www.pl2022.pollub.pl
e-mail: pl2022@pollub.pl

Zintegrowany Program Rozwoju Politechniki Lubelskiej

Biuro Projektu:
ul.Nadbystrzycka 38H
20 -618 Lublin

AUTONOMICZNE ROBOTY TRANSPORTOWE

Workbook

Autor: Paweł Stączek

Lublin, 2020 rok

PROGRAM WIEDZA EDUKACJA ROZWÓJ

Fundusze Europejskie Wiedza Edukacja Rozwój

Rzeczpospolita Polska

Unia Europejska Europejski Fundusz Społeczny

WPROWADZENIE

Celem przedmiotu autonomiczne roboty transportowe jest przekazanie podstawowej wiedzy z zakresu:

- roli pojazdów autonomicznych w przemysłowych systemach transportu bliskiego (w systemach intralogistycznych),
- technik, metod oraz algorytmów lokalizacji i nawigacji autonomicznych pojazdów transportowych w ich środowisku pracy,
- narzędzi informatycznych wspomagających projektowanie i rozwój pokładowych systemów automatycznego sterowania pojazdami autonomicznymi.

Niniejszy workbook zawiera instrukcje do realizacji zajęć praktycznych (ćwiczeń laboratoryjnych) z zastosowaniem m.in. oprogramowania do komputerowej symulacji i wizualizacji 3D robotów wraz z ich otoczeniem oraz oprogramowania do implementacji algorytmów sterowania dla autonomicznych robotów kołowych.

AUTONOMICZNE ROBOTY TRANSPORTOWE

Ćwiczenie ART-1

Instalowanie i konfiguracja wirtualnego komputera z systemami Linux + ROS

I. Cel ćwiczenia

- Celem dydaktycznym jest zapoznanie z podstawowymi elementami środowiska do tworzenia i testowania układów sterowań dla robotów Robot Operating System – ROS.
- Celem praktycznym jest zainstalowanie i konfiguracja wirtualnego komputera (wirtualnej maszyny) z systemem operacyjnym Linux i środowiskiem ROS oraz uruchomienie i przetestowanie symulatora 3D z kołowym robotem mobilnym.

II. Przebieg ćwiczenia

1. Pobierz i zainstaluj program "VMware Workstation Player" dla Windows emulujący wirtualny komputer (ang. virtual machine). Na komputerze wirtualnym, zwanym gościem (ang. Guest), uruchomisz później system operacyjny Linux oraz środowisko Robot Operating System.
   UWAGA! Instaluj "VMware Player" w wersji 15.5 (lub nowszą) z poniższego linku. Pozostaw domyślne ustawienia podczas instalacji.

   https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html
2. Pobierz i rozpakuj plik (1,7GB) z obrazem wirtualnego komputera z zainstalowanym systemem operacyjnym Linux (dystrybucja Kubuntu 14.04) oraz środowiskiem ROS (dystrybucja Indigo) według instrukcji pod adresem:

   https://www.mathworks.com/support/product/robotics/v3-installation-instructions.html

   Gdy komputer gospodarza (Host) pracuje pod Windows, to wykonaj instrukcje z sekcji:

   Platform-Specific Installation Instructions - Windows (64-bit)

   Wskazówka: pierwszy punkt instrukcji (Download and install the VMware Player software) już zrealizowałeś.

   UWAGA! Pobrany plik rozpakuj do „trwałego” katalogu na dysku komputera gospodarza. Na rozpakowanych plikach będzie pracował wirtualny komputer (tzn. nigdy nie modyfikuj ani nie przenoś tych rozpakowanych plików).
3. Uruchom wirtualny komputer z okna aplikacji programu VMware Workstation Player.

   UWAGA! Na pytanie o instalację dodatkowego pakietu z narzędziami odpowiedz „Remind Later” (tj. nie instaluj pakietu).

W przypadku problemów z uruchomieniem wirtualnego komputera należy zajrzeć do sekcji *Troubleshooting* na stronie WWW z poprzedniego punktu.

W przypadku starszego komputera gospodarza (z procesorem 2 rdzeniowym) może być wymagane zmniejszenie liczby rdzeni (tak naprawdę rdzeni logicznych) CPU do 1 lub/i rozmiaru RAM przydzielanych komputerowi wirtualnemu (chociaż praca z ustawieniami poniżej wartości domyślnych będzie uciążliwa).

Wskazówka: W każdej chwili możesz „zamrozić pracę" i zachować stan wirtualnego komputera przyciskiem PAUZA na pasku narzędzi u góry okna VMware Workstation. Stan zapisywany jest na dysku komputera gospodarza, zatem możesz nawet zamknąć system operacyjny gospodarza (Windows) i wyłączyć zasilanie komputera. W dowolnej chwili możesz „wznowić" pracę wirtualnego komputera.

4. Używaj wirtualnego komputera w Trybie pełnoekranowym (ikona na pasku narzędzi u góry okna - *Enter full screen mode*). Wtedy wirtualny komputer działa najstabilniej.
5. Kliknij (raz) ikonę *Gazebo Playground* na pulpicie. Uruchomi się symulator 3D Gazebo z robotem mobilnym Turtlebot. W przypadku niepowodzenia (czerwone komunikaty błędów w oknie terminala) naciśnij w oknie terminala klawisze Ctrl+C (ang. break) i poczekaj na zakończenie wszystkich uruchomionych w nim procesów. Ponów próbę uruchomienia (pracuj w trybie pełnoekranowym).
6. Wykonaj komendę, która m.in. zaktualizuje adresy serwerów z dostępnymi aktualizacjami dla systemu Linux oraz zainstalowanych aplikacji, pisząc w oknie terminala:

   sudo apt update

   (twoja nazwa użytkownika: *user*, hasło: *password*)
   Wskazówka: Treść komendy możesz skopiować z tej instrukcji do okna terminala tekstowego. W Linux skrót klawiszowy do wklejania tekstu ze schowka to (Ctrl+Shift+V).

7. Przeczytaj dokładnie komunikaty wyświetlone w oknie terminala po ostatniej komendzie. W przypadku błędów uwierzytelniania źródeł zasobów (przykład komunikatu poniżej) rozwiąż problem kierując się instrukcją na stronie:

   https://chrisjean.com/fix-apt-get-update-the-following-signatures-couldnt-be-verified-because-the-public-key-is-not-available/

   po czym ponów komendę z punktu 6.

   W: An error occurred during the signature verification. The repository is not updated and the previous index files will be used. GPG error: http://packages.ros.org trusty InRelease: The following signatures couldn't be verified because the public key is not available: NO_PUBKEY ……..

   W: Failed to fetch http://packages.ros.org/ros/ubuntu/dists/trusty/InRelease

   W: Some index files failed to download. They have been ignored, or old ones used instead.

8. Zainstaluj program Terminator usprawniający pracę z wieloma oknami terminali:

sudo apt install terminator

9. Dodaj ikonę programu Terminator do grupy Favourites (w celu szybkiego dostępu do tej aplikacji) - kliknij w lewym dolnym rogu pulpitu:

   KickOff Application Lancher (odpowiednik Menu Start) -> W polu Search wpisz: terminator -> Kliknij prawym przyciskiem myszy w ikonę odnalezionej aplikacji Terminator -> Z menu kontekstowego wybierz Add to Favourites

   Ikona Terminatora będzie widoczna na liście aplikacji po kliknięciu KickOff Application Lancher.

10. Analogicznie dodaj do grupy Favourites aplikację System monitor (lub ksysguard).

11. Zainstaluj program Midnight Commander do zarządzania plikami w oknie terminala:

    sudo apt install mc

12. Uruchom program Terminator. Podziel okno na trzy niezależne terminale (prawy przycisk myszy->Split Horizontally)

13. W wydzielonych oknach terminali uruchom kolejno (każda komenda w oddzielnym oknie terminala):

    roslaunch turtlebot_gazebo turtlebot_mw_playground.launch

    roslaunch turtlebot_gazebo amcl_demo.launch

    roslaunch turtlebot_rviz_launchers view_navigation.launch

    Zmniejsz i ustaw na ekranie okna Gazebo i RViz, tak aby oba były widoczne.

14. Uruchom nowe okno terminatora (z KickOff Application Lancher). W terminalu uruchom pakiet do sterowania ruchem robotem z klawiatury:

    roslaunch turtlebot_teleop keyboard_teleop.launch

    Wyświetlona zostanie m.in. lista klawiszy do sterowania robotem (przeczytaj te informacje).

    Zmniejsz mocno i przesuń okienko terminala tak, aby nie zasłaniało okien Gazebo i Rviz.

15. Steruj ruchem robota używając klawiatury (okno terminala z uruchomionym pakietem do sterowania musi być „na wierzchu” – kliknij je myszką). Obserwuj zachowanie robota w symulatorze Gazebo oraz w oknie do monitorowania pracy robota i jego sensorów (Rviz).

16. Wykonaj zrzut ekranu komputera (klawisz Print Screen) z uruchomionymi do tej pory aplikacjami. Plik zapisz w swoim katalogu domowym (będzie częścią sprawozdania z ćwiczenia).

17. Zapisz wartość liczbową z pola *Real Time Factor* w stopce okna Gazebo (po jej ustabilizowaniu lub jej przybliżoną wartość średnią). Wartość współczynnika jest stosunkiem czasu symulacji do czasu rzeczywistego. Wartość mniejsza od 1.00 oznacza, że symulator nie jest w stanie wykonywać obliczeń modeli w czasie rzeczywistym (komputer ma zbyt małą moc obliczeniową). Jeżeli wartość

współczynnika jest mniejsza od 0.97 wyłącz symulowanie cieni – w lewym panelu okna Gazebo: zakładka World -> Scene -> shadows = False. Zapisz wartość współczynnika po wyłączeniu cieni.

18. Uruchom aplikację System Monitor (z KickOff Application Lancher). Obserwuj stopień wykorzystania (obciążenia obliczeniami) procesorów logicznych. Wartości bliskie 100% dla wszystkich procesorów jednocześnie oznaczają zbyt małą wydajność obliczeniową komputera gospodarza.
19. Zamknij wszystkie programy w odwrotnej kolejności do uruchamiania – naciskaj Ctrl+C w kolejnych oknach terminali.
20. Kliknij (raz) ikonę *Gazebo PR2 Simulator* na pulpicie. Model robota PR2 jest bardzo złożony i wymaga wykonania czasochłonnych obliczeń przez symulator Gazebo. Zapisz wartość liczbową z pola *Real Time Factor* w stopce okna Gazebo (po jej ustabilizowaniu lub jej przybliżoną wartość średnią).
21. Dla dociekliwych:

    Zwiększ liczbę procesorów logicznych przydzielonych komputerowi gościa z 2 do 4 (wystarczy) – kliknij *Edit virtual machine settings* po wskazaniu wirtualnej maszyny *ROS Indigo Gazebo v3* w oknie *VMware Workstation Player.*

    UWAGA! Wirtualny komputer musi być uprzednio całkowicie wyłączony, tzn. nie może być w stanie Susspend OFF).

    Sprawdź wartości współczynnika *Real Time Factor* oraz obciążenie procesorów z nowymi ustawieniami.

<u>Potencjalne problemy:</u>

Podczas uruchomiania pakietów Gazebo lub RViz mogą pojawić się komunikaty o błędach (na czerwono), np.:

[gazebo_gui-3] process has died ...

[rviz-3] process has died ...

W takim przypadku należy zakończyć pracę pakietu przez Ctrl+C lub zamykając okno terminala, w którym uruchomiono pakiet. Ponownie uruchomić pakiet po wejściu w tryb pełnoekranowy (ewentualnie aż do skutku). Często powtarzający się problem należy odnotować w sprawozdaniu.

<u>Zestawienie komend uruchamiających pakiety ROS do symulacji i wizualizacji robota Turtlebot:</u>

=== Symulator GAZEBO z robotem i otoczeniem:

roslaunch turtlebot_gazebo turtlebot_mw_playground.launch

roslaunch turtlebot_gazebo turtlebot_mw_office.launch

roslaunch turtlebot_gazebo turtlebot_mw_empty_world.launch

=== Sterowanie robotem z klawiatury:
roslaunch turtlebot_teleop keyboard_teleop.launch

=== Pakiet wymagany do nawigacji (uruchamiać przed pakietami do wizualizacji RViz'a):
roslaunch turtlebot_gazebo amcl_demo.launch

=== Wizualizacja robota w RViz z zadawaniem punktu docelowego dla jazdy
roslaunch turtlebot_rviz_launchers view_navigation.launch

Pliki uruchamiające powyższe pakiety znajdują się w katalogu /opt/ros/indigo/share

AUTONOMICZNE ROBOTY TRANSPORTOWE

Ćwiczenie ART-2

Podstawy ROS

I. Cel ćwiczenia

Celem dydaktycznym jest zapoznanie z niezbędną (minimalną) konfiguracją zmiennych środowiskowych dla systemu ROS oraz z podstawowymi poleceniami diagnostycznymi ROS wydawanymi przez konsolę użytkownika (terminal tekstowy).

