



**Politechnika
Śląska**

Projekt

Bezzałogowe Obiekty Autonomiczne/Systemy Wbudowane
i Systemy Czasu Rzeczywistego

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

Spis treści

1 Cel projektu	3
2 Wstęp teoretyczny	4
2.1 Czym jest symulator fizyki?	4
2.2 Na czym polega idea cyfrowego bliźniaka?	5
2.3 Środowisko ISAAC SIM 4.5.0	6
2.4 Jak można sterować symulowanym obiektem?	7
2.5 Środowisko ROS2	7
2.6 Czym jest robot mobilny?	7
2.7 Typy robotów mobilnych	7
2.8 Unitree Go2 EDU	8
3 Instalacja i konfiguracja środowiska ISSAC SIM oraz ROS2 Humble	11
3.1 Wykorzystany sprzęt	11
3.2 Test możliwości sprzętowych i systemowych	11
3.3 Instalacja środowiska ISAAC SIM	11
3.4 Instalacja środowiska ROS2	12
3.4.1 Krok 1: Konfiguracja lokalizacji UTF-8	12
3.4.2 Krok 2: Instalacja zależności systemowych	12
3.4.3 Krok 3: Utworzenie przestrzeni roboczej i pobranie kodu ROS 2	12
3.4.4 Krok 4: Instalacja zależności pakietów ROS 2	12
3.4.5 Krok 5: Kompilacja ROS 2	13
3.4.6 Krok 6: Konfiguracja środowiska	13
3.4.7 Weryfikacja działania ROS2	13
4 Jazda robotem kołowym z użyciem ROS2 i teleop_twist_keyboard	14
5 Cyfrowy bliźniak robota Unitree Go2	15
5.1 Podejście od strony opisywanej w dokumentacji producenta	15
5.2 Wykorzystanie modelu z ISAAC SIM	15
6 Sterowania robotem Unitree Go2 przy pomocy środowiska ROS2	17
7 Napotkane problemy	22
8 Przykładowe symulacje	22
9 Podsumowanie osiągniętych celów projektu	26
10 Propozycje dalszego rozwoju	26

1 Cel projektu

Celem niniejszego projektu było przygotowanie środowiska symulacyjnego do pracy z cyfrowym bliźniakiem czteronożnego robota mobilnego Unitree Go2.

Projekt zakładał instalację oraz odpowiednią konfigurację platformy NVIDIA Isaac Sim w wersji 4.5.0, która oferuje zaawansowane możliwości w zakresie symulacji robotyki, w szczególności dzięki integracji z technologiami RTX oraz wsparciu dla systemów ROS 2 (Robot Operating System). Oprócz instalacji samego Isaac Sim, konieczne było również zainstalowanie oraz skonfigurowanie środowiska ROS 2 Humble, które umożliwia komunikację z symulatorem i pozwala na realizację zadań związanych z planowaniem ruchu, percepcją oraz kontrolą robota.

W pierwszej fazie projektu skoncentrowano się na uruchomieniu oraz przetestowaniu środowiska Isaac Sim 4.5.0 na systemie Ubuntu 22.04, z wykorzystaniem karty graficznej wspierającej akcelerację GPU. Przeprowadzono proces konfiguracji wymaganych zależności systemowych i bibliotek oraz zweryfikowano poprawność działania przykładowych scen dostępnych w środowisku symulacyjnym.

Kolejnym krokiem było przygotowanie środowiska ROS2 w wersji Humble. Instalacja obejmowała konfigurację narzędzi systemowych, zależności oraz integrację z Pythonem i CMake. Weryfikacja poprawności instalacji została przeprowadzona poprzez uruchomienie przykładowych węzłów oraz komunikację między nimi z wykorzystaniem mechanizmu tematów i usług.

Istotnym etapem prac było połączenie środowiska ROS 2 z Isaac Sim. Dzięki dostępnej integracji i dedykowanym narzędziom takim jak ros_bridge, możliwe stało się uruchamianie symulacji w Isaac Sim z jednoczesnym przesyłaniem danych do środowiska ROS 2 i odwrotnie. Konfiguracja tego połączenia wymagała odpowiedniego skonfigurowania ścieżek, środowisk uruchomieniowych oraz aktywacji wspólnych protokołów komunikacyjnych.

Następnie przystąpiono do implementacji cyfrowego bliźniaka robota Unitree Go2. W tym celu wykorzystano dostępne zasoby modeli 3D oraz odpowiednio przygotowane pliki opisujące właściwości fizyczne i kinematykę robota. Model został zimportowany do środowiska Isaac Sim i poddany wstępnej kalibracji. W ramach projektu uruchomiono także podstawowe symulacje poruszania się robota po płaskim podłożu w kontrolowanych warunkach.

Ostatnim etapem projektu było uruchomienie testowych scenariuszy symulacyjnych, które miały na celu sprawdzenie poprawności działania cyfrowego bliźniaka w środowisku Isaac Sim oraz jego interakcji z ROS 2. Sprawdzono m.in. komunikację dwukierunkową między ROS 2 a symulatorem, odbieranie danych z sensorów wirtualnych oraz możliwość sterowania ruchem robota w czasie rzeczywistym.

Podsumowując, w ramach projektu zrealizowano pełny łańcuch przygotowania środowiska symulacyjnego – od instalacji wymaganych platform, przez integrację i konfigurację komunikacji, aż po uruchomienie cyfrowego bliźniaka robota i przeprowadzenie próbnych symulacji.

2 Wstęp teoretyczny

2.1 Czym jest symulator fizyki?

Symulator fizyki to oprogramowanie komputerowe, które umożliwia modelowanie rzeczywistych zjawisk fizycznych w wirtualnym środowisku. Dzięki niemu można badać, testować i symulować zachowanie obiektów oraz interakcje między nimi w kontrolowanych warunkach, bez konieczności przeprowadzania eksperymentów w rzeczywistości. Symulatory te są wykorzystywane w wielu dziedzinach, takich jak inżynieria, robotyka, automatyka, a także w edukacji.

Kluczowe cechy symulatorów fizyki:

- **Simulacja ruchu ciał i oddziaływań fizycznych:**

Symulatory fizyki wykorzystują algorytmy do obliczania ruchu ciał w przestrzeni, uwzględniając siły grawitacji, tarcie, opór powietrza, kolizje i inne interakcje fizyczne. Może to obejmować ruch obiektów w przestrzeni 3D, takich jak roboty, pojazdy czy mechanizmy.

- **Zachowanie w czasie rzeczywistym:**

Symulatory działają w czasie rzeczywistym, co oznacza, że wszystkie interakcje i zmiany w układzie są na bieżąco obliczane. Obiekty w symulacji reagują na działające na nie siły, co pozwala na dokładne odwzorowanie dynamiki ich ruchu.

- **Modelowanie zjawisk fizycznych:**

W symulatorach stosuje się matematyczne modele fizyczne oparte na równaniach opisujących zjawiska takie jak przewodzenie ciepła, mechanika ciał stałych, przepływ płynów czy elektromagnetyzm. Te modele pozwalają na odwzorowanie rzeczywistych warunków w wirtualnym środowisku.

- **Interakcja z użytkownikiem:**

Symulator umożliwia interakcję użytkownika z obiektami w wirtualnym świecie. Użytkownicy mogą zmieniać parametry obiektów, takie jak masa, kształt, siły działające na nie, a także manipulować nimi, np. sterować robotami, maszynami lub pojazdami w symulacji.

