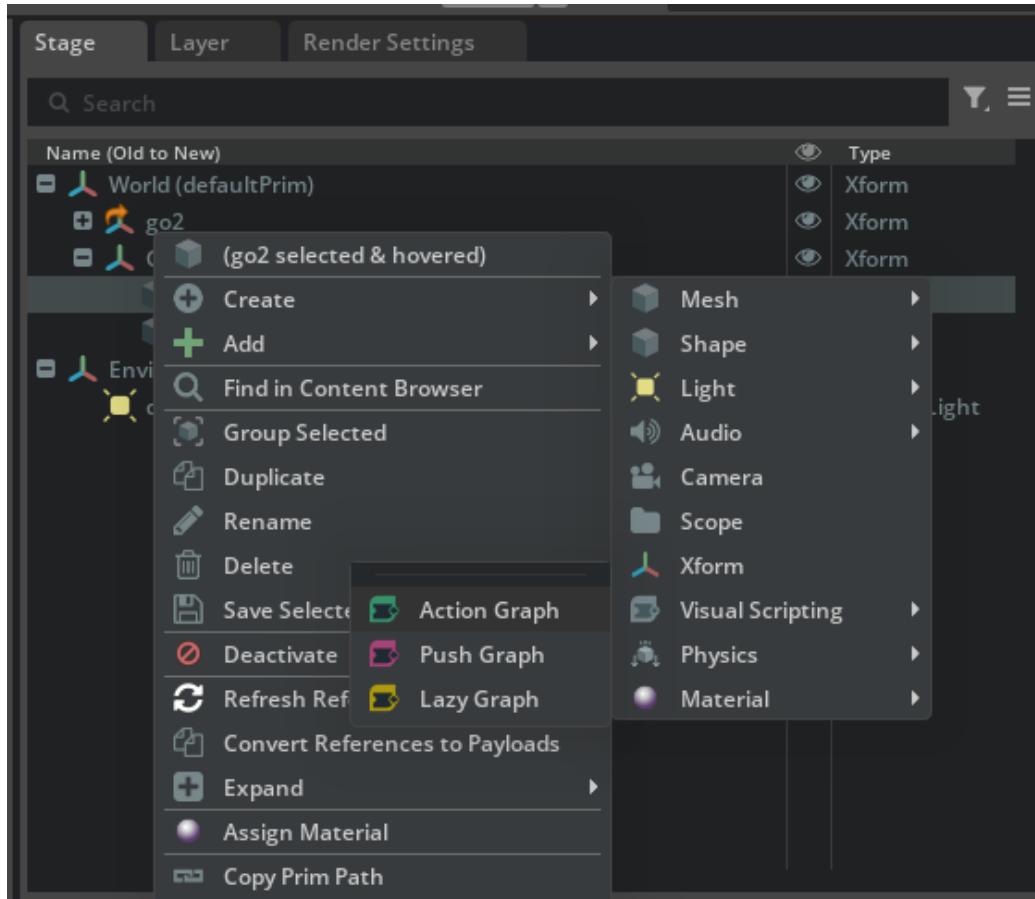
Rys. 5: Dodanie podłoża w ISSAC SIM.

5 Sterowania robotem przy pomocy środowiska ROS2

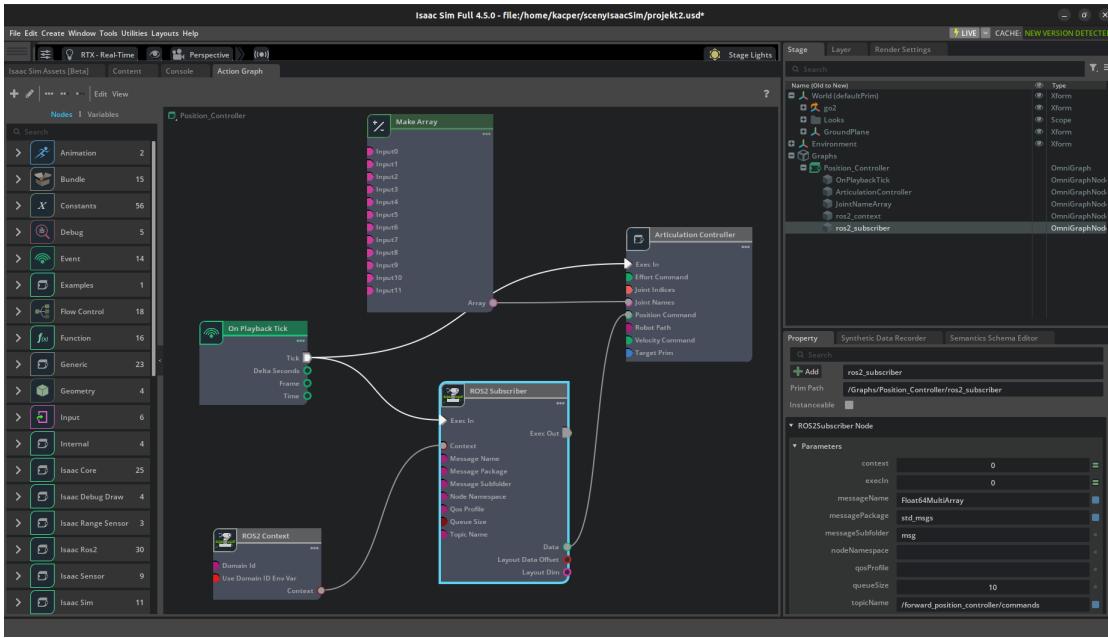
Wszystkie potrzebne pliki źródłowe do sterowania sterowaniem robotem znajdują się w linku poniżej.