Celem praktycznym jest:

- prawidłowe skonfigurowanie zmiennych środowiskowych dla systemu ROS,
- uruchomienie prostego symulatora robota-żółwia poruszającego się po płaszczyźnie oraz węzła umożliwiającego sterowania robotem z klawiatury,
- monitorowanie uruchomionego systemu ROS poleceniami konsoli użytkownika oraz z użyciem programu *rqt_graph*.

II. Przebieg ćwiczenia

1. Uruchom wirtualną maszynę z zainstalowanymi systemami Linux i ROS.
2. W oknie terminala (używaj aplikacji Terminator) wykonaj komendę *printenv*, w wyniku której wyświetlone zostaną wszystkie zmienne środowiskowe w systemie Linux zdefiniowane w bieżącej sesji użytkownika (będzie ich wiele, między nimi te dla ROS'a).
3. Wykonaj komendę *printenv* ale tak, aby wyświetlone zostały tylko zmienne środowiskowe zawierające w sobie ciąg trzech liter: *ROS*. Wskazówka: skieruj strumień wyjściowy (tekst) z komendy *printenv* na wejście komendy *grep* przez utworzenie tzw. potoku strumienia danych, szczegóły:

   http://linuxwiki.pl/wiki/Potok

   http://linuxwiki.pl/wiki/Grep

   Wartość zmiennej ROS_MASTER_URI określa adres komputera oraz numer portu TCP, na którym pracuje centralny serwer nazw systemu ROS – o nazwie *Master* (uruchamiany komendą *roscore*). Nazwa *localhost* to komputer na którym pracujesz (pod systemem Linux). *localhost* jest tożsamy z adresem IP: 127.0.0.1.
4. Sprawdź czy twój komputer jest widoczny w sieci „dla samego siebie" – wykonaj dwie równoważne komendy:

   ping localhost (przerwij po chwili Ctrl+C)

ping 127.0.0.1 (przerwij po chwili Ctrl+C)

Wykonaj kopię ekranu z oknem terminala z widocznymi komendami punktów 3. i 4. oraz rezultatami ich wykonania. Plik dołącz do sprawozdania.

5. Wartość zmiennej ROS_IP określa adres komputera, na którym uruchamiane są węzły (ang. nodes) łączące się z Master'em. Wartość zmiennej ROS_IP jest jawnym adresem IP komputera w sieci, na którym uruchamiane są węzły (tj. lokalnie). Alternatywnie można podać nazwę komputera (np. *localhost*), ale wtedy należy użyć zmiennej ROS_HOSTNAME zamiast ROS_IP.

6. Wartości zmiennych ROS_MASTER_URI oraz ROS_IP muszą być prawidłowo nadane na każdym komputerze w sieci ROS. Załóżmy, że serwer nazw *Master* pracuje na komputerze o adresie IP: 172.16.10.10 (czyli nie na twoim komputerze), a Ty chcesz uruchamiać węzły ROS na swoim komputerze. Ustaw zatem prawidłową wartość zmiennej środowiskowej ROS_MASTER_URI komendą *export* – patrz:

   https://bash.0x1fff.com/zmienne/zmienne-systemowe.html

   Sprawdź, czy zmienna ROS_MASTER_URI ma nową wartość (użyj komendy *printenv nazwa_zmiennej*).

7. Zamknij okno terminala i uruchom ponownie konsolę użytkownika (terminal). Wyświetl wartość zmiennej ROS_MASTER_URI. Zmienna ma inną wartość niż 172.16.10.10, którą nadałeś w zamkniętej właśnie sesji. Przyczyną tego jest mechanizm systemu Linux, który wykonuje szereg komend zawsze gdy uruchamiasz nowe okno terminala, gdzie m.in. zmiennej ROS_MASTER_URI nadawana jest nowa wartość.

8. Zawsze gdy uruchamiany jest nowy terminal system Linux wykonuje m.in. komendy zapisane w dedykowanym pliku tekstowym o nazwie *.bashrc* w twoim katalogu domowym (/home/user/). Plik ten możesz edytować i wpisać w nim wiersz z komendą nadającą twoją wartość zmiennej ROS_MASTER_URI. W tym celu:

   - otwórz *Menedżer plików*: *KickOff Application Launcher -> Favorites -> File Manager*,

   - z paska narzędzi wybierz *Control -> Show Hidden Files* (plik *.bashrc* jest plikiem ukrytym, bo jego nazwa zaczyna się kropką),

   - w katalogu domowym utwórz katalog o nazwie *Kopie* (tam będziesz przechowywał kopie rezerwowe plików),

   - skopiuj plik *.bashrc* do katalogu *Kopie* i zmień jego nazwę na *.bashrc_oryginal*,

   - otwórz plik *.bashrc* z katalogu domowego – uruchomi się edytor tekstu *Kate* z zawartością pliku.

   Dzięki temu plikowi możesz „zautomatyzować" wykonywanie komend wydawanych normalnie z konsoli użytkownika. Wskazówka: znak # w treści pliku *.bashrc* rozpoczyna tzw. tekst komentarza (znaki w linii na prawo od # są ignorowane przez system Linux).

9. Przejdź na koniec zawartości pliku *.bashrc*. Dodaj w trzech nowych liniach komendy:

   - nadającą zmiennej ROS_MASTER_URI twoją wartość (tj. wg punktu 6.),

- *echo '*** LISTA ZMIENNYCH SRODOWISKOWYCH ROS ***'* (komenda *echo* wyświetla na ekranie tekst podany po niej, tekst musi być między apostrofami),
- wyświetlającą wszystkie zmienne środowiskowe z tekstem ROS (tj. wg punktu 3.).

Zapisz plik na dysku.

10. Uruchom nowe okno terminala. Powinieneś ujrzeć listę wartości zmiennych środowiskowych ROS'a poprzedzoną nagłówkiem: *** LISTA ZMIENNYCH SRODOWISKOWYCH ROS ***
11. Dobrą praktyką w ROS jest używanie jawnie adresu IP komputera w sieci zamiast adresu *localhost*. Aby zobaczyć adresy wszystkich interfejsów sieciowych w swoim komputerze wpisz komendę *ifconfig*. Powinieneś zobaczyć tekst podobny do poniższego:

    eth0 Link encap:Ethernet HWaddr 00:0c:29:49:96:ef

    inet addr:192.168.157.128 Bcast:192.168.157.255 Mask:255.255.255.0

    inet6 addr: fe80::20c:29ff:fe49:96ef/64 Scope:Link

    …

    eth0 to nazwa interfejsu sieciowego (kablowego ethernet'u), a jego adres IP wyświetlany jest po nazwie parametru *inet addr:* (w drugiej linii powyższego przykładu).
12. W pliku *.bashrc* nadaj zmiennej ROS_MASTER_URI wartość *http://adresIP:11311*, zastąp *adresIP* odczytanym wcześniej numerem IP interfejsu *eth0* (gdyż serwer nazw *Master* będzie uruchamiany na twoim komputerze).
13. W pliku *.bashrc* na początku linii *export ROS_IP=$(ifconfig…* wstaw znak komentarza, a poniżej nadaj zmiennej ROS_IP jawnie wartość *adresIP* twojego komputera. Zapisz plik. Otwórz nowe okno terminala i sprawdź czy zmienne ROS_MASTER_URI i ROS_IP mają poprawne wartości (*adresIP*).
14. Otwórz nowe okno terminala. Uruchom serwer nazw ROS *Master* komendą *roscore*. Przeczytaj uważnie wyświetlone komunikaty. Te w kolorze czerwonym są prawdopodobnie spowodowane błędami popełnionymi w trakcie realizacji poprzednich punktów (wykonaj ponownie konfigurację zmiennych środowiskowych w przypadku komunikatów o błędach).
15. W nowym oknie terminala wyświetl listę zarejestrowanych węzłów ROS (ang. nodes) komendą:

    rosnode list

    oraz listę publikowanych/nasłuchiwanych przez nie kanałów (ang. topics) komendą:

    rostopic list
16. W nowym oknie terminala uruchom węzeł będący prostym symulatorem robota mobilnego-żółwia na płaszczyźnie:

    rosrun turtlesim turtlesim_node

    (wyświetlone zostanie kwadratowe okno z żółwiem na środku).

17. Ponownie wyświetl listę zarejestrowanych węzłów ROS, a następnie wyświetl szczegółowe informacje o nowym węźle na liście (*/turtlesim*) komendą:

    rosnode info nazwa_węzła

    Przeczytaj dokładnie wyświetlone informacje. Węzeł symulatora robota-żółwia publikuje informacje do trzech kanałów (m.in. o swojej aktualnej pozycji i orientacji) oraz nasłuchuje (*Subscriptions*:) kanału z zadanymi prędkościami ruchu dla żółwia (publikowanymi przez inny węzeł, który uruchomisz później).

18. Wyświetl zawartość wiadomości publikowanej w kanale o nazwie */turtle1/pose* komendą:

    rostopic echo nazwa_kanału

    (wyświetlanie możesz przerwać przez Ctrl+C).

19. Uruchom nowe okno Terminala a w nim węzeł do sterowania symulowanym robotem komendą:

    rosrun turtlesim turtle_teleop_key

    (przyciski strzałek kursorów na klawiaturze sterują ruchem robota).

20. Ponownie wyświetl listę zarejestrowanych węzłów ROS, a następnie wyświetl szczegółowe informacje o właśnie uruchomionym węźle komendą:

    rosnode info nazwa_węzła

    Przeczytaj dokładnie wyświetlone informacje. Węzeł sterujący robotem publikuje wiadomości do kanału o nazwie */turtle1/cmd_vel* (skrót od ang. *commanded_velocity*), a kanału tego nasłuchuje węzeł symulatora robota (porównaj z pkt. 17.).

21. Wyświetl zawartość wiadomości publikowanych przez kanał o nazwie */turtle1/cmd_vel* komendą:

    rostopic echo nazwa_kanału

    Naciskaj klawisze kursorów w oknie węzła do sterowania robotem. Obserwuj treść wiadomości w kanale */turtle1/cmd_vel*.

22. Ponownie wyświetl zawartość wiadomości publikowanych przez kanał */turtle1/pose*. Przemieszczaj robota przy pomocy klawiatury i obserwuj treść wiadomości.

23. Wykonaj zrzut ekranu z widocznymi oknami:

    - symulatora robota (po wykonaniu kliku ruchów),

    - podglądem treści wiadomości w kanale */turtle1/pose*,

    - podglądem treści wiadomości w kanale */turtle1/cmd_vel*.

    <u>Plik dołącz do sprawozdania.</u>

24. Przerwij działanie komendy podglądu obu kanałów – Ctrl+C (pozostaw symulator robota i węzeł sterowania z klawiatury). Uruchom komendę:

    rqt_graph

W nowym oknie wyświetlony zostanie diagram ilustrujący uruchomione węzły ROS'a (owale) oraz kanały informacyjne (strzałki), przez które węzły przesyłają wiadomości. Z listy wyboru u góry okna wybierz *Nodes only*, a następnie *Nodes/Topics (All)* aby zmienić konwencję prezentacji diagramu. Przeanalizuj strukturę węzłów oraz sposób w jaki wymieniają ze sobą informacje.

25. Zamknij okno programu *rqt_graph*. W oknach terminali ponownie wyświetlaj wiadomości z kanałów */turtle1/pose* oraz */turtle1/cmd_vel* (komendy *rostopic echo …*).
26. Uruchom ponownie *rqt_graph*. Diagram powinien być rozszerzony w stosunku do poprzedniego o dwa nowe węzły nasłuchujące odpowiednich kanałów (w istocie komenda *rostopic echo …* tworzy nowy węzeł nasłuchujący określonego kanału).
27. Wykonaj zrzut ekranu z widocznym diagramem i dołącz do sprawozdania.

Pytanie sprawdzające

- Objaśnij role zmiennych systemowych ROS_MASTER_URI oraz ROS_IP

AUTONOMICZNE ROBOTY TRANSPORTOWE

Ćwiczenie ART-3

Program diagnostyczny rqt

I. Cel ćwiczenia

Celem dydaktycznym jest zapoznanie z najczęściej używanymi wtyczkami programu diagnostycznego systemu ROS o nazwie *rqt*.

Celem praktycznym jest uruchomienie symulatorów robotów (*TurtleSim* oraz *Gazebo*) oraz monitorowanie ich działania z użyciem wtyczek programu *rqt*: *Node Graph, Topic Monitor, Robot Steering, Image View*.