- **Wirtualne środowisko:**

Symulatory fizyki oferują realistyczne środowiska 3D, które odwzorowują rzeczywiste warunki, takie jak tereny, pomieszczenia czy przestrzenie robocze. Dzięki zaawansowanej grafice użytkownicy mogą wizualizować interakcje między obiektami oraz obserwować, jak zmieniają się warunki w trakcie symulacji.

- **Sztuczna inteligencja (AI):**

W symulatorach często wykorzystuje się algorytmy sztucznej inteligencji do testowania zachowań autonomicznych systemów, takich jak roboty. AI jest wykorzystywana do uczenia maszynowego, planowania trajektorii, unikania przeszkód czy podejmowania decyzji w dynamicznych warunkach.

- **Testowanie i optymalizacja:**

Symulatory fizyki pozwalają na testowanie różnych scenariuszy i optymalizację systemów bez ryzyka uszkodzenia rzeczywistych obiektów. Dzięki temu można przeprowadzać testy bezpieczeństwa, efektywności energetycznej, wydajności czy niezawodności w różnych warunkach, zanim wprowadzi się je do rzeczywistej produkcji lub użytku.

- **Zastosowania w różnych branżach:**

Symulatory fizyki są wykorzystywane w wielu dziedzinach, w tym w inżynierii, robotyce, produkcji, edukacji, medycynie oraz w naukach przyrodniczych. Dzięki nim można tworzyć prototypy, testować

urządzenia, przeprowadzać eksperymenty fizyczne, a także badać efektywność różnych technologii w wirtualnym świecie.

2.2 Na czym polega idea cyfrowego bliźniaka?

Cyfrowy bliźniak jest cyfrowym modelem, który dokładnie odzwierciedla wszystkie istotne cechy fizycznego obiektu. Może to być maszyna, urządzenie, cały zakład produkcyjny, pojazd, a nawet miasto. W przypadku maszyn, reprezentacja może obejmować takie dane jak kształt, wymiary, właściwości materiałów, a także wszelkie informacje o zużyciu, stanie technicznym i awariach.

Integracja z rzeczywistością: Cyfrowy bliźniak działa w pełnej synchronizacji z fizycznym obiektem. Dzięki zbieraniu danych z różnych źródeł, takich jak czujniki, urządzenia IoT (Internet of Things), systemy sterowania, kamery czy urządzenia monitorujące, wirtualny model jest na bieżąco aktualizowany, odzwierciedlając zmiany w fizycznym obiekcie w czasie rzeczywistym. Na tej podstawie można przeprowadzać analizy i prognozy dotyczące stanu urządzeń.

Symulacje i analizy: Cyfrowy bliźniak umożliwia przeprowadzanie symulacji i analiz w wirtualnym środowisku bez potrzeby ingerencji w fizyczny obiekt. Dzięki tym symulacjom można testować różne scenariusze, takie jak zmiany warunków operacyjnych, awarie, zmiany parametrów środowiskowych, wpływ nowych procesów technologicznych, itp. Pozwala to na optymalizację procesów, planowanie konserwacji, identyfikowanie problemów zanim staną się one krytyczne, a także na przewidywanie przyszłych zachowań.

Monitorowanie w czasie rzeczywistym: Zbieranie danych z czujników zamontowanych w obiekcie (np. czujniki temperatury, ciśnienia, vibracji) pozwala na monitorowanie jego stanu na bieżąco. Dzięki temu możliwe jest szybkie wykrycie usterek, błędów operacyjnych lub innych anomalii. Systemy monitorowania mogą również wysyłać powiadomienia lub generować raporty dotyczące zdrowia obiektu.

Optymalizacja i podejmowanie decyzji: Cyfrowe bliźniaki wspierają podejmowanie decyzji na podstawie analizy dużych zbiorów danych. Dzięki symulacjom oraz analizie danych w czasie rzeczywistym, można optymalizować procesy produkcyjne, przewidywać konieczność przeprowadzenia konserwacji lub modernizacji, a także poprawić wydajność obiektów i systemów. Pozwala to na minimalizację kosztów operacyjnych, wydłużenie cyklu życia urządzeń i zwiększenie efektywności.

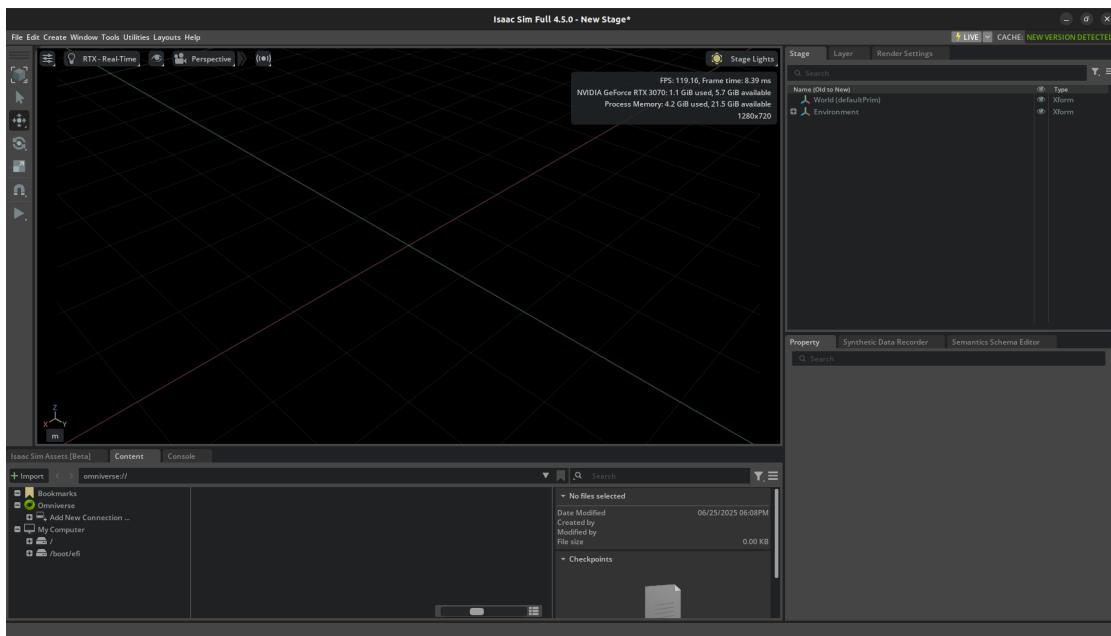
Cyfrowy bliźniak to innowacyjne podejście do monitorowania i optymalizacji obiektów, systemów i procesów poprzez tworzenie ich wirtualnych replik. Dzięki zbieraniu danych w czasie rzeczywistym, symulacjom i analizom, cyfrowy bliźniak pozwala na optymalizację operacji, przewidywanie awarii, a także lepsze zarządzanie zasobami, co prowadzi do zwiększenia efektywności, bezpieczeństwa i oszczędności.

2.3 Środowisko ISAAC SIM 4.5.0

NVIDIA Isaac Sim 4.5.0 to przykład zaawansowanego środowiska symulacyjnego opartego na silniku Omniverse, stworzonego z myślą o rozwoju i testowaniu algorytmów dla robotyki. Umożliwia on 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 percepcji, lokalizacji, mapowania, planowania ruchu czy sterowania w warunkach zbliżonych do rzeczywistych.