<https://drive.google.com/drive/folders/1fdtBgbXUCxln3wdQgWUAGIBdBf-2Qv9z>

Żeby możliwe było sterowanie robotem Unitree Go2 przy pomocy środowiska ROS2, wymagane jest dodanie do projektu specjalnego grafu z instrukcjami w formie bloków (Action Graph). Na rysunku 6 pokazane jest jak dodać Action Graph, na rysunku 7 widoczne są potrzebne do sterowania robotem bloki oraz połączenia między nimi.



Rys. 6: Dodanie Action Graph do cyfrowego bliźniaka Unitree Go2



Rys. 7: Action Graph do nawiązania komunikacji między ROS2 oraz ISAAC SIM.

Blok On Playback Tick uruchamia się w każdej klatce symulacji i pozwala na wykonywanie się pozostałych bloków synchronicznie. Blok ROS2 Context inicjalizuje ROS2 w środowisku ISAAC SIM. Articulation Controller służy do przekazywania danych do poszczególnych członów robota. W bloku Make Array należy wstawić poszczególne nazwy członów robota, nazwy te można odczytać z obiektu robota go2. Blok Make Array przekazuje nazwy do bloku Articulation Controller po to żeby wiadomo było jakie człony mają być wysterowane na jaką wartość. Wstawione nazwy członów w bloku Make Array zostały pokazane na rysunku 8 ważne jest żeby kolejność nazw członów była taka sama jak na rysunku, ze względu na fakt, że w takiej kolejności są odbierane z bloku ROS2Subscribe. Kolejność członów w bloku Make Array musi zgadzać się z kolejnością członów w kodzie przedstawioną na rysunku 9

Blok ROS2Subscriber służy do otrzymywania danych z programu środowiska ROS2, tak zwanego subskrybowania danych, na rysunku 10 pokazano jak poprawnie skonfigurować blok ROS2Subscriber żeby dane były odpowiednio pobierane. Otrzymane dane są przesyłane do bloku Articulation Controller.

W celu uruchomienia programu sterującego, po wcześniejszym wypakowaniu potrzebnych plików, należy przejść do folderu `/quadruped_robot_ROS2` i w terminalu wpisać (żeby uruchomić terminal w danym folderze, należy, będąc w aplikacji pliki w danym folderze, wcisnąć prawy przycisk myszy i ‘otwórz w terminalu’):
`colcon build`

Następnie w nowym terminalu wpisać:

```
source quadruped_robot_ROS2/install/setup.bash
ros2 launch robot_control robot_control.launch.py
```

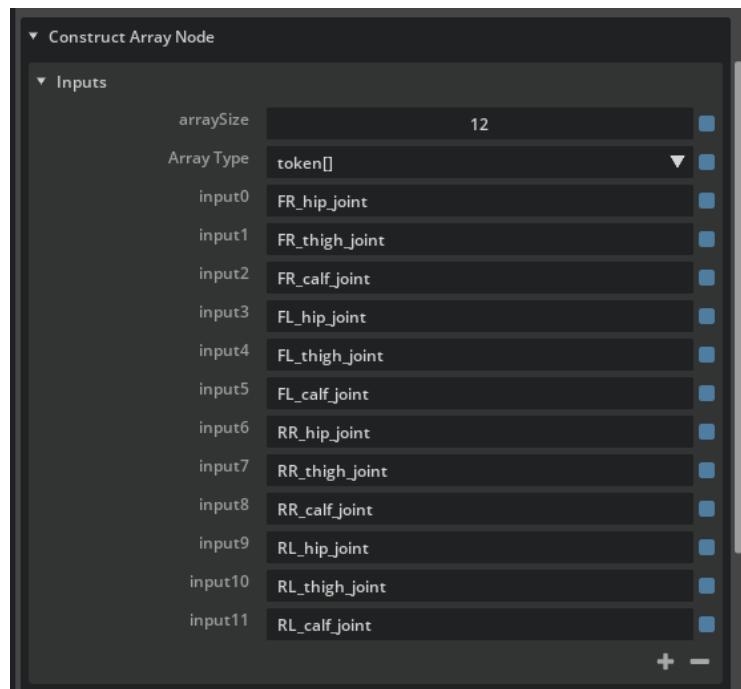
Po czym przechodzi się do folderu `/quadruped_robot_ROS2/UI` uruchamia się nowy terminal w tym folderze i wpisuje:

```
python3 controller.py
```