II. Przebieg ćwiczenia

1. Uruchom wirtualną maszynę z zainstalowanymi systemami Linux i ROS wraz z konfiguracją zmiennych środowiskowych ROS w pliku *.bashrc* wykonaną w trakcie poprzedniego ćwiczenia.
2. Uruchom serwer nazw ROS *Master* komendą: roscore
3. Uruchom węzeł symulatora prostego robota komendą: rosrun turtlesim turtlesim_node
4. Uruchom węzeł do sterowania robotem komendą: rosrun turtlesim turtle_teleop_key
5. Uruchom program diagnostyczny *rqt* komendą: rqt

   Program *rqt* dostarcza szereg narzędzi przydatnych w: monitorowaniu, diagnostyce oraz podstawowym zarządzaniu systemem pracującym w środowisku ROS (m.in. węzłami, kanałami, usługami). *rqt* posiada graficzny interfejs użytkownika, a uruchamiane moduły narzędzi mogą pracować równocześnie (w wielu oknach). Wszystkie moduły, nazywane plugin'ami (wtyczkami), uruchamiane są z pozycji *Plugins* w menu głównym programu *rqt*.
6. W oknie programu *rqt* uruchom wtyczkę *Introspection>Node Graph*. Wyświetlone zostanie okno z diagramem zarejestrowanych węzłów i kanałów znanym Ci z poprzedniego ćwiczenia (tą samą wtyczkę można uruchomić z linii komend poleceniem *rqt_graph*). Z rozwijanej listy u góry okna wtyczki (*Node Graph*) wybierz *Nodes Only* (wtedy na diagramie kanały rysowane są tylko jako strzałki „łączące" owalne bloki symbolizujące węzły).
7. W oknie terminala uruchom węzeł nasłuchiwania wiadomości z kanału */turtle1/pose* (komendą: rostopic echo /turtle1/pose).

   Zauważ, że pomimo uruchomienia nowego węzła, diagram w oknie *Node Graph* nie zmienił się. Aby zaktualizować diagram kliknij zieloną strzałkę (*Refresh ROS graph*) w lewym-górnym rogu okna *Node Graph*.

8. W programie *rqt* uruchom wtyczkę *Topics>Topic Monitor*. Wyświetlone zostanie nowe okno z listą (drzewem) wszystkich zarejestrowanych kanałów. Rozwiń szczegółowy widok struktury wiadomości w kanale */turtle1/pose* klikając znak > przed nazwą kanału. Wyświetlone zostaną nazwy pól (zmiennych) wewnątrz struktury wiadomości wraz z ich typami w kolumnie *Type*; typ *float32* oznacza liczbę zmiennoprzecinkową (rzeczywistą) zapisaną w słowie 32-bitowym.
9. Zaznacz szary kwadrat przed nazwą kanału */turtle1/pose*. Od tej chwili wyświetlane są aktualne wartości wszystkich zmiennych w strukturze wiadomości (pod warunkiem, że są nadawane do kanału przez co najmniej jeden węzeł).
10. Analogicznie rozwiń szczegóły wiadomości w kanale */turtle1/cmd_vel* i włącz monitorowanie wartości zmiennych tej struktury (*cmd_vel* to skrót od *commanded_velocity* czyli prędkość zadana). Zauważ, że nowe wartości zmiennych w strukturze wiadomości pojawiają się dopiero po naciśnięciu klawiszy kursorów w oknie terminala z uruchomionym węzłem *turtle_teleop_key* (węzeł nadaje wiadomości właśnie do tego kanału ale tylko przez chwilę po naciśnięciu klawisza kursora).
11. W programie *rqt* uruchom wtyczkę *Robot Tools>Robot Steering*. Wyświetlone zostanie nowe okno z dwoma potencjometrami suwakowymi, dzięki którym można zadawać i publikować wartości prędkości liniowej oraz kątowej do kanału o nazwie wpisanej w polu u góry okna wtyczki (domyślna nazwa kanału: */cmd_vel*, a typ wiadomości: *geometry_msg/Twist*).
12. Zmieniaj pozycje obu suwaków i obserwuj wartości zmiennych w wiadomości kanału */cmd_vel* (a ogólniej: w kanale o nazwie widocznej powyżej suwaków wtyczki *Robot Steering*).
13. Spraw, aby suwaki w oknie *Robot Steering* oddziaływały na ruch symulowanego robota w oknie *TurtleSim*. Wskazówka: odczytaj z diagramu pełną nazwę kanału, z którego węzeł symulujący robota *(/turtlesim*) odczytuje wartości zadane dla prędkości robota. Porównaj (i ewentualnie zmodyfikuj) nazwę kanału, do którego nadaje wtyczka *Robot Steering*.
14. Używając suwaków w oknie *Robot Steering* poprowadź robota po trajektorii o kształcie przypominającym spiralę lub ósemkę (np. ustaw prędkość liniową na wartość 1m/s a następnie „manewruj" suwakiem prędkości kątowej).
15. Wykonaj kopię ekranu z widocznymi oknami: *Node Graph, Topic Monitor, Robot Steering* oraz symulatora robota z przebytą trajektorią (wykorzystaj cały rozmiar pulpitu aby informacje wyświetlane w oknach były czytelne). Plik dołącz do sprawozdania.
16. Zamknij WSZYSTKIE uruchomione dotąd programy i węzły (włącznie z *Master'em*).
17. Uruchom symulator *Gazebo* z robotem *Turtlebot* (komenda *roslaunch* uruchomi między innymi serwer nazw ROS *Master* oraz kilka węzłów):

    roslaunch turtlebot_gazebo turtlebot_mw_office.launch

18. Uruchom program *rqt*. W oknie programu powinny pojawić się okienka wtyczek, które były ostatnio używane (jeżeli nie, to uruchom wtyczki: *Introspection>Node Graph, Topics>Topic Monitor, Tools>Robot Steering*). Przejrzyj listę kanałów (okno *Topic Monitor*). Te rozpoczynające się od

/camera/... dostarczają wiadomości z symulowanego stereoskopowego sensora odległości *Microsoft Kinect* (kamera 3D) zamontowanego na robocie.

19. Spraw, aby suwaki w oknie *Robot Steering* oddziaływały na ruch robota (analogicznie jak w pkt. 13 z tym, że węzeł symulatora robota ma nazwę */gazebo*).
20. W programie *rqt* uruchom wtyczkę *Visualization>Image View*. Wyświetlone zostanie nowe okno służące do podglądu wiadomości z obrazami 2D. Z rozwijanej listy kanałów u góry okna wtyczki wybierz */camera/rgb/image_raw*. *rgb* to skrót od wyrazów: *reg, green, blue* i oznacza, że wiadomość w tym kanale zawiera obraz kolorowy z tradycyjnej kamery 2D - *Microsoft Kinect* ma wbudowaną także tradycyjną kamerę cyfrową. W oknie wtyczki powinno być widoczne symulowane otoczenie robota widziane okiem kamery.

    Steruj ruchem robota i obserwuj obraz z symulowanej kamery.
21. W oknie wtyczki *Image View* wybierz podgląd obrazu z kanału */camera/depth/image_raw*. Wyraz *depth* (z ang. głębia) w nazwie kanału oznacza, że obraz jest dwuwymiarową reprezentacją zmierzonej odległości przedmiotów od obiektywów kamery stereoskopowej *Microsoft Kinect*. Odległość reprezentowana jest jako odcienie szarości pikseli: im odcień bliższy bieli, tym większa odległość (wyjątkiem jest idealna czerń – która reprezentuje obiekty poza zakresem pomiarowym sensora >8m – np. obszary odpowiadające otworom okiennym).
22. W programie *rqt* uruchom wtyczkę *Introspection>Process Monitor*. Wyświetlone zostanie nowe okno z listą uruchomionych węzłów oraz ich zapotrzebowaniem na zasoby komputera (CPU, RAM). Przyciskiem *Kill Node* (z ang. zabij węzeł) można selektywnie zakończyć działanie wybranego węzła (w przeciwieństwie do użycia *Ctrl+C* w oknie terminala, które powoduje zakończenie wszystkich węzłów uruchomionych komendą *launch*).
23. Uruchomione w programie *rqt* wtyczki wraz z ich aktualnym układem w oknie programu *rqt* można zapisywać na dysku komputera w plikach konfiguracji (ang. *Perspective*). Zapisaną wcześniej konfigurację (*Perspective*) można w każdej chwili przywołać (wczytać z dysku). Zapisz aktualną konfigurację pod swoją nazwą – *Menu Perspectives > Create perspective*. Po zapisaniu na dysku, w Menu *Perspectives* powinna być widoczna lista możliwych do wyboru konfiguracji (w tym także twoja – właśnie zapisana).
24. Wykonaj kopię ekranu z oknem programu *rqt* (ze wszystkimi uruchomionymi dotąd wtyczkami). Plik dołącz do sprawozdania.

Pytania sprawdzające

- Wymień co najmniej trzy wtyczki (plugins) programu diagnostycznego *rqt* i objaśnij ich przeznaczenie.

AUTONOMICZNE ROBOTY TRANSPORTOWE

Ćwiczenie ART-4

Pakiet tf. Program RViz

I. Cel ćwiczenia

Celem dydaktycznym jest zapoznanie z tworzeniem i zarządzaniem układami odniesienia (układami współrzędnych) w ROS oraz z rolą i głównymi komendami pakietu *tf* (od ang. *transform*).

Celem praktycznym jest utworzenie poprawnego drzewa relacji pomiędzy trzema łańcuchowo połączonymi układami współrzędnych oraz ich wizualizacja 3D w programie *RViz*.

II. Przebieg ćwiczenia

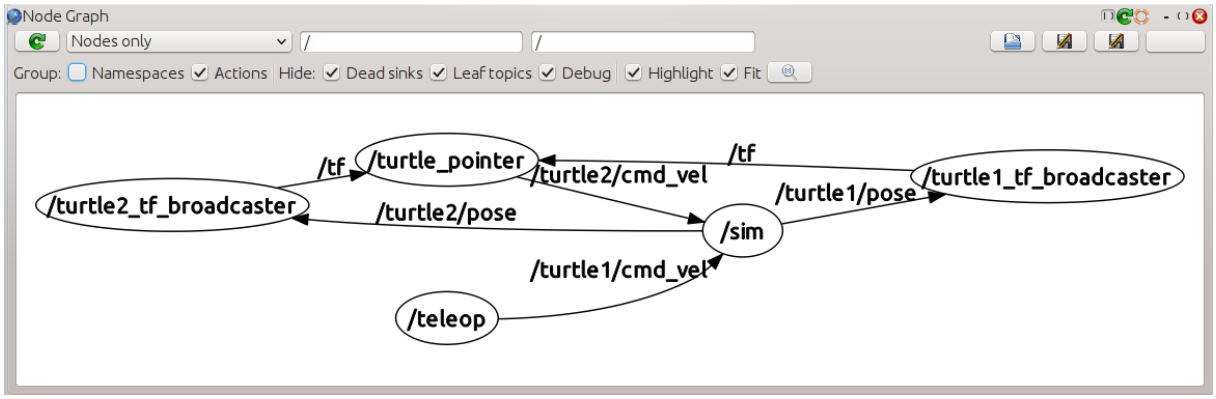
1. Uruchom wirtualną maszynę z zainstalowanymi systemami Linux i ROS.
2. Uruchom serwer nazw ROS Master komendą: *roscore*. Nie zamykaj okna tego terminala (z uruchomionym Master'em) aż do końca ćwiczenia.
3. Uruchom skrypt ROS'a z demonstracją działania pakietu *tf* (zarządzającego układami współrzędnych w przestrzeni 3D) komendą:

   roslaunch turtle_tf turtle_tf_demo.launch

   Komendy w pliku skryptowym *turtle_tf_demo.launch* uruchomią m.in. (rys. 1.):

   - prosty symulator (węzeł o nazwie */sim*) z dwoma robotami-żółwiami poruszającymi się w płaszczyźnie XY,
   - dwa węzły o nazwach */turtle1_tf_broadcaster* oraz /*turtle2_tf_broadcaster* publikujące położenie środka każdego żółwia oraz orientacji w globalnym układzie współrzędnych,
   - węzeł */teleop* do sterowania żółwiem nr 1 z klawiatury,
   - węzeł */turtle_pointer*, którego zadaniem jest automatyczne sterowanie żółwiem nr 2, tak aby zawsze doganiał i zajmował pozycję zółwia nr 1.
4. Z okna terminala, w którym uruchomiłeś skrypt *turtle_tf_demo.launch*, możesz sterować ruchem żółwia nr 1 przy pomocy czterech przycisków kursorów na klawiaturze. Obserwuj zachowanie żółwia nr 2.
5. Uruchom program diagnostyczny *rqt* komendą: *rqt*. W oknie programu prawdopodobnie zobaczysz okienka wtyczek uruchomione w poprzedniej sesji programu (*Perspective default*). Zamknij wszystkie okienka wtyczek w oknie *rqt*.
6. W programie *rqt* uruchom wtyczkę *Introspection>Node Graph*. Sprawdź, czy wyświetlony diagram jest zgodny (co do ogólnej struktury) z tym z rys. 1. Możesz zmienić sposób prezentacji struktury wybierając

różne opcje z rozwijanej listy u góry okienka wtyczki *Node Graph* – zalecane jest wybranie *Nodes Only* (wyświetlanie bloków-owali tylko dla węzłów) oraz wyłączenie pola *Group: Namespaces* (wyłączenie wyświetlania ramek z nazwami grup).