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



Rys. 1: Środowisko ISAAC SIM 4.5.0

2.4 Jak można sterować symulowanym obiektem?

2.5 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.

2.6 Czym jest robot mobilny?

Robot mobilny to autonomiczne lub zdalnie sterowane urządzenie, które posiada zdolność przemieszczania się w przestrzeni. W odróżnieniu od robotów stacjonarnych (np. ramion robotycznych), roboty mobilne mogą zmieniać swoje położenie względem otoczenia, co pozwala im na realizację zadań takich jak eksploracja, transport, inspekcja czy wsparcie operacyjne w środowiskach dynamicznych. Ruch robota może być realizowany na różne sposoby – za pomocą kół, gąsienic, nóg lub kombinacji tych elementów. Roboty mobilne są szeroko wykorzystywane zarówno w przemyśle, jak i w zastosowaniach cywilnych oraz wojskowych, a rozwój technologii czujników, algorytmów sterowania i sztucznej inteligencji umożliwia im coraz bardziej samodzielne funkcjonowanie w złożonych środowiskach.

2.7 Typy robotów mobilnych

Roboty mobilne można podzielić na różne typy w zależności od sposobu ich poruszania się. Najpopularniejsze z nich to roboty kołowe oraz roboty kroczące.

Roboty kołowe wykorzystują koła jako podstawowy mechanizm napędu i kierowania. Tego typu konstrukcje są technicznie prostsze, bardziej energooszczędne i łatwiejsze w sterowaniu, zwłaszcza na płaskich, utwardzonych powierzchniach. Dzięki prostemu modelowi kinematycznemu, sterowanie ruchem takiego robota (np. skręcanie, jazda po łuku) można łatwo realizować za pomocą stosunkowo nieskomplikowanych algorytmów. Z tego względu roboty kołowe są często stosowane w robotyce edukacyjnej, magazynowej czy inspekcyjnej.

Roboty kroczące typu quadruped, czyli roboty czworonożne, są znacznie bardziej zaawansowane pod względem konstrukcyjnym i sterowania. Poruszają się w sposób zbliżony do zwierząt, co pozwala im na pokonywanie trudnych, nieregularnych terenów, gdzie roboty kołowe nie mogłyby się poruszać. Jednak taka forma lokomocji wymaga złożonych algorytmów równoważenia, planowania kroków i stabilizacji postawy, co czyni sterowanie nimi znacznie bardziej trudnym. Pomimo tej złożoności, roboty quadruped zyskują na popularności w zastosowaniach terenowych, ratowniczych i wojskowych ze względu na swoją mobilność i adaptacyjność.

2.8 Unitree Go2 EDU

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 každa (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 integracją 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~ 33.6V
- **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 rozdzielczości 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: Robot Unitree Go2.

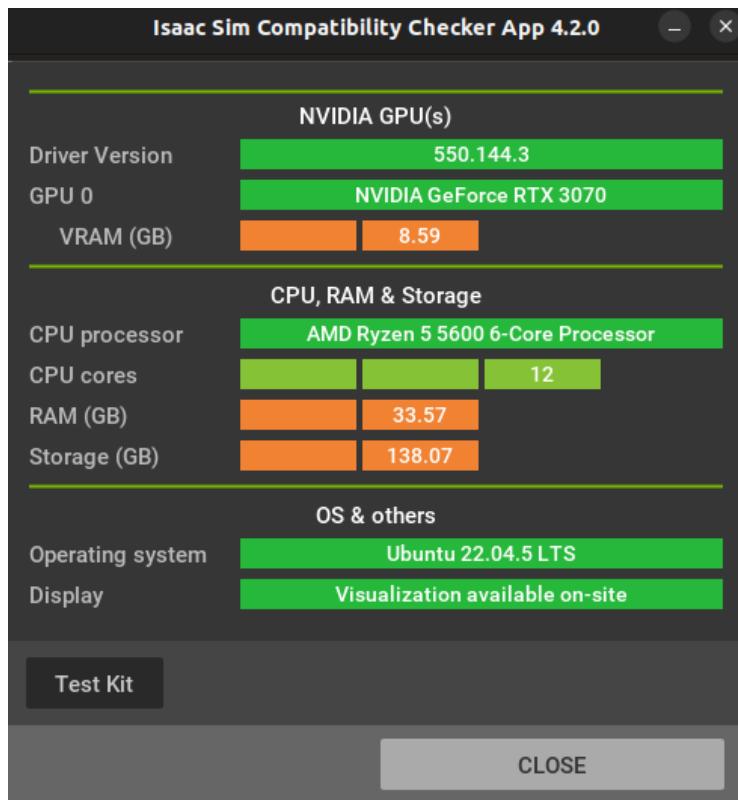
3 Instalacja i konfiguracja środowiska ISSAC SIM oraz ROS2 Humble

3.1 Wykorzystany sprzęt

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.

3.2 Test możliwości sprzętowych i systemowych

W oprogramowaniu NVIDIA dostępna jest specjalna aplikacja, pozwalająca na sprawdzenie, czy komputer z którego korzystamy, spełnia wymagania dla oprogramowania ISAAC SIM. Na rysunku 3 pokazany jest wynik sprawdzenia jednego z wykorzystanych w projekcie komputerów.



Rys. 3: Wynik sprawdzenia kompatybilności sprzętu z ISAAC SIM.

3.3 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.

3.4 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
```

3.4.1 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
```

3.4.2 Krok 2: Instalacja zależności systemowych

Zainstalowaliśmy wymagane pakiety narzędziowe i kompilacyjne:

```
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
```

3.4.3 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
```

3.4.4 Krok 4: Instalacja zależności pakietów ROS 2

Zainstalowaliśmy wszystkie zależności wymagane do kompilacji (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"
```

3.4.5 Krok 5: Kompilacja ROS 2

Rozpoczęliśmy komplikację całego środowiska z wykorzystaniem colcon:

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

Ten proces trwał kilkanaście minut.

3.4.6 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.

3.4.7 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 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 wcisnięcie 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

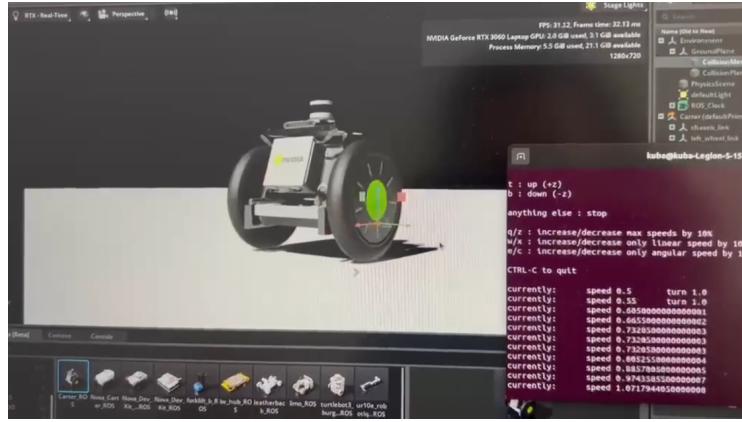
source install/setup.bash

Uruchomienie

ros2 run teleop_twist_keyboard teleop_twist_keyboard

Do testów wykorzystano gotowy przykład robota mobilnego dostępny 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. Oznacza to, że po uruchomieniu symulacji oraz włączeniu mostu ROS 2, komendy wysyłane z klawiatury mogą bezpośrednio sterować ruchem robota w symulacji.