Na rysunku 11 oraz na rysunku 12 widoczne jest przykładowe sterowanie cyfrowy bliźniakiem robota Unitree Go2. Sterowanie odbywa się przy pomocy okna z kontrolerem widocznego na obu rysunkach, okno ma nazwę controller.

Do wykonania tej części projektu korzystaliśmy z poniższego filmu.

[Link do filmu](#)



Rys. 8: Ustawienia bloku Make Array.

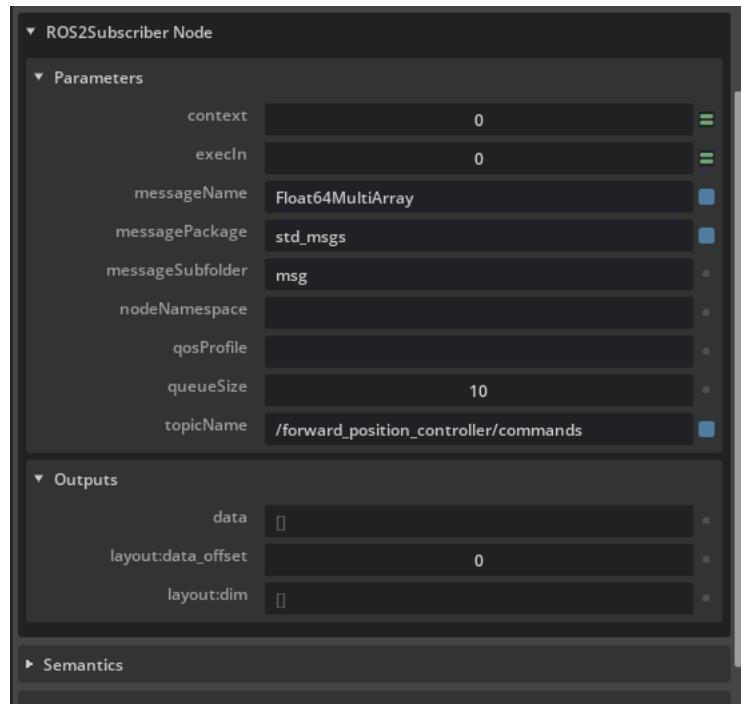
```
controller_manager:
  ros__parameters:
    update_rate: 1000 # Hz

  forward_position_controller:
    type: forward_command_controller/ForwardCommandController

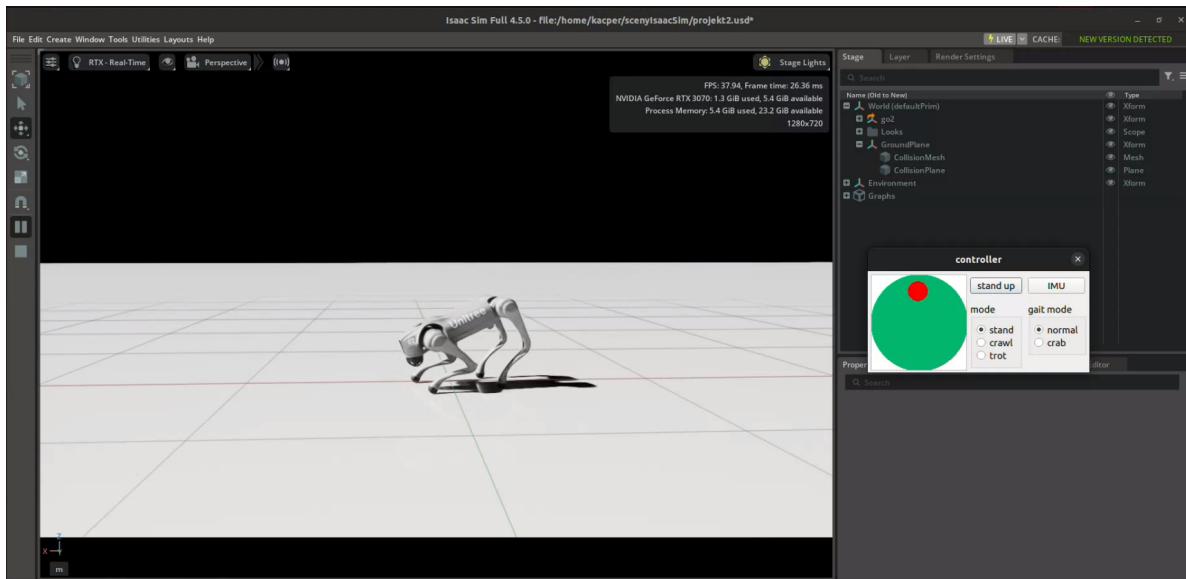
  joint_state_broadcaster:
    type: joint_state_broadcaster/JointStateBroadcaster

forward_position_controller:
  ros__parameters:
    joints:
      - front_right_leg_joint
      - front_right_thigh_joint
      - front_right_shin_joint
      - front_left_leg_joint
      - front_left_thigh_joint
      - front_left_shin_joint
      - rear_right_leg_joint
      - rear_right_thigh_joint
      - rear_right_shin_joint
      - rear_left_leg_joint
      - rear_left_thigh_joint
      - rear_left_shin_joint
    interface_name: position
    command_interfaces:
      - position
    state_interfaces:
      - position
      - velocity
```