Rys. 1.

7. Przeanalizuj strukturę diagramu wracając do opisu uruchomionych węzłów w p. 3. Kanał */tf* jest dedykowany do przesyłania wiadomości o wzajemnych relacjach między układami współrzędnych (relacji *parent->child*).
8. W programie *rqt* uruchom wtyczkę *Visualization>TF Tree*. Wyświetlone zostanie okno z diagramem publikowanych układów współrzędnych oraz ich wzajemnymi relacjami.

   UWAGA! Elementy graficzne diagramu (bloki i strzałki) są podobne do elementów diagramu wtyczki *Node Graph*, jednak przedstawiają one zupełnie inną strukturę – tj. drzewo relacji układów odniesienia.

   Układ współrzędnych o nazwie *world* jest globalnym układem odniesienia (to kwestia umowy), związanym z płaszczyzną (podłogą), po której poruszają się symulowane żółwie.

   Układ współrzędnych o nazwie *turtle1* jest związany ze środkiem żółwia nr 1, a położenie początku układu i jego orientacja opisane są w układzie współrzędnych *world* przez: wektor translacji i kwaternion publikowane przez węzeł */turtle1_tf_broadcaster* do kanału */tf*. Para układów współrzędnych *world-turtle1* związane są relacją rodzic-dziecko (od ang. parent-child).

   Analogiczną relację tworzy para układów współrzędnych *world-turtle2*, gdzie początek i orientacja układu *turtle2* są związane ze środkiem żółwia nr 2. Położenie początku układu *turtle2* oraz jego orientacja względem układu *world* publikowane są przez węzeł */turtle2_tf_broadcaster*.
9. Uruchom komendę, dzięki której będziesz mógł monitorować położenie i orientację układu *turtle1* (związanego ze środkiem żółwia nr 1) w układzie odniesienia *world* (związanego z „podłogą" symulatora):

   rosrun tf tf_echo world turtle1

Używając kursorów klawiatury przemieść i zorientuj żółwia nr 1 w takie miejsce okna symulatora aby oba te układy współrzędnych pokryły się, tzn. aby wyświetlane wartości: wektora translacji i kątów rotacji były równe (lub bliskie) zerom. Najwygodniej jest obserwować miary kątów Euler'a wyrażone w stopniach kątowych – *in RPY (degree) [rot_X, rot_Y, rot_Z]*.

Odpowiedz na pytania:

a) W którym punkcie okna symulatora *TurtleSim* zlokalizowany jest początek układu współrzędnych *world*?

b) Wiedząc, że oś X układu współrzędnych *turtle1* skierowana jest „w przód" żółwia nr 1, odpowiedz: jak zorientowane są trzy osie X,Y,Z układu współrzędnych *world* względem płaszczyzny ekranu komputera? Przyjmij, że patrzysz na płaszczyznę ekranu zorientowaną pionowo i posługuj się pojęciami: w prawo, w lewo, w górę, w dół, za ekran, przed ekran. Pamiętaj także, że w kartezjańskim układzie współrzędnych obowiązuje reguła śruby prawoskrętnej.

Odpowiedzi załącz do pliku sprawozdania.

10. Wizualizację położenia i orientacji układów współrzędnych w przestrzeni 3D umożliwia program *RViz*. Uruchom go komendą:

rosrun rviz rviz -d `rospack find turtle_tf`/rviz/turtle_rviz.rviz

RViz to program narzędziowy systemu ROS służący do trójwymiarowej wizualizacji i analizy konfiguracji robotów, informacji z jego sensorów oraz monitorowania algorytmów nawigacji i sterowania robotami.

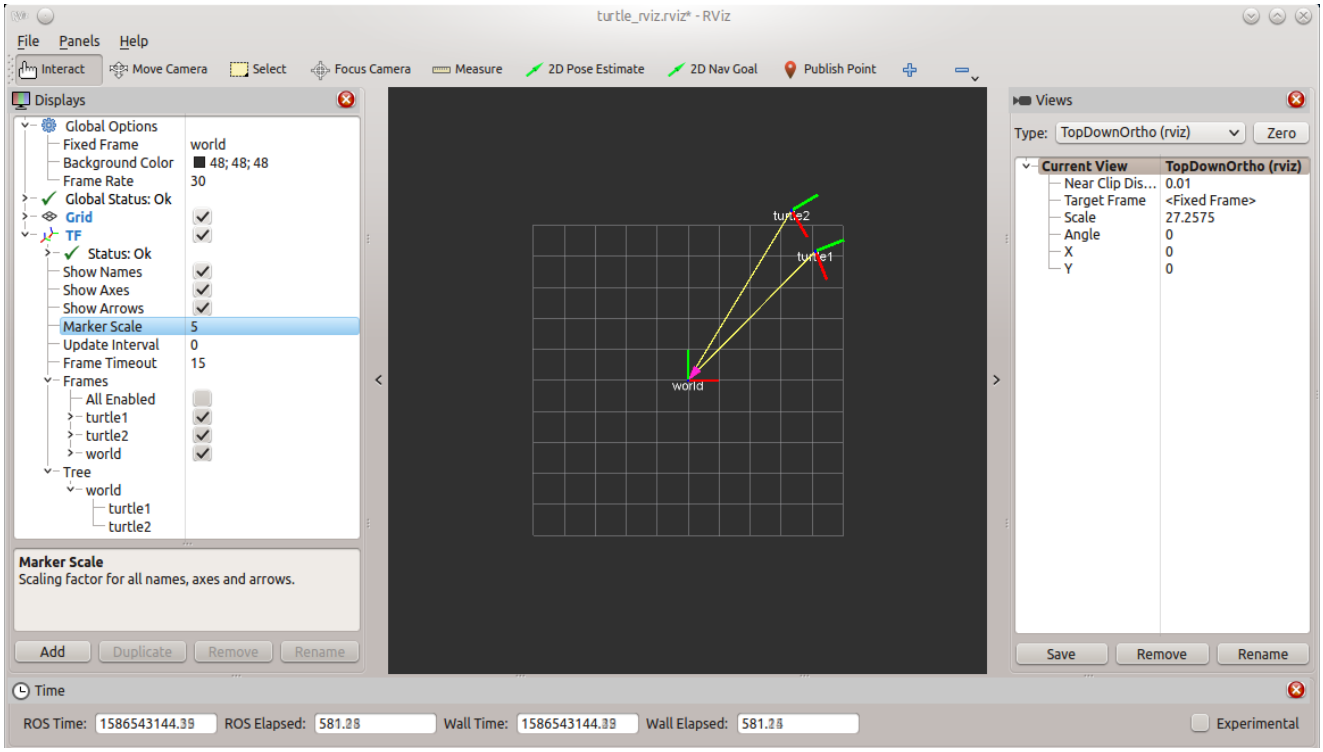
Okno programu *RViz* podzielone jest na trzy części (panele) – rys. 2.

Panel *Displays* po lewej stronie zawiera listę wizualizowanych obiektów wraz z konfiguracją sposobu ich wyświetlania.

Panel środkowy to widok wirtualnej przestrzeni 3D z wyświetlanymi w niej obiektami. Jednym z nich jest prostokątna siatka linii (ang. *Grid* - krata) „rozpięta" na płaszczyźnie XY umownego układu odniesienia (zwykle układ ten związany jest z podłożem robota lub płaszczyzną Ziemi).

Panel *Views* (po prawej) służy do pozycjonowania i orientowania wirtualnej kamery, która dostarcza widoku w środkowym panelu. UWAGA! Jeżeli panel *Views* nie jest wyświetlany, to wybierz z menu *Panels>Views*.

Na panelu *Views* (po prawej) wybierz z listy predefiniowanych ustawień kamery *Type: TopDownOrtho (rviz)* i kliknij przycisk *Zero* obok listy. Obiektyw wirtualnej kamery zostanie skierowany prostopadle „w dół - na podłogę".



Rys. 2.

11. W panelu *Displays* (po lewej) rozwiń listę właściwości obiektu *TF*. Zaznacz pola: *Show Names*, oraz *Show Arrows*. Parametr *Marker Scale* ustaw na 5. Rolką myszki nad środkowym panelem możesz zbliżać i oddalać kamerę od płaszczyzny z siatką XY.

    Obiekt *TF* wyświetla w wirtualnej przestrzeni 3D układy współrzędnych kartezjańskich, każdy w postaci trzech ortogonalnych odcinków: oś X na czerwono, oś Y na zielono, oś Z na niebiesko. Żółte strzałki między początkami układów współrzędnych to wektory translacji określające wzajemną relację w parze układów parent->child (zgodnie z wiadomościami publikowanymi w kanale */tf*).

    UWAGA! W programie *RViz* grot wektora translacji skierowany jest do początku układu nadrzędnego (parent), tj. przeciwnie niż na diagramie generowanym wtyczką *TF Tree*.

12. Przemieszczaj żółwia nr 1 w symulatorze przy pomocy klawiatury. Obserwuj ruch żółwi w oknie symulatora oraz okno programu *RViz*.

13. W panelu *Displays* (po lewej) rozwiń listę właściwości obiektu *Grid* (siatka). Dobierz wartości parametrów:

    Offset.X (przesunięcie[m] środka siatki w osi X),

    Offset.Y (przesunięcie[m] środka siatki w osi Y),

PlaneCellCount (liczba oczek siatki wzdłuż każdej osi),

CellSize (długość boku [m] oczka siatki),

tak, aby wyświetlana siatka (jej położenie i rozmiar) pokrywała się DOKŁADNIE z obszarem dostępnym dla ruchu żółwi w oknie symulatora *TurtleSim*.

14. W panelu *Displays* rozwiń listę właściwości obiektu *Global Options*. Parametr *Fixed Frame* (układ nieruchomy) określa, który z układów współrzędnych traktowany jest jako globalny (nieruchomy) układ odniesienia dla wizualizacji obiektów w oknie *RViz*. Z reguły jest to układ związany z Ziemią lub podłogą (do tej pory był to układ o nazwie *world*).

    Zmień wartość parametru *Fixed Frame* na nazwę układu współrzędnych *turtle1* (związanego ze środkiem żółwia nr 1). Przemieszczaj żółwia nr 1 w symulatorze przy pomocy klawiatury. Obserwuj ruch żółwi w oknie symulatora oraz okno programu *RViz*.

15. Zauważ, że w wyniku zmiany układu współrzędnych w parametrze *Fixed Frame*, początek układu związanego z żółwiem nr 2 jak i układu *world* „wychodzą” poza siatkę XY, która przedtem pokrywała się z obszarem dostępnym dla ruchu żółwi. Aby przywrócić właściwą formę prezentacji siatki ustaw wartość parametru *Reference Frame* w obiekcie *Grid* na *world* (w ten sposób siatka *Grid* będzie rysowana względem układu *world* - jak wcześniej).

    Przemieszczaj żółwia nr 1 w symulatorze. W oknie *RViz* układy współrzędnych związane z dwoma żółwiami nie powinny wychodzić poza siatkę pomimo, że siatka (związana z układem *world*) obraca się i przesuwa względem układu *turtle1* unieruchomionego dla celów wizualizacji.

    Wykonaj kopię ekranu z widocznymi oknami: symulatora *TurtleSim* oraz *RViz*. Plik załącz do sprawozdania.

16. Dotąd współrzędne środków symulowanych żółwi oraz ich orientacje (kąty obrotu) wyrażone były w układzie odniesienia *world* związanym z oknem symulatora *TurtleSim*. Załóżmy, że interesować nas będą współrzędne (x,y,z) środka żółwia nr 1 oraz jego orientacja ale wyrażone w nowym układzie współrzędnych związanym z ekranem twojego komputera.

    Zdefiniujmy zatem (na razie tylko koncepcyjnie) nowy układ kartezjański o nazwie *screen_ab* związany z ekranem. Niech początek nowego układu (0,0,0) będzie zlokalizowany w lewym dolnym narożniku wyświetlacza LCD.

    UWAGA! Litery *ab* w nazwie układu *screen_ab* zastąp swoimi inicjałami (imię, nazwisko).

    Niech oś X układu *screen_ab* pokrywa się z dolną krawędzią wyświetlacza i niech będzie skierowana w prawo.

    Niech oś Y układu *screen_ab* pokrywa się z lewą pionową krawędzią wyświetlacza i niech będzie skierowana do góry.

    Z zasady prawoskrętności układu kartezjańskiego wynika, że oś Z nowego układu musi być skierowana „do ciebie”.