Rys. 4: Robot kołowy w symulacji ISAAC SIM.

5 Cyfrowy bliźniak robota Unitree Go2

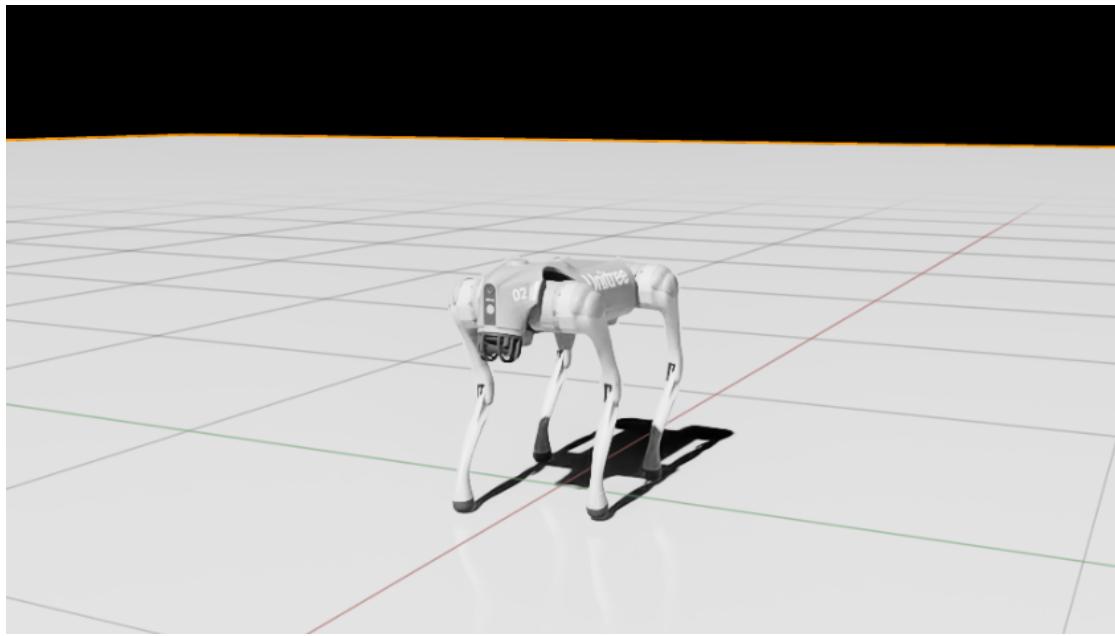
5.1 Podejście od strony opisywanej w dokumentacji producenta

5.2 Wykorzystanie modelu z ISAAC SIM

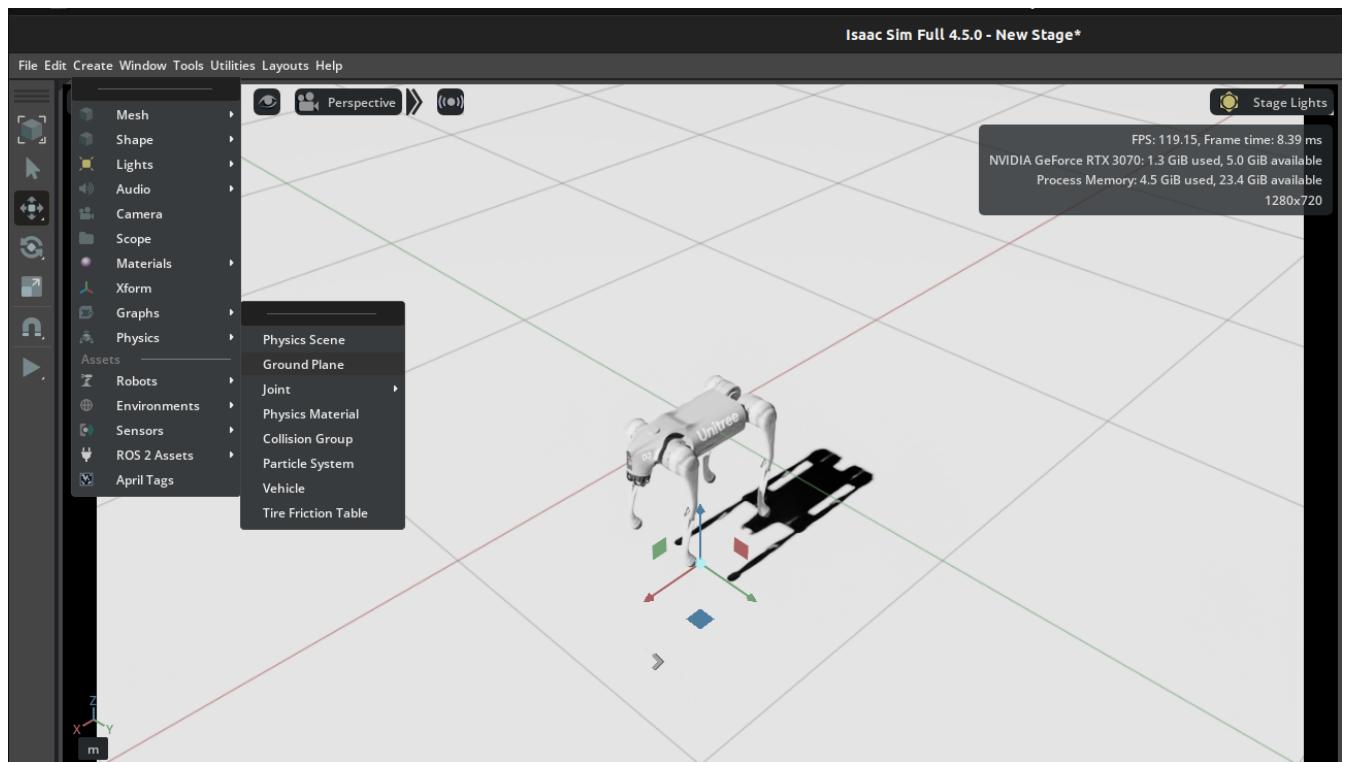
W środowisku ISSAC SIM znajduje się gotowy model cyfrowego bliźniaka robota Unitree Go2, w celu umieszczenia go na scenie otwartego programu należy odnaleźć go w oknie Isaac Sim Assets i dodać go, wciskając przycisk Load as Reference lub przeciągając jego ikonę na scenę. Widok okna Isaac Sim Assets przedstawiony jest na rysunku 5. Widok umieszczonego w symulacji robota przedstawiony jest na rysunku 6. Na rysunku 7 pokazane jest, jak dodać podłogę w scenie, należy wybrać w górnym pasku Create, następnie Physics, a na końcu wcisnąć Ground Plane.



Rys. 5: Biblioteka dostępnych robotów Isaac Sim Assets.



Rys. 6: Cyfrowy bliźniak robota Unitree Go2 w środowisku ISAAC SIM.