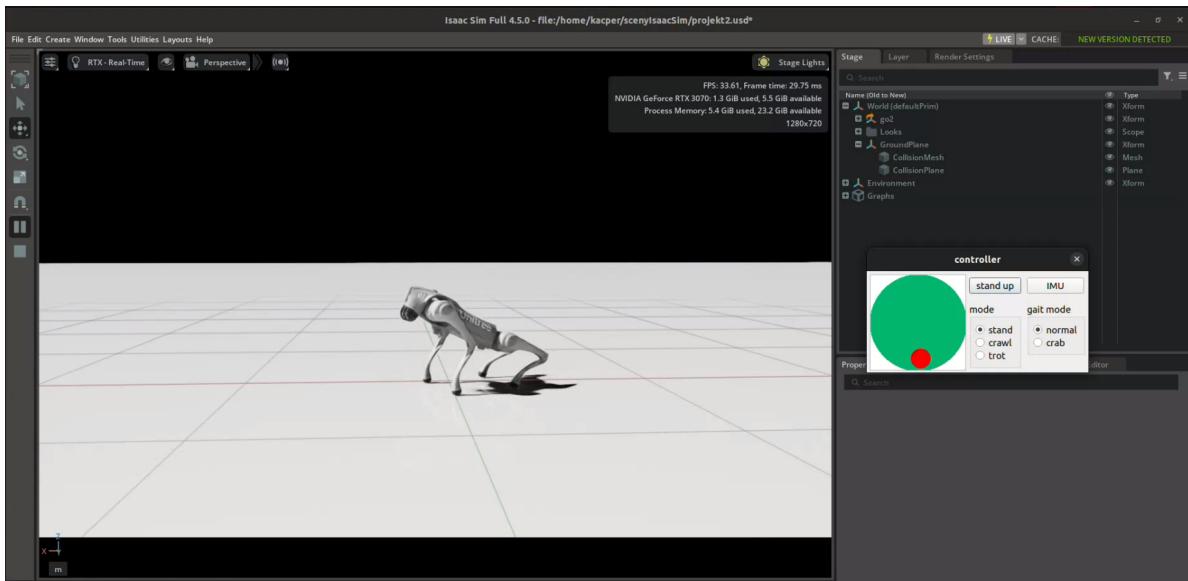
Rys. 9: Kolejność członów w kodzie.



Rys. 10: Ustawienia bloku ROS2 Subscriber.



Rys. 11: Robot Unitree Go2 pochylający się do przodu.



Rys. 12: Robot Unitree Go2 pochylający się do tyłu.

6 Jazda robotem kołowym z użyciem ROS2 i teleop_twist_keyboard

Pakiet teleop_twist_keyboard umożliwia ręczne sterowanie robotem mobilnym w systemie ROS2 z użyciem klawiatury. Wysyła komunikaty typu geometry_msgs/msg/Twist na zadany temat (domyślnie /cmd_vel). Pozwala sterować prędkością liniową oraz kątową- co jest typowym sposobem poruszania się robotów mobilnych (np. typu differential drive).