17. Okno symulatora *TurtleSim*, z którym związany jest układ współrzędnych *world*, może zajmować dowolną pozycję na płaszczyźnie XY wyświetlacza ekranu – tj. w nowym układzie odniesienia *screen_ab*. Aby opisać nową relację pary układów współrzędnych *screen_ab-world* należy uruchomić kolejny węzeł typu *TransformBroadcaster*, który będzie publikował położenie i orientację układu *world* (dziecka) w nowym układzie współrzędnych *screen_ab* (rodzica). Ogólna składnia polecenia uruchamiającego taki węzeł wygląda następująco:

    rosrun tf static_transform_publisher x y z yaw pitch roll frame_id child_frame_id period_in_ms

    gdzie argumenty oznaczają:

    x y z – składowe wektora translacji (w metrach) początku układu dziecka względem początku układu rodzica,
    yaw – kąt obrotu (w radianach) układu dziecka wokół jego osi Z,
    pitch – kąt obrotu (w radianach) układu dziecka wokół jego osi Y,
    roll – kąt obrotu (w radianach) układu dziecka wokół jego osi X,
    frame_id – nazwa układu współrzędnych rodzica (*screen_ab*),
    child_frame_id – nazwa układu współrzędnych dziecka (*world*),
    period_in_ms – okres (w milisekundach) pomiędzy kolejnymi wiadomościami nadawanymi do kanału */tf* lub */tf_static* (wystarczy 100).

    Więcej informacji pod adresem: *http://wiki.ros.org/tf#static_transform_publisher* (w punkcie 6.3).

    Zatem dla naszych potrzeb komenda przyjmie prawie ostateczną postać:

    rosrun tf static_transform_publisher x y z yaw pitch roll screen_ab world 100

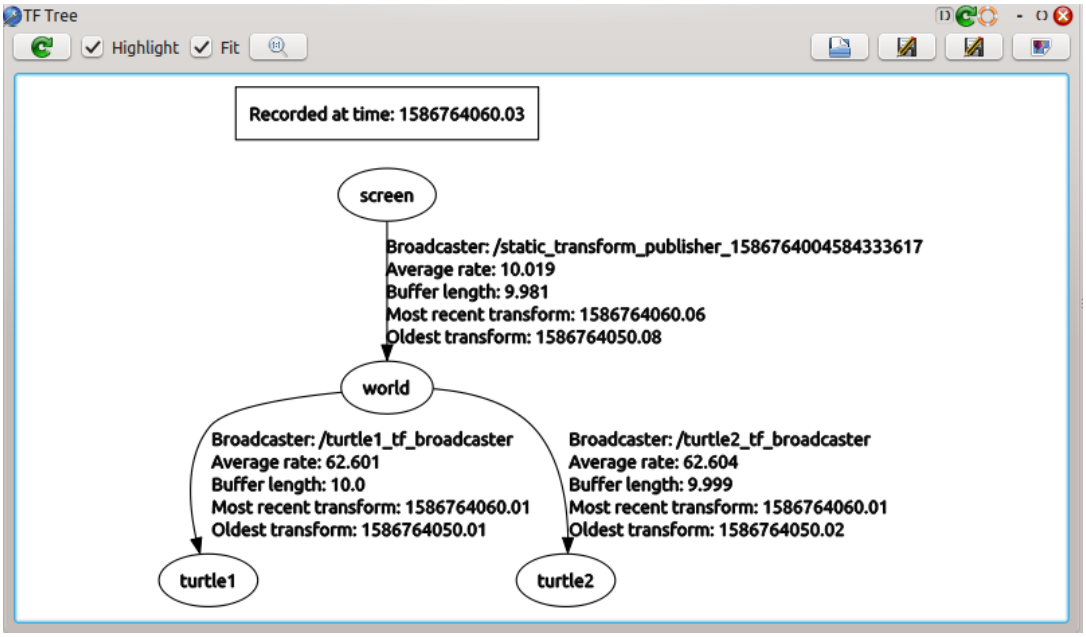
    gdzie odległości *x y z* zmierzysz linijką na ekranie komputera (oczywiście dla ustalonej pozycji okna *TurtleSim*). Wyznaczenie kątów obrotu wokół osi *yaw pitch roll* układu *world* jest zadaniem trywialnym.

    UWAGA! Wartości *x y z* podawaj w metrach ale jako liczby niemianowane (nie wpisuj jednostki „m”). Podobnie kąty obrotu podawaj w radianach także jako liczby niemianowane, np.:

    rosrun tf static_transform_publisher 10.4 -3.2 0.8 3.14 0 1.57 screen_ab world 100

18. Po uruchomieniu nowego węzła (w poprzednim punkcie) kliknij zieloną strzałkę *Refresh TF Tree* w okienku wtyczki *TF Tree* programu *rqt*. Powinieneś zobaczyć nowy diagram (drzewo) z nową relacją między parą układów współrzędnych *screen_ab-world* (rys.3.).

    UWAGA! Pamiętaj, aby w tym samym czasie działał TYLKO JEDEN WĘZEŁ publikujący nową relację układów *sreen-world*. Gdy chcesz zmienić parametry relacji to zakończ działanie poprzednio uruchomionego węzła (CTRL+C) i uruchom ponownie węzeł z nowymi argumentami.



Rys. 3.

19. Przejdź do okna *RViz*. W środkowym panelu powinny pojawić się trzy osie nowego układu współrzędnych o nazwie *screen_ab*.

    Ustaw układ *screen_ab* jako globalny (nieruchomy) układ odniesienia – patrz parametr *Fixed Frame* obiektu *Global Options* na panelu *Displays*.

    Sprawdź, czy siatka linii XY jest wyświetlana układzie *world* – patrz parametr *Reference Frame* obiektu *Grid*.

20. Na panelu *Views* (po prawej) wybierz z listy predefiniowanych ustawień kamery *Type: Orbit (rviz)* – teraz za pomocą myszki możesz dowolnie zmieniać orientację wirtualnej kamery programu *RViz*.

    Zweryfikuj, czy układ *world* jest zorientowany w nowym układzie odniesienia (*screen_ab*) tak jak w rzeczywistości okno *TurtleSim* względem dolnego lewego rogu wyświetlacza twojego ekranu (tj. podobnie do rys. 4.). Jeżeli nie, to zmodyfikuj parametry translacji lub/i obrotów (punkt 18.).

    UWAGA! Symulowana podłoga, po której poruszają się żółwie, to kwadrat o boku 11 metrów (i taki rozmiar powinna mieć siatka *Grid* rozpięta w układzie *world*). Oczywiście rozmiar okna *TurtleSim* (z podłogą dla żółwi) na ekranie komputera jest około 100 razy mniejszy.

21. Załóżmy, że płaszczyzna ekranu twojego monitora jest prostopadła do prostokątnego blatu stołu, na którym stoi monitor i równoległa do krawędzi blatu, przed którą siedzisz.

    Zdefiniujmy (na razie tylko koncepcyjnie) kolejny układ odniesienia o nazwie *table_ab* związany z

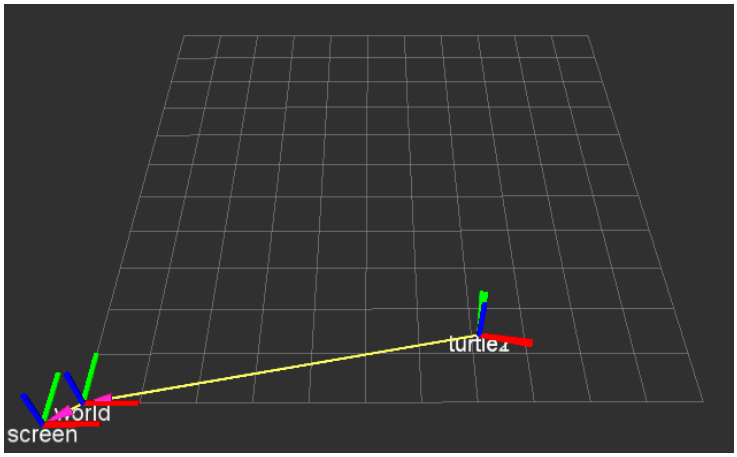
blatem stołu. Niech początek nowego układu (0,0,0) będzie zlokalizowany w narożniku blatu po twojej lewej stronie, bliżej ciebie.

UWAGA! Litery *ab* w nazwie układu *table_ab* zastąp swoimi inicjałami (imię, nazwisko).

Niech oś X układu *table_ab* pokrywa się z krawędzią blatu przed tobą (dodatni zwrot w prawo).

Niech oś Y układu *table_ab* pokrywa się z krawędzią blatu prostopadła do osi X i niech będzie skierowana „od ciebie".

Z zasady prawoskrętności układu kartezjańskiego wynika, że oś Z nowego układu musi być skierowana pionowo w górę.



Rys. 4.

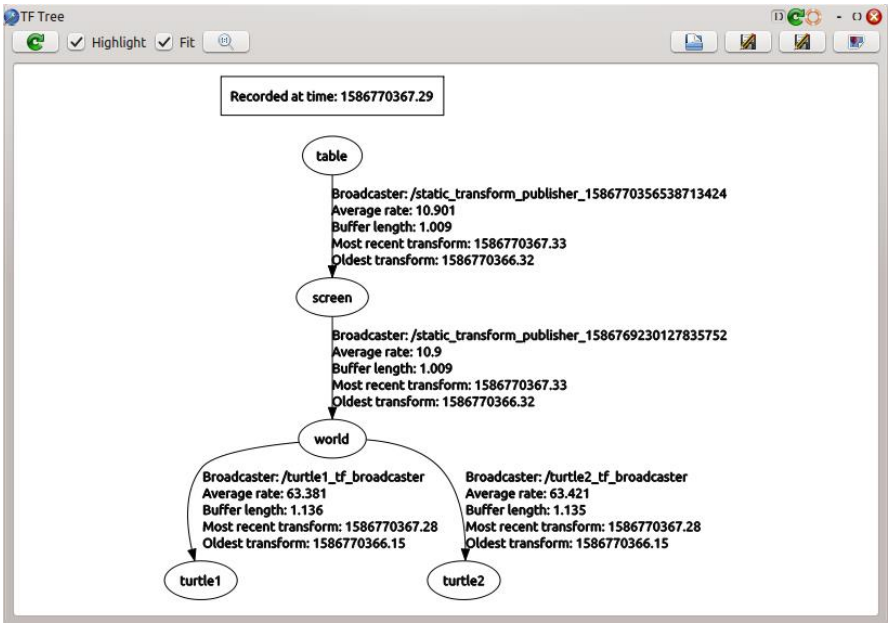
22. Załóżmy, że interesować nas będą współrzędne (x,y,z) środka żółwia nr 1 oraz jego orientacja ale wyrażone w układzie współrzędnych *table_ab* związanym z blatem stołu (czyli względem narożnika blatu). W tym celu należy uruchomić kolejny węzeł typu *TransformBroadcaster*, który będzie publikował położenie i orientację układu *screen_ab* (dziecka) w nowym układzie współrzędnych *table_ab* (rodzica).

    Polecenie uruchamiające nowy węzeł będzie analogiczne do tego z punktu 18:

    rosrun tf static_transform_publisher x y z yaw pitch roll table_ab screen_ab 100

    gdzie odległości *x y z* musisz z grubsza oszacować (lub zmierzyć linijką), a wartości kątów Euler'a *yaw pitch roll* wydedukować z prostej analizy przestrzennej (zauważ, że osie X układów *table_ab* i *screen_ab* są do siebie równoległe i mają ten sam zwrot – „w prawo").

23. Odśwież diagram w okienku wtyczki *TF Tree* programu *rqt*. Diagram relacji układów powinien wyglądać tak jak na rys.5.



Rys. 5.

24. Przejdź do okna *RViz*. W środkowym panelu powinny pojawić się trzy osie nowego układu współrzędnych o nazwie *table_ab*.

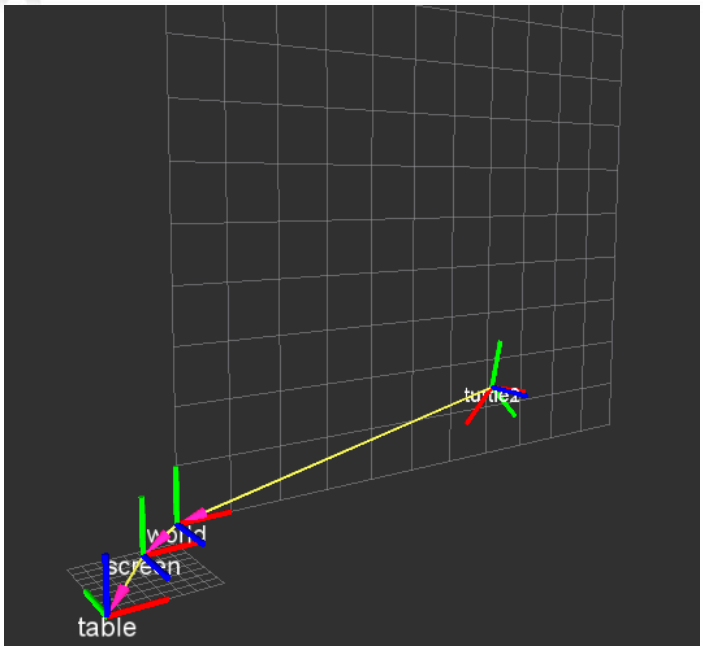
    Ustaw układ *table_ab* jako globalny (nieruchomy) układ odniesienia – patrz parametr *Fixed Frame* obiektu *Global Options* na panelu *Displays*.