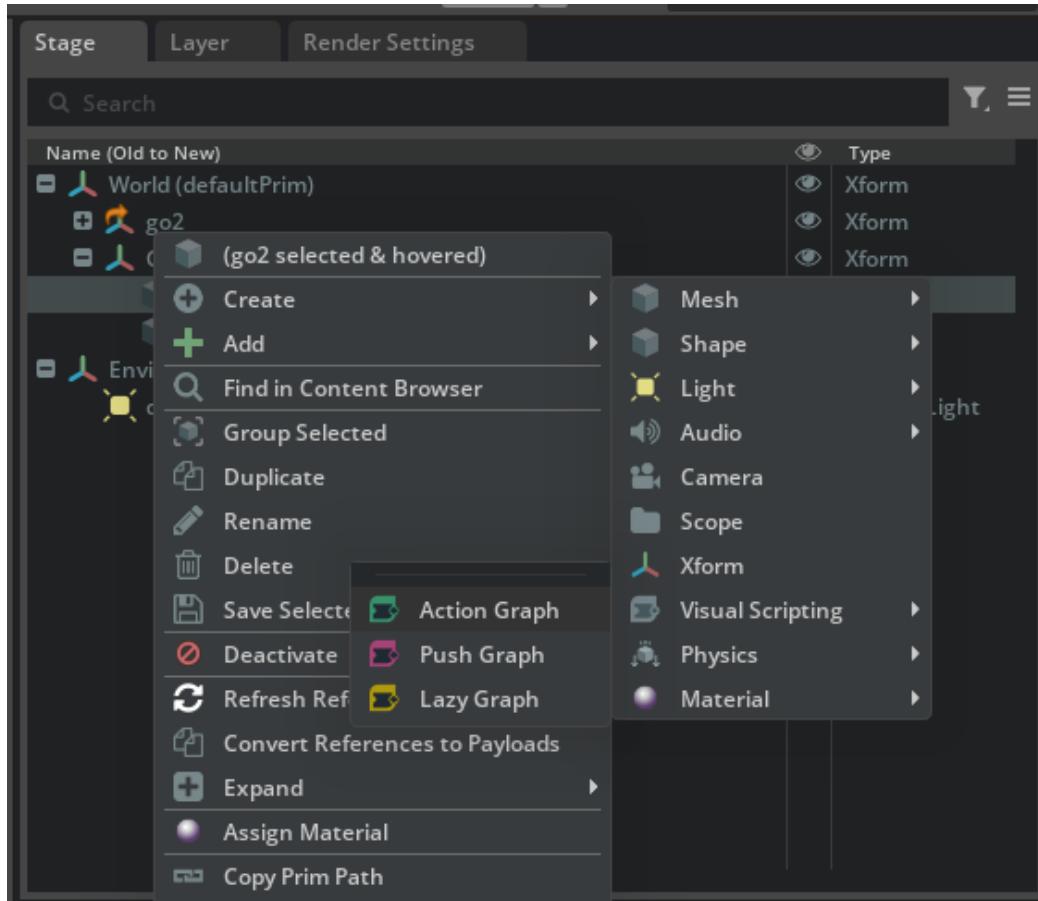
Rys. 7: Dodanie podłoża w ISSAC SIM.

6 Sterowania robotem Unitree Go2 przy pomocy środowiska ROS2

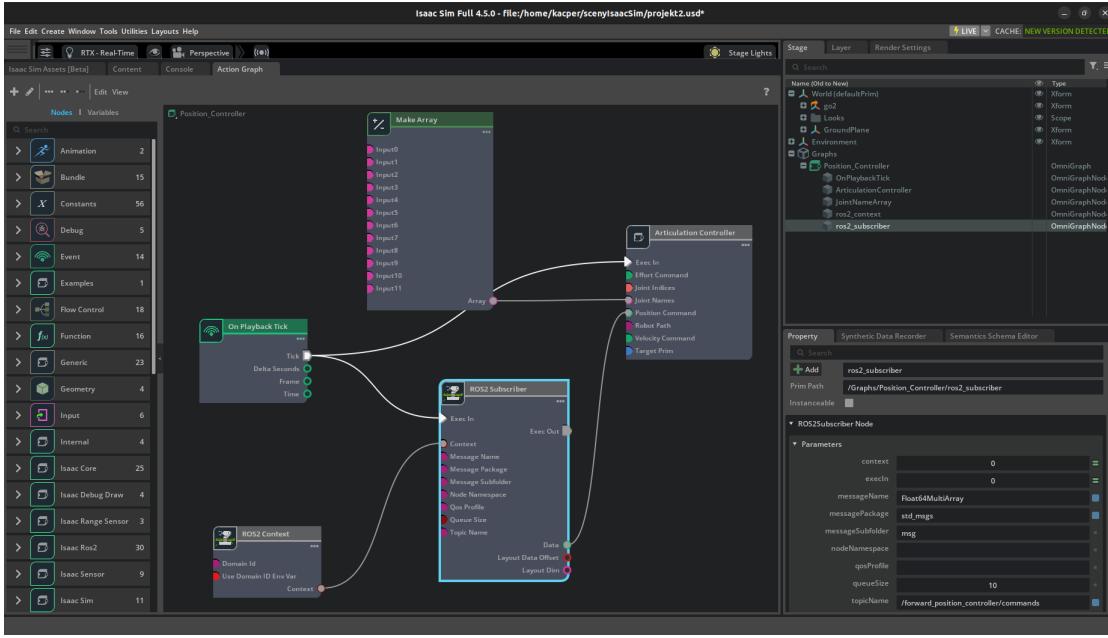
Wszystkie potrzebne pliki źródłowe do sterowania sterowania robotem znajdują się w linku poniżej.

<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 8 pokazane jest jak dodać Action Graph, na rysunku 9 widoczne są potrzebne do sterowania robotem bloki oraz połączenia między nimi.



Rys. 8: Dodanie Action Graph do cyfrowego bliźniaka Unitree Go2



Rys. 9: 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 10 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 11

Blok ROS2Subscriber służy do otrzymywania danych z programu środowiska ROS2, tak zwanego subskrybowania danych, na rysunku 12 pokazano jak poprawnie skonfigurować blok ROS2Subscriber żeby dane były odpowiednio pobierane. Otrzymane dane są przesyłane do bloku Articulation Controller.

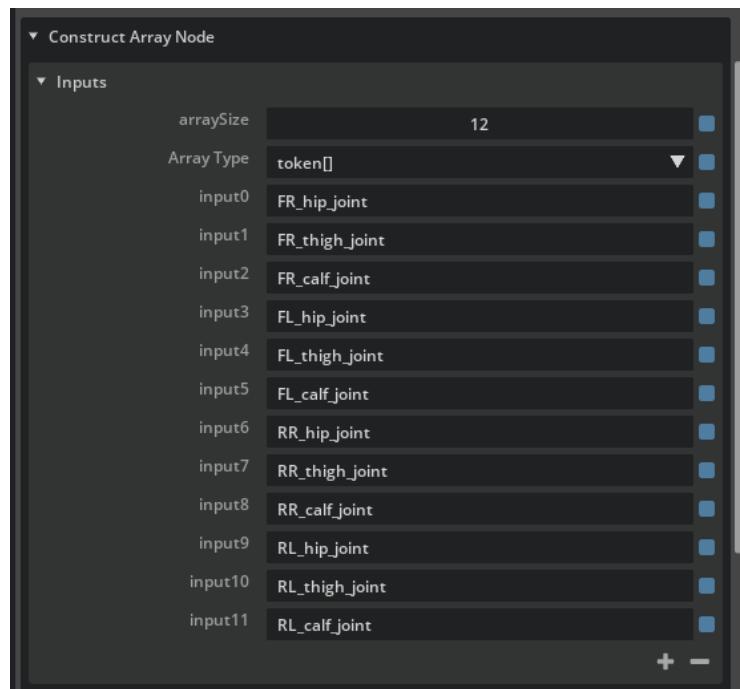
W celu uruchomienia programu sterującego, po wcześniejszym wypakowaniu potrzebnych plików, należy przejść do folderu `/quadruped_robot_ROS2` i w terminalu wpisać (żeby uruchomić terminal w danym folderze, należy, będąc w aplikacji pliki w danym folderze, wcisnąć prawy przycisk myszy i ‘otwórz w terminalu’):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