Klawisze sterujące

Podstawowe sterowanie odbywa się z użyciem następujących klawiszy:

Klawisz Działanie

i Do przodu

k Zatrzymanie(zerowanie prędkości)

, Do tyłu

j Skręt w lewo(obrót w miejscu)

l Skręt w prawo(obrót w miejscu)

u Do przodu + w lewo

o Do przodu + w prawo

m Do tyłu + w lewo

. Do tyłu + w prawo

Zmiana prędkości:

Klawisz Działanie

q Zwiększa prędkość liniową i kątową

z Zmniejsza prędkość liniową i kątową

w Zwiększa tylko prędkość liniową

x Zmniejsza tylko prędkość liniową

e Zwiększa tylko prędkość kątową

c Zmniejsza tylko prędkość kątową

Każde wciśnięcie odpowiedniego klawisza powoduje wysłanie wiadomości Twist na zadany temat, zgodnie z bieżącymi wartościami prędkości.

Instalacja pakietu

a) instalacja binarna zalecana

sudo apt update

sudo apt install ros-humble-teleop-twist-keyboard

b) Instalacja ze źródła

cd ~/ros2_ws/src

git clone https://github.com/ros2/teleop_twist_keyboard.git

cd ..

colcon build

source install/setup.bash

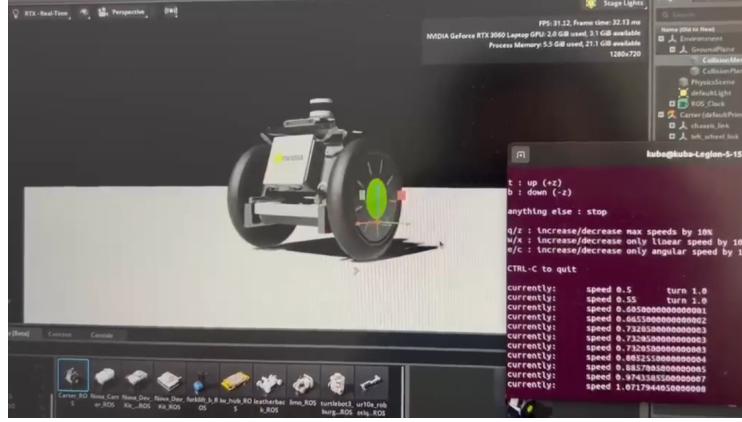
Uruchomienie

ros2 run teleop_twist_keyboard teleop_twist_keyboard

Do testów wykorzystano gotowy przykład robota mobilnego dostępnego w środowisku Isaac Sim 4.5.0, który został przygotowany do współpracy z systemem ROS2. Przykład ten zawiera wstępnie skonfigurowany zestaw węzłów (nodes) ROS2, które umożliwiają komunikację pomiędzy symulowanym robotem, a zewnętrznym środowiskiem ROS 2.

Robot ten subskrybuje wiadomości typu geometry_msgs/msg/Twist, publikowane na temacie /cmd_vel, który jest domyślnym kanałem komunikacyjnym wykorzystywanym przez pakiet teleop_twist_keyboard. Oz-

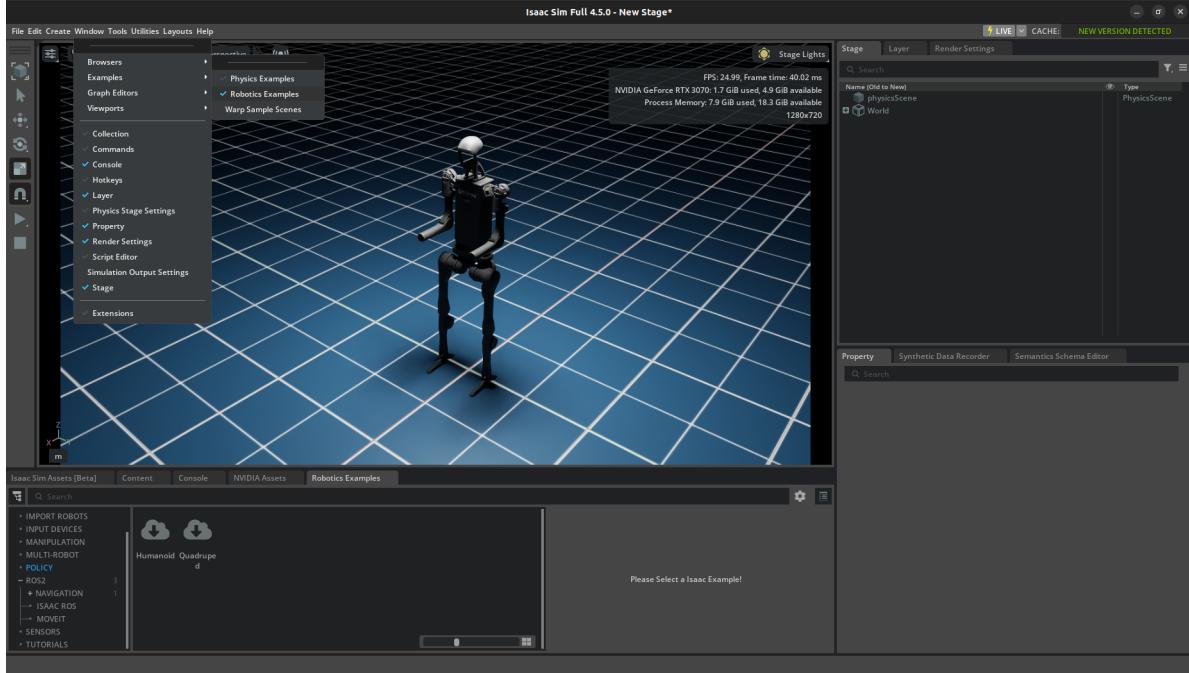
nacza to, że po uruchomieniu symulacji oraz włączeniu mostu ROS 2, komendy wysyłane z klawiatury mogą bezpośrednio sterować ruchem robota w symulacji.