25. W panelu *Displays* kliknij przycisk *Add* (dodaj obiekt do wyświetlania), a następnie z listy typów obiektów wybierz *Grid* (siatka). Niech nowy obiekt graficzny wyświetla drugą siatkę linii w płaszczyźnie blatu stołu (płaszczyzna XY układu współrzędnych *table_ab*).

    Ustaw rozmiar oczka (atrybut *CellSize*) nowej siatki na 0.25m, a rozmiar siatki 8x8 komórek. Niech narożnik siatki zaczyna się w początku układu współrzędnych *table_ab* (tak jak prawdziwy blat stołu).

    Zweryfikuj, czy wszystkie układy współrzędnych są ustawione w przestrzeni tak jak w rzeczywistości (płaszczyzny XY wyświetlacza LCD oraz okna symulatora *TurtleSim* powinny być prostopadłe do blatu stołu i znajdować się ponad nim) – podobnie do rys. 6. Popraw ewentualne błędy.

    Wykonaj kopię ekranu z oknem RViz. Uprzednio ustaw wirtualną kamerę w taki sposób, aby dało się ocenić wzajemne położenie układów współrzędnych (podobnie jak na rys. 6.). Plik dołącz do sprawozdania



Rys. 6.

26. Mając poprawnie zdefiniowane relacje miedzy układami współrzędnych można otrzymać współrzędne początku dowolnego z nich oraz jego orientację wyrażone względem dowolnego innego układu współrzędnych. I tak komenda:

    rosrun tf tf_echo table_ab world

    wyświetla składowe wektora translacji oraz kąty obrotu układu *world* (związanego z lewym dolnym narożnikiem okna symulatora *TurtleSim*) wyrażone w układzie odniesienia *table_ab* (czyli względem narożnika blatu stołu), w formacie:

    - Translation: [x, y, z]
    - Rotation: in Quaternion [qx, qy, qz, qw]
      in RPY (radian) [roll, pitch, yaw]
      in RPY (degree) [roll, pitch, yaw]

    UWAGA! Wyświetlana kolejność kątów Euler'a: *roll, pitch, yaw* jest odwrócona w stosunku do kolejności parametrów podawanych w komendzie uruchamiającej węzeł *TransformBroadcaster* – porównaj z punktem 18.

    Skopiuj i załącz do sprawozdania cyfrowego wynik wykonania powyższej komendy w twoim systemie ROS.

27. Wyświetl składowe wektora translacji oraz kąty obrotu układu *turtle1* (związanego ze środkiem żółwia nr1) wyrażone w układzie odniesienia *table_ab* (czyli względem narożnika blatu stołu).

    Skopiuj i załącz do sprawozdania cyfrowego wynik wykonania komendy w twoim systemie ROS.

28. Zapisz swoją konfigurację programu *RViz* na dysku wybierając z menu *File>Save Config As* (zapisywać pliki możesz tylko w swoim katalogu domowym */home/user*…).

    Zapisaną na dysku konfigurację możesz wczytać wybierając z menu programu *RViz*: *File>Open Config*.

    Program *RViz* możesz uruchomić poleceniem *rviz*.

Pytania sprawdzające

- Objaśnij przeznaczenie węzła *static_transform_publisher* z pakietu *tf* oraz wymień argumenty przekazywane węzłowi w chwili jego uruchamiania.

AUTONOMICZNE ROBOTY TRANSPORTOWE

Ćwiczenie ART-5

Polecenie rosrun

I. Cel ćwiczenia

Celem dydaktycznym jest zapoznanie z podstawowym poleceniem uruchamiającym węzeł w systemie ROS oraz z tzw. mapowaniem nazw.

Celem praktycznym jest uruchomienie kilku węzłów (z odpowiednim mapowaniem nazw) tworzących dwa niezależne podsystemy z symulatorami robotów-żółwi.

II. Składnia polecenia *rosrun*

Polecenie *rosrun* służy do uruchomiania jednego węzła w systemie ROS, tj. samodzielnego programu, najczęściej pod postacią pliku wykonywalnego (ewentualnie pliku skryptu wykonywanego przez interpreter języka Python). Ogólna składnia polecenia *rosrun*:

rosrun nazwa_pakietu nazwa_typu_wezla domyslna_nazwa_1:=nowa_nazwa_1 domyslna_nazwa_2:=nowa_nazwa_2

gdzie:

nazwa_pakietu – jest nazwą katalogu na dysku z pakietem ROS (pakiet ROS ma określoną strukturę wewnętrzną podkatalogów, zawiera pliki wykonywalne węzłów, może zawierać ich kody źródłowe i inne pliki);

nazwa_typu_wezla – jest nazwą typu węzła w pakiecie *nazwa_pakietu*, w pakiecie mogą być węzły różnych typów, ważne: nazwa typu węzła nie jest tożsama z nazwą, pod którą będzie pracował uruchomiony węzeł;

kolejne argumenty polecenia *rosrun* są opcjonalne i służą do nadawania nazwy uruchamianemu węzłowi oraz nadawania nazw kanałom, przez które węzeł wymienia informacje (szczegóły w dalszej części instrukcji).

Przykład:

rosrun turtlesim turtlesim_node /turtlesim:=symulator1

III. Przebieg ćwiczenia

1. Uruchom wirtualną maszynę z zainstalowanymi systemami Linux i ROS.
2. Uruchom serwer nazw ROS Master komendą *roscore*
3. Uruchom węzeł symulujący prostego robota-żółwia komendą:

rosrun turtlesim turtlesim_node

Zauważ, że powyższa komenda *rosrun* uruchamia węzeł typu *turtlesim_node* z pakietu o nazwie *turtlesim* (porównaj z ogólną składnią komendy *rosrun*). Pakiet *turtlesim* jest zlokalizowany w katalogu: */opt/ros/indigo/share* (obok kilkuset innych pakietów ROS w tym katalogu).

4. Uruchom program diagnostyczny *rqt* komendą: *rqt*. W oknie programu prawdopodobnie zobaczysz okienka wtyczek uruchomione w poprzedniej sesji programu (*Perspective default*). Zamknij wszystkie okienka wtyczek w oknie *rqt*.

5. W programie *rqt* uruchom wtyczkę *Introspection>Node Graph*. Na diagramie powinieneś zobaczyć jeden węzeł (owal) o nazwie */turtlesim*. */turtlesim* jest domyślną nazwą węzła typu *turtlesim_node* (nazwa ta jest „zaszyta” w kodzie programu wykonywalnego tego węzła).

   Wskazówka: wyczyszczenie pola „zaznaczenia” przed *Namespaces* (u góry okna *Node Graph*) uprości diagram.

   Zakończ działanie uruchomionego węzła */turtlesim* (Ctrl+C).

6. Załóżmy, że należy jednocześnie uruchomić więcej niż jeden węzeł z symulatorem żółwia, każdy pod inną nazwą. Można to zrobić na dwa sposoby:

   a) Gdy znasz domyślną nazwę nadawaną węzłowi typu *turtlesim_node* (w tym przypadku */turtlesim*) możesz wykonać tzw. mapowanie (ang. *remap*) nazwy domyślnej na nową podając szczegóły w kolejnym (trzecim) argumencie komendy *rosrun*:

      rosrun turtlesim turtlesim_node /turtlesim:=symulator1

      znak mapowania *:=* zastępuje pierwotną nazwę (po lewej stronie znaku) nową nazwą (po prawej stronie znaku).

      Uruchom powyższą komendę i odśwież diagram w programie *rqt* (zielona strzałka *Refresh ROS graph* u góry wtyczki *Node Graph*). Na diagramie powinien być widoczny jeden węzeł o nazwie */symulator1*.

   b) Gdy nie znasz domyślnej nazwy nadawanej węzłowi, to możesz wykonać mapowanie podobnie jak wyżej, ale używając specjalnego symbolu *__name* w miejscu nazwy domyślnej węzła:

      rosrun turtlesim turtlesim_node __name:=symulator2

      __name (dwa znaki podkreślenia na początku) jest specjalnym symbolem oznaczającym domyślną nazwę węzła.

      Uruchom powyższą komendę i odśwież diagram w programie *rqt*. Powinien pojawić się drugi węzeł o nazwie */symulator2*.

7. W programie *rqt* uruchom wtyczkę *Topics>Topic Monitor*. Na liście zarejestrowanych kanałów widoczny jest kanał */turtle1/pose*, do którego węzeł symulatora żółwia wysyła wiadomość z pozycją i orientacją żółwia.

   W nowym oknie terminala nasłuchuj kanału */turtle1/pose* komendą:

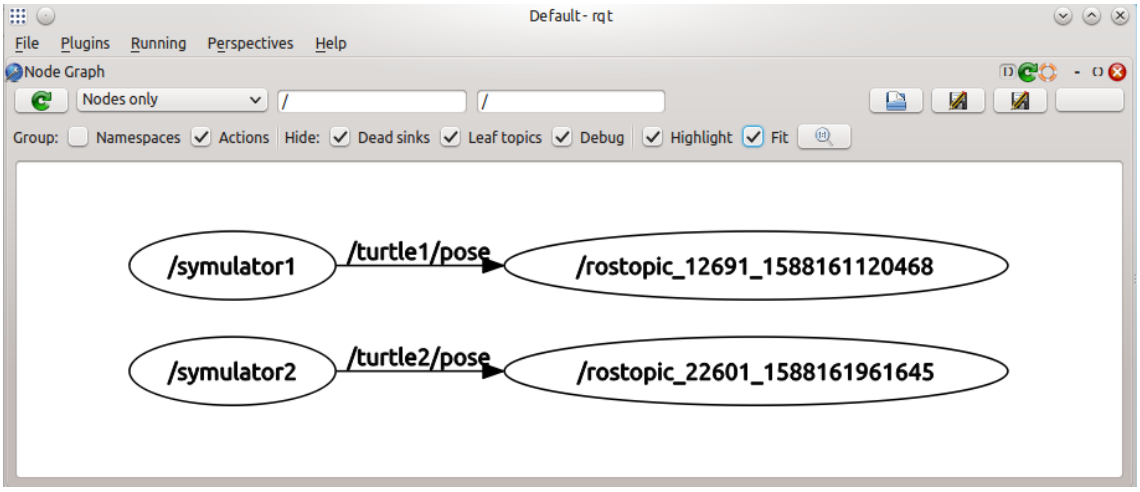
rostopic echo nazwa_kanału

8. Odśwież diagram w programie *rqt*. Zobaczysz, że oba węzły symulatorów (*/symulator1* i */symulator2*) wysyłają wiadomości do TEGO SAMEGO kanału (pomimo, że na diagramie widoczne są dwie strzałki podpisane */turtle1/pose*, to jest to jeden - ten sam kanał). Dzieje się tak dlatego, że w kodzie wykonywalnym węzła symulatora żółwia (typu *turtlesim_node*) „zaszyta" jest domyślna nazwa kanału, do którego węzeł publikuje wiadomości (tj. */turtle1/pose*). W uruchomionej właśnie strukturze systemu ROS nie jest możliwe „odróżnienie" (oddzielenie) wiadomości o pozycjach obu żółwi.