Rys. 10: Ustawienia bloku Make Array.

```

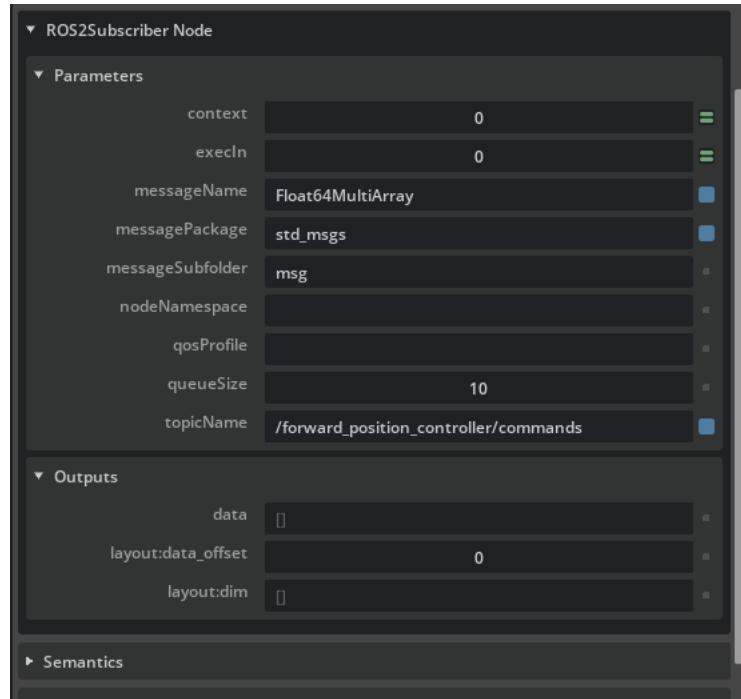
controller_manager:
  ros__parameters:
    update_rate: 1000  # Hz

  forward_position_controller:
    type: forward_command_controller/ForwardCommandController

  joint_state_broadcaster:
    type: joint_state_broadcaster/JointStateBroadcaster

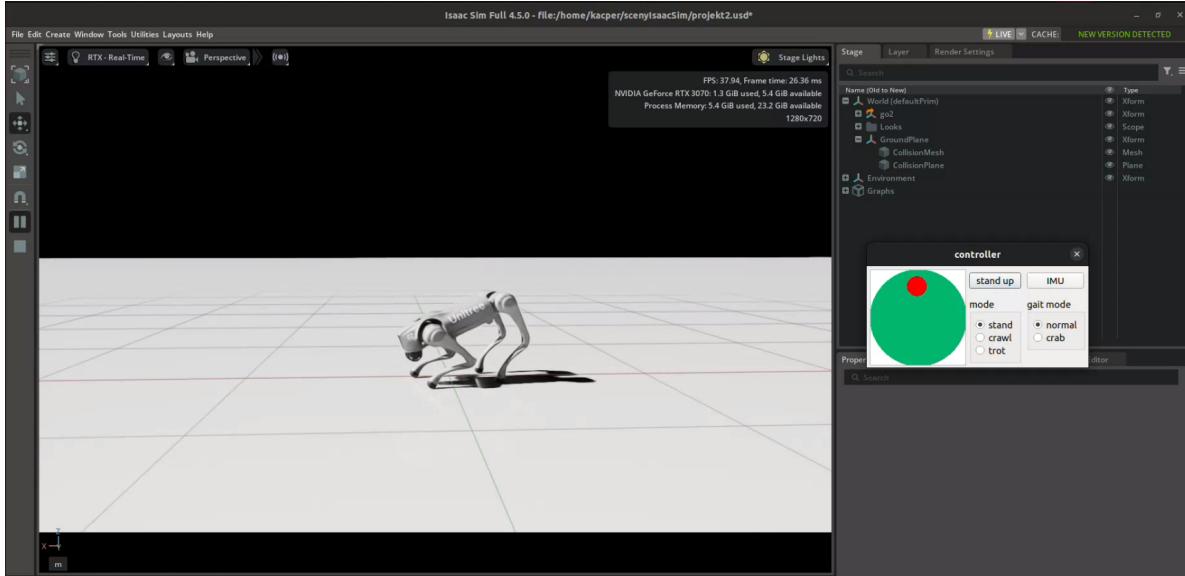
forward_position_controller:
  ros__parameters:
    joints:
      - front_right_leg_joint
      - front_right_thigh_joint
      - front_right_shin_joint
      - front_left_leg_joint
      - front_left_thigh_joint
      - front_left_shin_joint
      - rear_right_leg_joint
      - rear_right_thigh_joint
      - 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. 11: Kolejność członów w kodzie.

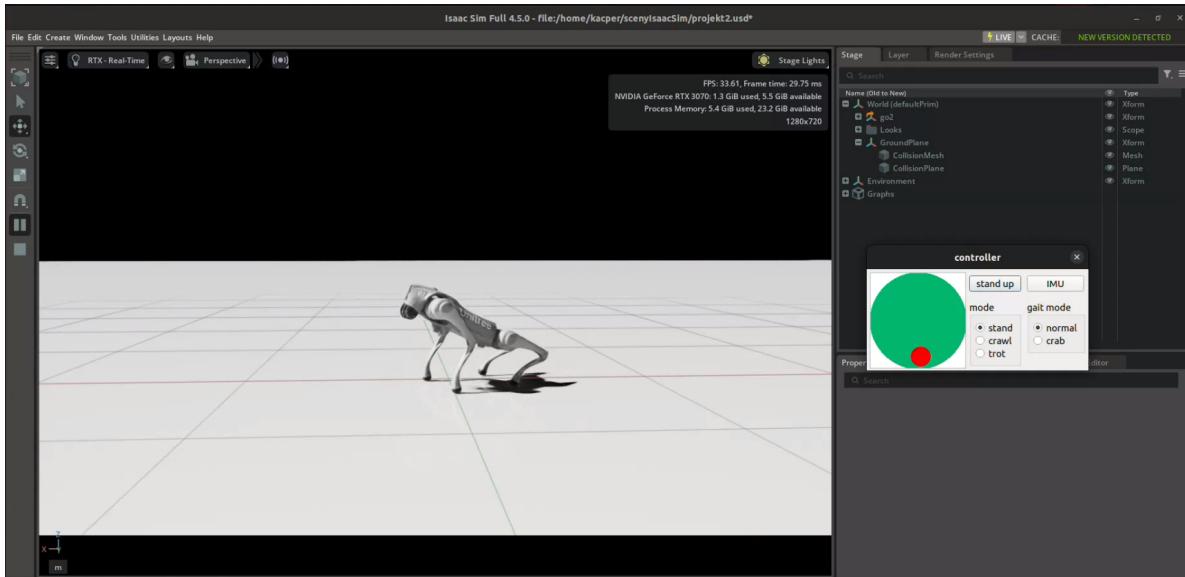


Rys. 12: Ustawienia bloku ROS2 Subscriber.

Na rysunku 13 oraz na rysunku 14 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.