Rys. 13: Robot kołowy w symulacji ISAAC SIM.

7 Przykładowe symulacje

W środowisku ISAAC SIM dostępnych jest wiele przykładowych symulacji do, których można uzyskać dostęp, wybierając w górnym pasku Window, następnie Examples, a na końcu Robotics Examples, pokazane jest to na rysunku 14. W dolnej części programu otwiera się zakładka Robotics Examples, w której znajdują się różnorodne przykłady umożliwiające sterowanie robotami przy pomocy klawiatury. W dalszej części zaprezentowano wybrane przykłady.



Rys. 14: Okno do dodawania dostępnych w ISAAC SIM przykładowych symulacji.

Na rysunku 15 przedstawiony jest widok na cyfrowego bliźniaka firmy Boston Dynamics. W przykładowej symulacji jest możliwe sterowanie nim przy pomocy strzałek i przycisków klawiatury. Sterowanie odbywa się przy pomocy następujących klawiszy:

Klawisz Działanie

Strzałka w góre/numpad 8 Idź do przodu

Strzałka w dół/numpad 2 Idź do tyłu

Lewa strzałka/numpad 4 Idź w lewo

Prawa strzałka/numpad 6 Idź w prawo

N/numpad 7 Kręć się przeciwnie do ruchu wskazówek zegara

M/numpad 9 Kręć się zgodnie z ruchem wskazówek zegara

Rysunek 16 pokazuje humanoidalnego robota firmy Unitree, którym również możliwe jest sterowanie w dostępnej przykładowej symulacji. Sterowanie odbywa się przy pomocy:

Klawisz Działanie

Strzałka w góre/numpad 8 Idź do przodu

Strzałka w lewo/numpad 4 Kręć się przeciwnie do ruchu wskazówek zegara

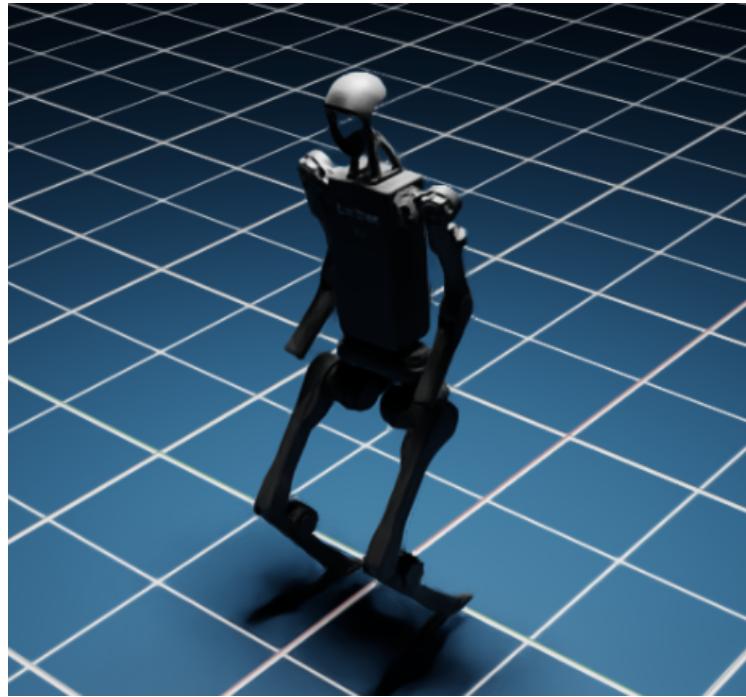
Strzałka w prawo/numpad 6 Kręć się zgodnie z ruchem wskazówek zegara