9. Logicznym wydaje się, aby oba symulatory żółwi wysyłały wiadomości o swojej pozycji i orientacji do oddzielnych kanałów (tj. o różnych nazwach). Do tego celu można wykorzystać znany Ci już mechanizm mapowania nazw:

   - Zakończ działanie węzła */symulator2* (Ctrl+C).
   - Uruchom ponownie węzeł drugiego symulatora (analogicznie jak w p. 6) ale dodając w komendzie *rosrun* kolejny (czwarty) argument: *turtle1/pose:=turtle2/pose*. W ten sposób uruchomiony węzeł */symulator2* powinien nadawać wiadomości do nowego kanału o nazwie */turtle2/pose* (zamiast do domyślnego */turtle1/pose*).
   - Odśwież diagram w programie *rqt*. Zauważ, że nowy węzeł */symulator2* nie nadaje do kanału */turtle1/pose*. Jednak na diagramie nie widać nowego kanału */turtle2/pose*. Wtyczka *Node Graph* narysuje strzałkę z nazwą kanału dopiero gdy będzie działała para węzłów nadawca-odbiorca (ang. publisher-subscriber). Dlatego w nowym oknie terminala uruchom węzeł nasłuchujący wiadomości z kanału */turtle2/pose*:

     rostopic echo nazwa_kanału
   - Odśwież diagram w programie *rqt*. Diagram powinien przypominać ten z rys. 1. (dwa węzły symulatorów połączone niezależnymi kanałami z dwoma węzłami */rostopic…*)

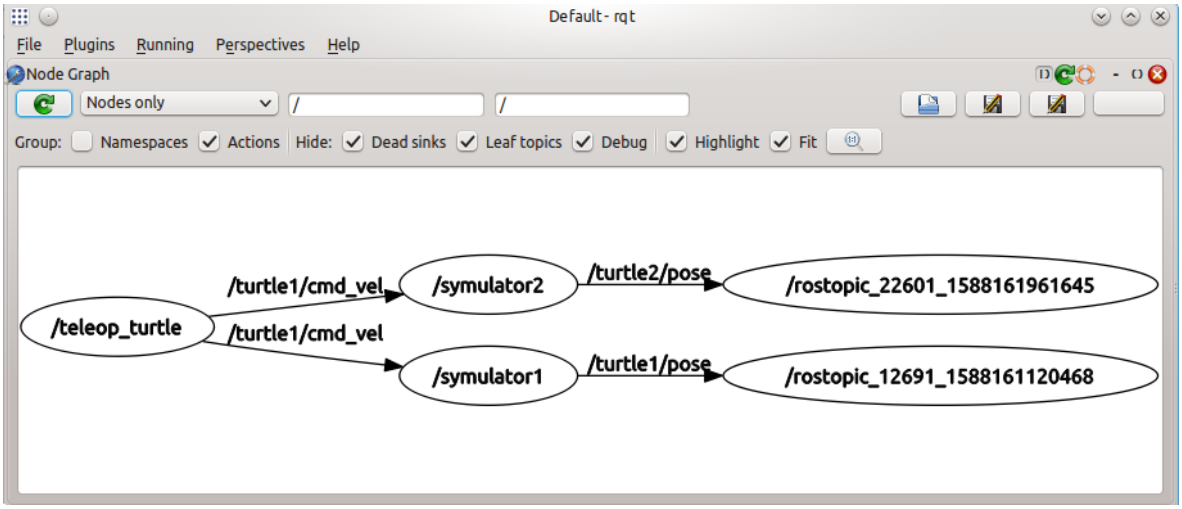


Rys. 1.

10. Węzeł symulatora żółwia (typu *turtlesim_node*) nasłuchuje wiadomości z komendami ruchu z kanału o domyślnej nazwie: */turtle1/cmd_vel*.

    Do kanału o tej samej domyślnej nazwie *(/turtle1/cmd_vel)* nadaje wiadomości węzeł typu *turtle_teleop_key* z pakietu *turtlesim* (węzeł umożliwia sterowanie ruchem żółwia klawiszami kursorów). Uruchom zatem ten węzeł komendą *rosrun* (patrz ogólna składnia komendy *rosrun*).

    Odśwież diagram w programie *rqt*. Zauważ, że uruchomiony węzeł o domyślnej nazwie */teleop_turtle* steruje jednocześnie żółwiami w obu symulatorach (gdyż oba węzły symulatorów nasłuchują tego samego kanału) – rys. 2. (sprawdź także reakcje obu żółwi w oknach symulatorów na klawisze kursorów).



Rys. 2.

11. Zakończ działanie wszystkich uruchomionych dotąd programów (węzłów, *rqt*, *ROS Master*).

12. Załóżmy, że wymagane jest <u>niezależne sterowanie dwoma żółwiami przez dwa niezależne węzły typu *turtle_teleop_key*</u> (oczywiście uruchomione w dwóch oddzielnych oknach terminala).

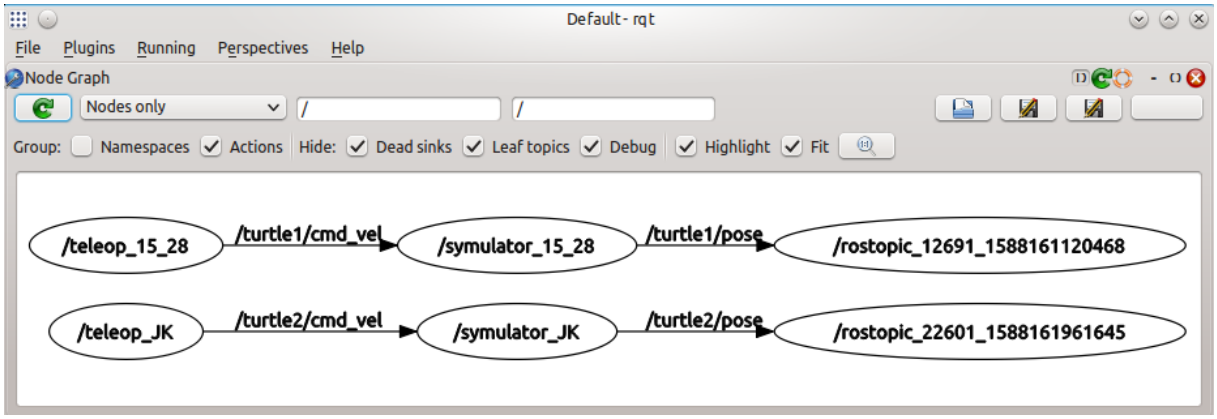
    Utwórz (od początku) strukturę węzłów połączoną kanałami informacji, która zrealizuje powyższe założenie. Twoja docelowa struktura systemu ROS powinna przypominać tą z rys. 3.

    UWAGA! W nazwie węzłów dla pierwszego symulatora żółwia użyj godziny i minuty rozpoczęcia realizacji ćwiczenia (na rys. 3. jest to _15_28 od 15:28).

    UWAGA! W nazwie węzłów dla drugiego symulatora żółwia użyj swoich inicjałów (na rys. 3. jest to *_JK* od *Jan Kowalski*).

    WSKAZÓWKI:

- Uruchamiaj kolejne węzły (w sumie 4) komendą *rosrun* z argumentami mapującymi ich domyślne nazwy (*__name*) oraz, w zależności od potrzeb, domyślne nazwy kanałów (*turtle1/pose* oraz *turtle1/cmd_vel*) na nowe nazwy.
- Odświeżaj diagram w oknie wtyczki *Node Graph* po każdym uruchomieniu (ew. zakończeniu pracy) węzła aby ocenić poprawność struktury twojego systemu ROS.
- Strzałka z nazwą kanału na diagramie będzie widoczna dopiero po uruchomieniu pary węzłów nadawca-odbiorca połączonych wspólnym kanałem informacyjnym. W razie potrzeby użyj komendy *rostopic echo …*
- Unikaj niepotrzebnych już „węzłów-duchów" – tj. zakończ działanie węzłów, których już nie potrzebujesz (Ctrl+C). Listę uruchomionych węzłów możesz zobaczyć we wtyczce *Introspection>Process Monitor* programu *rqt*. Jeżeli chcesz zakończyć pracę węzła a nie możesz odnaleźć okna terminala, w którym został uruchomiony, to użyj przycisku *Kill Node* w oknie tej wtyczki.
- Ważne: kanały zarejestrowane w serwerze nazw *Master* przez uruchomiony węzeł będą widoczne na liście kanałów (np. komendą *rostopic list*) nawet po zakończeniu pracy węzła, który zarejestrował dany kanał.



Rys. 3.

<u>Wykonaj kopię ekranu z widocznymi: diagramem twojego systemu ROS oraz dwoma oknami symulatorów żółwi po wykonaniu kilku ruchów. Plik załącz do sprawozdania.</u>

<u>UWAGA! Sprawdź czy w obrazie z kopią ekranu czytelne są nazwy kanałów w oknie z diagramem systemu ROS.</u>

Zanotuj wszystkie użyte komendy *rosrun…* podczas realizacji polecenia 12. (będą przydatne w kolejnym ćwiczeniu).

Pytania sprawdzające

- Podaj ogólną składnie polecenia *rosrun*.

AUTONOMICZNE ROBOTY TRANSPORTOWE

Ćwiczenie ART-6

Symulator Gazebo. Tworzenie mapy dostępności (algorytm SLAM)

I. Cel ćwiczenia

Celem dydaktycznym jest zapoznanie z techniką tworzenia mapy środowiska pracy robota z zastosowaniem algorytmu *Simultaneous Localization And Mapping* (*SLAM*).

Celem praktycznym ćwiczenia jest wyposażenie robota *Turtlebot* (symulowanego w *Gazebo*) w laserowy skaner odległości 360° oraz utworzenie mapy dostępności modelu 3-pokojowego mieszkania do celów nawigacji.

Ćwiczenie składa się z następujących etapów:

a) Utworzenie struktury katalogów i plików nowego pakietu ROS o nazwie *my_tb*.

b) Utworzenie skrótów dla poleceń wydawanych z linii komend (tzw. *aliasów*) dla często powtarzanych poleceń (usprawnienie pracy).

c) Utworzenie w nowym pakiecie pliku konfiguracyjnego typu *launch* uruchamiającego twoją wersję symulatora *Gazebo*.

d) Utworzenie (modyfikacja) modelu otoczenia robota – tzw. „świata” (w *Gazebo*).

e) Utworzenie (modyfikacja) konfiguracji programu *RViz* do wizualizacji pracy robota oraz jego sensorów.

f) Wyposażenie modelu robota w laserowy skaner odległości działający w zakresie 360°.

g) Utworzenie mapy dostępności modelu pomieszczeń z wykorzystaniem nowego skanera.

II. Przebieg ćwiczenia

1. Uruchom wirtualną maszynę z zainstalowanymi systemami Linux i ROS.
2. Utwórz strukturę plików nowego pakietu ROS o nazwie *my_tb* (od ang. *my_turtlebot*) w katalogu *catkin_ws/src*:

   cd ~/catkin_ws/src

   catkin_create_pkg my_tb
3. Utwórz 4 katalogi: *launch, worlds, rviz, maps* w katalogu pakietu *my_tb* (komendą *mkdir* lub w aplikacji *File Manager*).

4. Do pliku *~/.bashrc* dodaj poniższe skróty poleceń wydawane z linii komend (tzw. *aliasy*). Wpisanie w oknie terminala tekstu widocznego przed znakiem równości będzie tożsame z wykonaniem komendy po znaku równości.

   # === Skroty uruchamiajace "oryginalne" pliki launch zwiazane z robotem turtlebot ===
   # - symulator GAZEBO z robotem i otoczeniem:
   alias tb_gaz='roslaunch turtlebot_gazebo turtlebot_mw_office.launch'

   # - sterowanie robotem z klawiatury:
   alias tb_kbrd='roslaunch turtlebot_teleop keyboard_teleop.launch'

   # - algorytm SLAM do tworzenia mapy dostepnosci (gmapping)
   alias tb_gmap'=roslaunch turtlebot_gazebo gmapping_demo.launch'

   # - algortm amcl do samodzielnej nawigacji robota
   alias tb_amcl='roslaunch turtlebot_gazebo amcl_demo.launch'

   # - wizualizacja robota i pracy jego sensorow (rviz)
   alias tb_rviz='roslaunch turtlebot_rviz_launchers view_navigation.launch'

   # === Skroty uruchamiajace moje skladniki systemu ROS ===
   # - symulator GAZEBO z robotem i otoczeniem:
   alias my_tb_gaz='roslaunch my_tb my_tb_gazebo_office.launch'

   # - wizualizacja robota i pracy jego sensorow (rviz)
   alias my_tb_rviz='rosrun rviz rviz -d ~/catkin_ws/src/my_tb/rviz/my_tb_rviz_gmap.rviz'

   Przeczytaj komentarze do skrótów oraz odpowiadające im polecenia. Zauważ, że skróty uruchamiające twoje wersje składników systemu ROS mają przedrostek *my_* (w stosunku do poleceń „oryginalnych”).

   Zapisz plik na dysku i zamknij wszystkie okna terminali (zmiany w pliku *.bashrc* odniosą skutek dopiero w nowo uruchamianych terminalach).

   Wypróbuj działanie skrótów *tb_gaz* i *tb_kbrd*, po czym zakończ uruchomione procesy (Ctrl+C).

5. Uruchom serwer nazw ROS *Master* komendą *roscore*. <u>Nie przerywaj jego działania aż do zakończenia realizacji ćwiczenia.</u>

6. Utwórz nowy dokument w edytorze plików tekstowych (*Kickoff Application Launcher > Search: Kate*) i skopiuj do niego całą zawartość pliku typu *launch*:

   /opt/ros/indigo/share/turtlebot_gazebo/launch/turtlebot_mw_office.launch

   Właśnie ten plik uruchamia symulator *Gazebo* (patrz komenda do skrótu *tb_gaz*).