Rys. 13: Robot Unitree Go2 pochylający się do przodu.



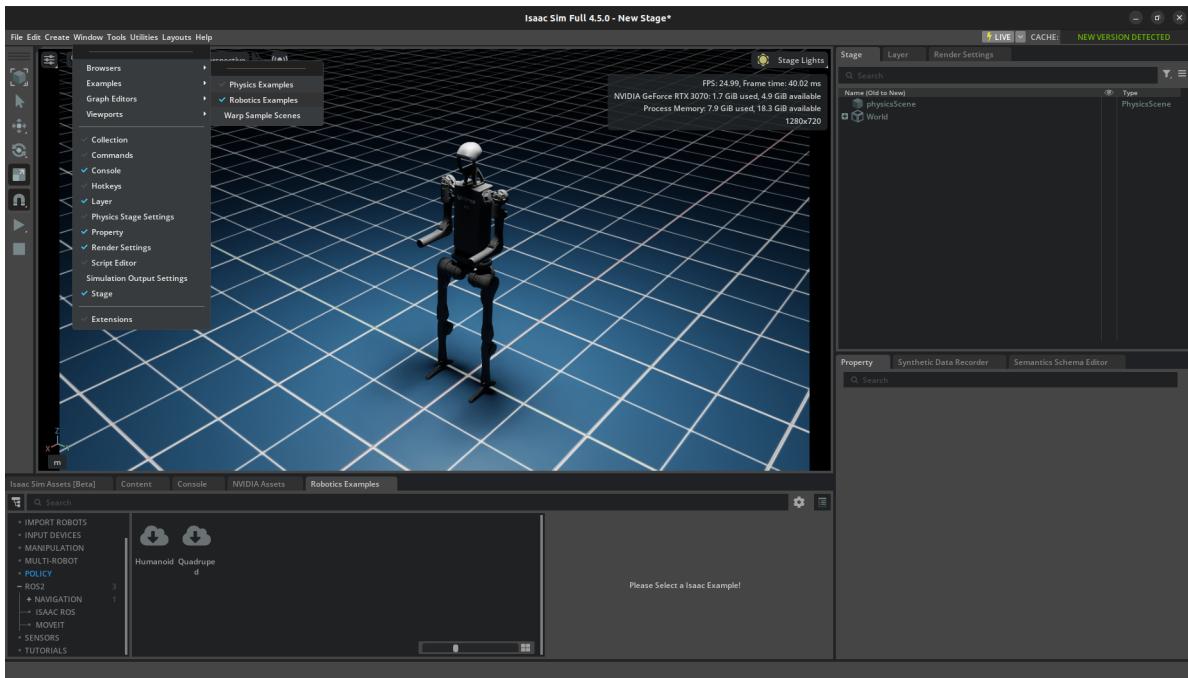
Rys. 14: Robot Unitree Go2 pochylający się do tyłu.

Do wykonania tej części projektu korzystaliśmy z poniższego filmu.
Link do filmu

7 Napotkane problemy

8 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 15. 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. 15: Okno do dodawania dostępnych w ISAAC SIM przykładowych symulacji.

Na rysunku 16 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

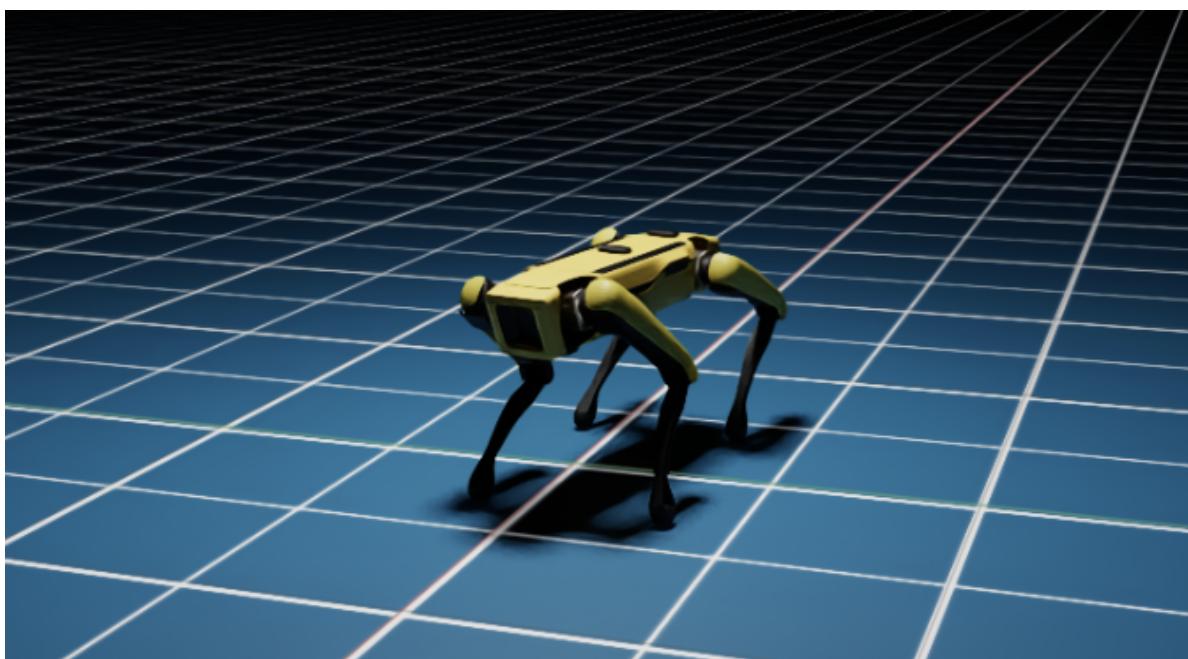
Rysunek 17 pokazuje humanoidalnego robota firmy Unitree, którym również możliwe jest sterowanie w dostępnej przykładowej symulacji. Sterowanie odbywa się przy pomocy:

Klawisz Działanie

Strzałka w góre/numpad 8 Idź do przodu

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

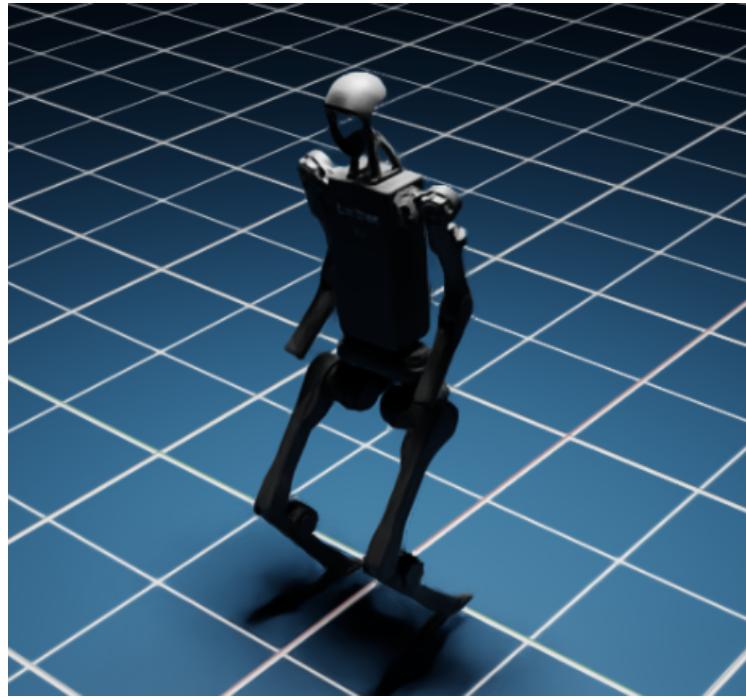
Strzałka w prawo/numpad 6 Kręć się zgodnie z ruchem wskazówek zegara



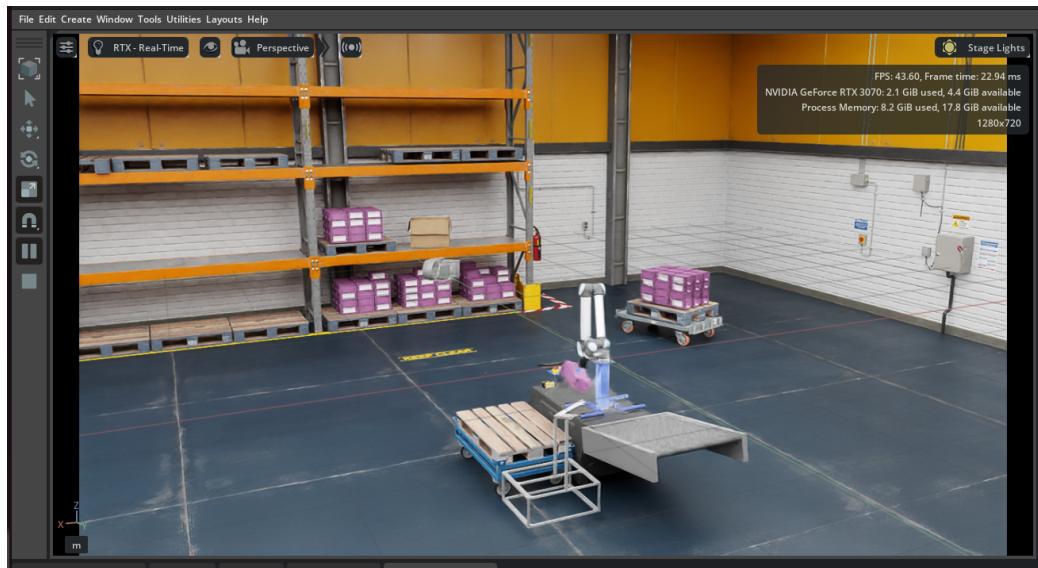