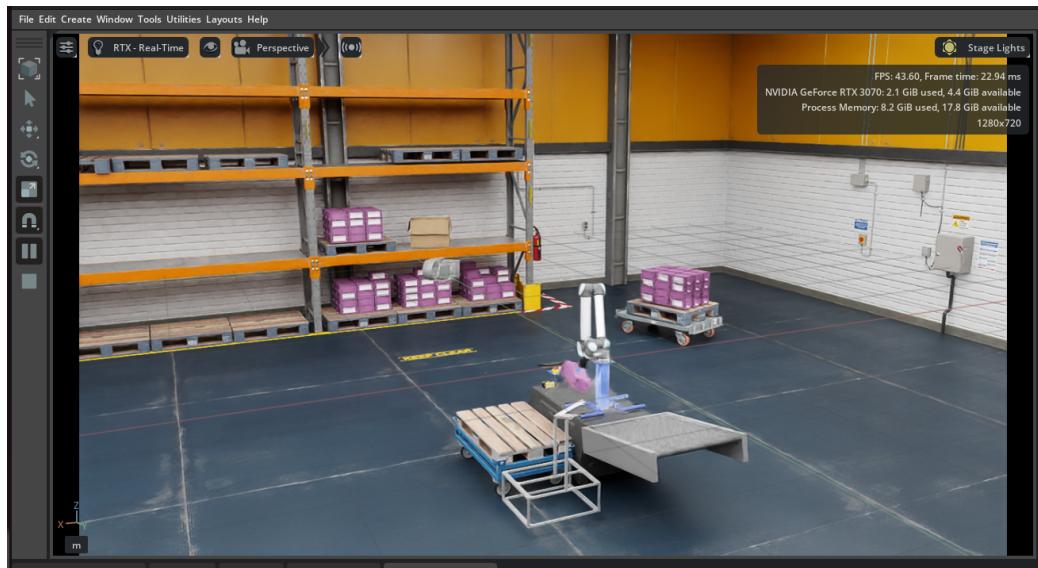
Rys. 15: Przykład z ISAAC SIM z robotem Boston Dynamics.

W środowisku ISAAC SIM dostępny jest też przykład z manipulatorem układającym kolejne dostarczane przez ruchomy taśmociąg palety. Okno z widoku tej przykładowej symulacji jest pokazane na rysunku 17.

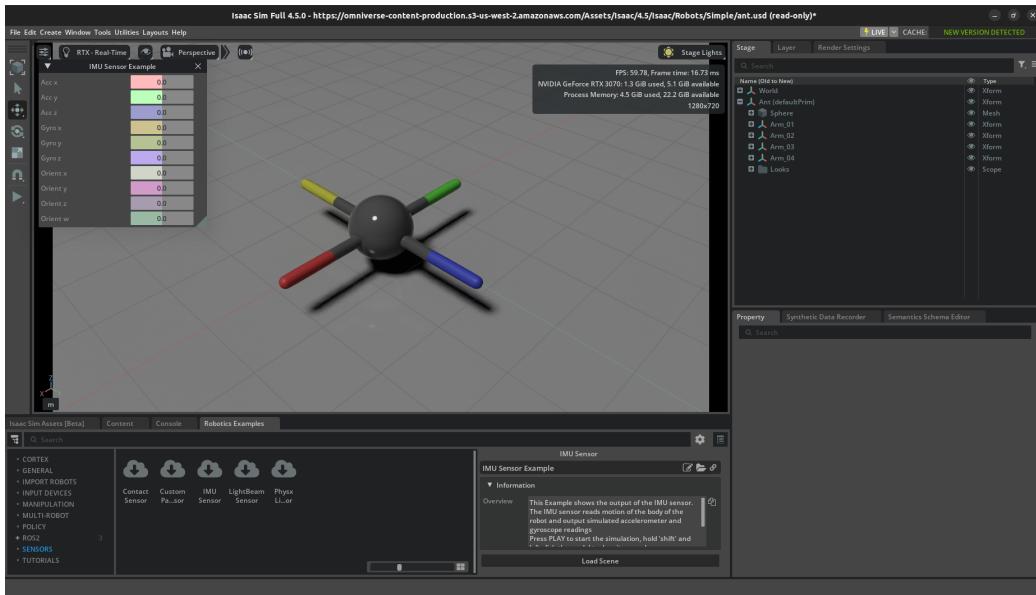
Na rysunku 18 przedstawiony jest przykład z czujnikiem IMU gdzie możliwe jest poruszanie obiektem i obserwowanie zmian wartości zwracanych przez czujnik IMU. Rysunek 19 pokazuje jak poruszanie obiektem wpływa na zmianę wartości mierzonych przez czujnik IMU. Sterowanie odbywa się poprzez jednoczesne wcisnięcie shift oraz lewego przycisku myszy na obiekcie pozwala to na poruszanie nim przez ruchy myszki.



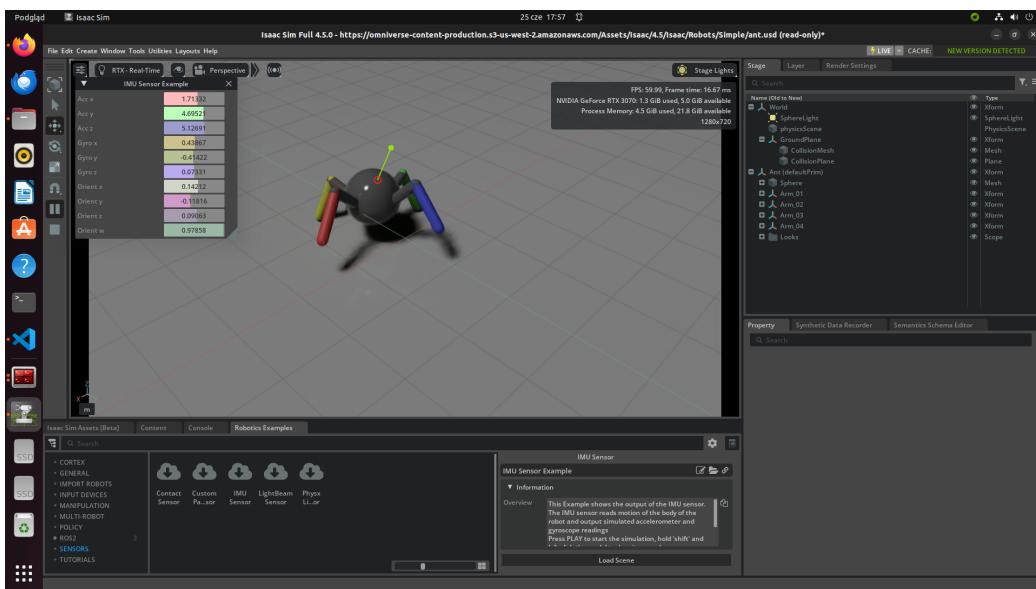
Rys. 16: Przykład z ISAAC SIM z robotem humanoidalnym.



Rys. 17: Przykład z ISAAC SIM z manipulatorem i paletami.



Rys. 18: Przykład z ISAAC SIM z czujnikiem IMU.



Rys. 19: Przykład z ISAAC SIM z czujnikiem IMU i poruszaniem obiektem.

8 Podsumowanie osiągniętych celów projektu

W trakcie wykonywania projektu udało się wykonać wszystkie narzucone odgórnie cele narzucone w nim. Czyli:

- Zainstalowano środowisko ISAAC SIM
- Skonfigurowano środowisko tak, żeby możliwa była jego komunikacja z ROS2
- W symulacji został przygotowany cyfrowy bliźniak robota Unitree Go2
- Nawiązano połączenie między środowiskiem ISAAC SIM oraz ROS2
- Cyfrowy bliźniak robota Unitree Go2 był sterowany w środowisku ISAAC SIM przy użyciu ROS2
- Przeprowadzono sterowanie robotem kołowym z użyciem ROS2 w symulacji ISAAC SIM
- Testowano przykładowe symulacje znajdujące się w ISAAC SIM między innymi sterowanie robotem humanoidalnym i robotem czworonożnym przy pomocy klawiatury oraz przykład symulacji z ramieniem robota ustawiającym palety

9 Propozycje dalszego rozwoju

Do propozycji dalszego rozwoju należą:

- Poprawa algorytmu sterującego cyforwym bliźniakiem robota Unitree Go2 lub napisanie całego algorytmu od nowa
- Sterowanie cyfrowym bliźniakiem innego robota przy pomocy ROS2
- Stworzenie własnego modelu URDF robota i sterowanie nim w środowisku ISAAC SIM
- Dodanie czujników do cyfrowego bliźniaka robota Unitree Go2