Rys. 16: Przykład z ISAAC SIM z robotem Boston Dynamics.

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 18.

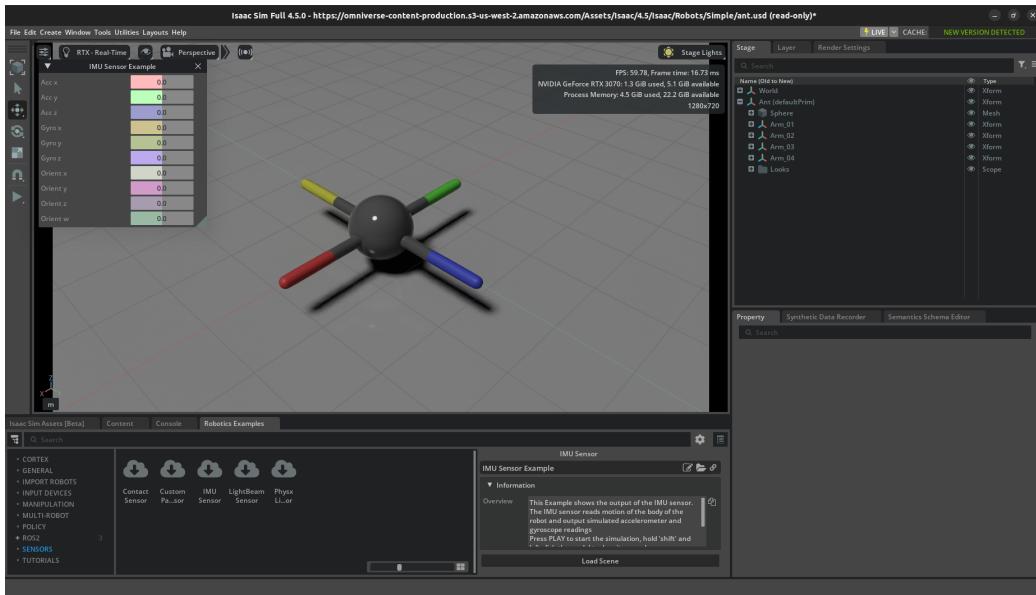
Na rysunku 19 przedstawiony jest przykład z czujnikiem IMU, gdzie możliwe jest poruszanie obiektem i obserwowanie zmian wartości zwracanych przez czujnik IMU. Rysunek 20 pokazuje, jak poruszanie obiektem wpływa na zmianę wartości mierzonych przez czujnik IMU. Sterowanie odbywa się poprzez jednoczesne wcisnięcie shift oraz lewego przycisku myszy na obiekcie pozwala to na poruszanie nim przez ruchy myszki.



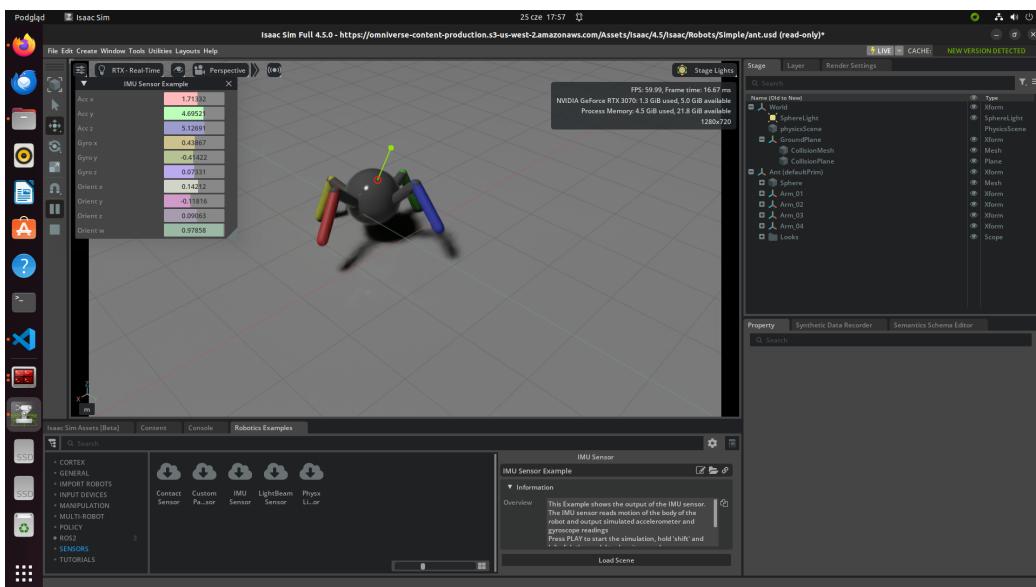
Rys. 17: Przykład z ISAAC SIM z robotem humanoidalnym.



Rys. 18: Przykład z ISAAC SIM z manipulatorem i paletami.



Rys. 19: Przykład z ISAAC SIM z czujnikiem IMU.



Rys. 20: Przykład z ISAAC SIM z czujnikiem IMU i poruszaniem obiektem.

9 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

10 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