   Zapisz utworzony plik w katalogu *~/catkin_ws/src/my_tb/launch* pod nazwą *my_tb_gazebo_office.launch*

7. Uruchom utworzony plik typu *launch* z pakietu *my_tb* skrótem *my_tb_gaz* (patrz komenda do skrótu *my_tb_gaz*).
8. Na lewym panelu okna *Gazebo* w zakładce *World* rozwiń listę *Models*. Kliknij prawym przyciskiem myszy element *mobile_base* (model robota *turtleboot*) i z wyświetlonego menu wybierz *View>Transparent*. Przetestuj pozostałe tryby wizualizacji modelu (np. *View>Collisions* wyświetla bryły reprezentujące „uproszczone" zarysy części robota, które mogą oddziaływać przez kontakt z innymi bryłami).
9. W lewym panelu okna Gazebo kliknij zakładkę Insert. Na liście dostępnych modeli kliknij *Jersey Barrier* (betonowa bariera) i zastaw barierą otwór drzwi zewnętrznych w najmniejszym pomieszczeniu (rys. 1.).

   Podobnie ustaw na podłodze modelu mieszkania *N* pachołków (*Construction Cone*), gdzie:

   $$N = \text{część\_całkowita}(\text{cyfra\_jedności\_twojego\_albumu}/2) + 1$$

10. Kliknij prawym przyciskiem myszy element *mobile_base* i z menu wybierz *Delete* (usuń ze świata).

    Z głównego Menu okna *Gazebo* wybierz *File>Save World As* i nagraj kopię świata w katalogu *~/catkin_ws/src/my_tb/worlds* pod nazwą *my_tb_office.world*

    Wyjaśnienie: model robota (*mobile_base*) jest zdefiniowany w niezależnym pliku od pliku opisującego świat. Po załadowaniu do *Gazebo* pliku świata odpowiednia komenda w pliku *launch* umieszcza w nim robota. Dlatego przed nagraniem pliku świata należy usunąć z niego model robota. W przeciwnym wypadku definicja robota zostanie włączona do pliku opisującego świat (jako jego część), co może być przyczyną problemów w realizacji ćwiczenia.

    Zakończ działanie procesów uruchomionych skrótem *my_tb_gaz* (Ctrl+C).

11. W edytorze plików tekstowych otwórz właśnie nagrany plik z opisem świata *my_tb_office.world*.

    Wykonaj następujące zmiany w treści pliku:

    - wyłączenie symulowania cieni:

    z <shadows>1 na <shadows>0

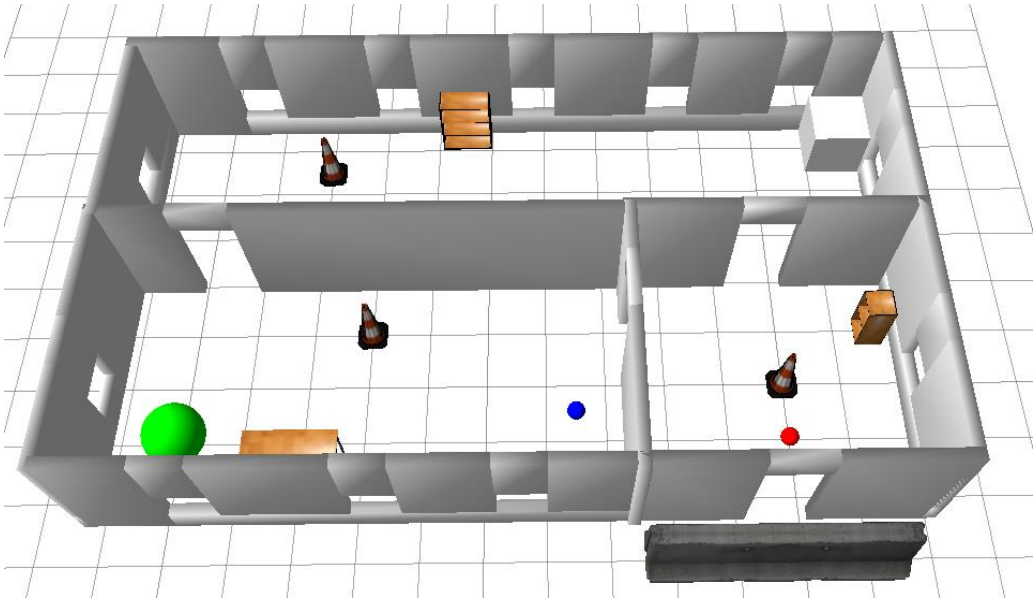
    - zwiększenie interwału czasowego symulacji zjawisk fizycznych do 0.005s (200Hz):

    z <max_step_size>0.001 na <max_step_size>0.005

    z <real_time_update_rate>1000 na <real_time_update_rate>200

    UWAGA! Iloczyn wartości parametrów *<max_step_size>* oraz *<real_time_update_rate>* musi być równy 1.

    Zapisz plik na dysku.



Rys. 1.

12. W pliku *my_tb_gazebo_office.launch* wykonaj następujące zmiany:

    z:

    <arg name="world_name" value="$(find turtlebot_gazebo)/worlds/turtlebot_mw_office.world"/>

    na:

    <arg name="world_name" value="$(find my_tb)/worlds/my_tb_office.world"/>

    W ten sposób, twój plik *launch* będzie uruchamiał symulator *Gazebo* z twoim opisem modelu świata.

    Zapisz plik na dysku.

13. Uruchom zmodyfikowany plik *my_tb_gazebo_office.launch* skrótem *my_tb_gaz*.

    Uruchom dwa kolejne pliki *launch* skrótami: *tb_gmap* oraz *tb_rviz*.

14. W lewym panelu *Displays* okna *RViz* rozwiń listę właściwości obiektu *Global Options*. Nadaj parametrowi *Frame Rate* wartość 7 (częstotliwość wizualizacji wszystkich obiektów w Hz).

    W obiekcie *LaserScan* (skan z dalmierza laserowego 2D) nadaj wartość 0.03 parametrowi *Size* (rozmiar punktów reprezentujących wyniki pomiarów).

    Wyłącz wizualizację obiektów: *Global Map* oraz *Local Map*.

15. Kliknij przycisk *Add* (dodaj obiekt do wizualizacji) u dołu panelu *Displays*. Przejdź na zakładkę *By topic* (wybór z listy zarejestrowanych kanałów). Rozwiń szczegóły listy (kanału) */camera>/rgb>/image_raw*.

Kliknij element *Image* (Obraz). Pod panelem *Displays* powinno wyświetlić się okno z podglądem obrazu z kamery robota.

Zapisz (*File>Save Config As*) aktualną konfigurację programu *RViz* pod nazwą *my_tb_rviz_gmap.rviz* w katalogu *~/catkin_ws/src/my_tb/rviz*.

Kolejnym razem uruchamiaj *RViz* z właśnie zapisaną konfiguracją skrótem *my_tb_rviz* (patrz odpowiedni alias w pliku *.bashrc*).

16. Uruchom nową aplikację Teminator'a, a w niej węzeł do sterowania robotem z klawiatury skrótem *tb_kbrd*. Zmniejsz okno Terminala i steruj ruchem robota.

    Użyty wcześniej skrót *tb_gmap* uruchomił między innymi węzeł z algorytmem *SLAM* (*Simultaneous Localization And Mapping*), który na podstawie „obrazu" z laserowego skanera odległości i informacji o drodze przebytej przez koła robota, tworzy „mapę dostępności" otoczenia (ang. *occupancy grid*).

    W trakcie ruchu robota tworzona mapa dostępności powiększa się (obracaj często robota o 360 stopni). Jednocześnie działa algorytm samolokalizacji robota na tworzonej mapie. Pozycja i orientacja robota względem mapy w oknie *RViz* są wielkościami estymowanymi, niekoniecznie zgodnymi z pozycją i orientacją rzeczywistą (w symulatorze *Gazebo*).

    Zauważ, że zakres kąta pomiaru skanera laserowego to zaledwie ok. 60 stopni (tyle samo co szerokość pola widzenia kamery stereoskopowej *Microsoft Kinect* zainstalowanej na robocie). Dlatego algorytm SLAM nie jest szczególnie efektywny (estymowana pozycja robota na mapie zmienia się skokowo, a tworzona mapa dostępności nie jest w pełni zgodna z układem ścian i przedmiotów w symulatorze Gazebo).

    Rozwiązaniem problemu powinno być zastosowanie laserowego skanera odległości o kącie pomiaru 360 stopni. W dalszej części wyposażysz symulowanego robota w taki sensor.

17. Zakończ działanie procesów uruchomionych skrótami: *my_tb_rviz, tb_gmap, my_tb_gaz*.

    W edytorze plików tekstowych otwórz plik *my_tb_gazebo_office.launch* w twoim pakiecie *my_tb*. Poniższy fragment pliku:

```
<!-- Fake laser -->
  <node pkg="nodelet" type="nodelet" name="laserscan_nodelet_manager" args="manager"/>
  <node pkg="nodelet" type="nodelet" name="depthimage_to_laserscan"
      args="load depthimage_to_laserscan/DepthImageToLaserScanNodelet
laserscan_nodelet_manager">
    <param name="scan_height" value="10"/>
    <param name="output_frame_id" value="/camera_depth_frame"/>
    <param name="range_min" value="0.45"/>
    <remap from="image" to="/camera/depth/image_raw"/>
    <remap from="scan" to="/scan"/>
  </node>
```

    tworzy m.in. węzeł, który na podstawie danych z kamery 3D (*Microsoft Kinect*) na robocie publikuje do

kanału o nazwie */scan* wyniki pomiarów odległości od obiektów tylko w jednej płaszczyźnie - równoległej do podłoża. Stąd nazwa w komentarzu *Fake laser* (imitacja skanera laserowego).

Usuń z pliku powyższy fragment (imitacja skanera nie będzie już potrzebna).

Dodaj pod wierszem *<launch>* (na początku pliku) linię (objaśnienie w dalszej części):

<remap from="/scan_2D" to="/scan"/>

Zapisz plik na dysku.

18. Definicja sensorów robota znajduje się w pliku:

    /opt/ros/indigo/share/turtlebot_description/urdf/turtlebot_gazebo.urdf.xacro

    Jednak aby modyfikować zawartość tego pliku (jego właścicielem jest administrator *root*), musisz zmienić jego właściciela na *user*, który jest zwykłym użytkownikiem. W tym celu wykonaj komendy:

    cd /opt/ros/indigo/share/turtlebot_description/urdf

    ls -l turtlebot_gazebo.urdf.xacro (wyświetlona zostanie informacja m.in. o właścicielu pliku)

    sudo chown user turtlebot_gazebo.urdf.xacro (zmiana właściciela pliku na user)

    ls -l turtlebot_gazebo.urdf.xacro (weryfikacja wyniku zmiany właściciela pliku)

    Utwórz kopię zapasową tego pliku w katalogu *~/Kopie*.

19. W edytorze plików tekstowych otwórz plik *turtlebot_gazebo.urdf.xacro* z katalogu

    /opt/ros/indigo/share/turtlebot_description/urdf

    W pliku znajdują się parametry symulowanego w *Gazebo* sensora odległości 3D *Microsoft Kinect* (który posiada także wbudowaną tradycyjną kamerę).

    Zmień liczbę wykonywanych klatek w ciągu sekundy przez symulowaną kamerę na 5/s (zmiana przyspieszy działanie *Gazebo*):

    <update_rate>5.0</update_rate>

    Tuż przed linią *</xacro:macro>* pod koniec pliku wklej poniższy fragment tekstu, który będzie uruchamiał dodatkowo symulację laserowego skanera odległości o zakresie 360 stopni.

    UWAGA! Nadaj prawidłowy przedrostek *xy_* nazwie kanału w poniższym fragmencie tekstu (linia: *<frameName>xy_laser_scanner</frameName>*), gdzie przedrostek *xy_* to pierwsze litery twojego imienia i nazwiska.

```
<gazebo reference="plate_bottom_link">
  <sensor type="gpu_ray" name="laser_scanner_2D">
    <pose>0 0 0.02 0 0 0</pose>
    <visualize>false</visualize>
    <update_rate>15</update_rate>
    <ray>
      <scan>
        <horizontal>
          <samples>360</samples>
          <resolution>1</resolution>
          <min_angle>-3.14</min_angle>
          <max_angle>3.14</max_angle>
        </horizontal>
      </scan>
      <range>
        <min>0.2</min>
        <max>30.0</max>
        <resolution>0.01</resolution>
      </range>
      <noise>
        <type>gaussian</type>
        <mean>0.0</mean>
        <stddev>0.01</stddev>
      </noise>
    </ray>
    <plugin name="gazebo_ros_head_hokuyo_controller" filename="libgazebo_ros_gpu_laser.so">
      <topicName>/scan_2D</topicName>
      <frameName>xy_laser_scanner</frameName>
    </plugin>
  </sensor>
</gazebo>
```

Przeanalizuj wartości parametrów nowego sensora. Zauważ, że skaner będzie umieszczony 0.02m nad dolną płytą montażową robota (*plate_bottom_link*). Wyniki pomiarów będą publikowane do kanału */scan_2D*, a układem odniesienia dla pomiarów będzie nowy (wymagający zdefiniowania) układ o nazwie *xy_laser_scanner*.

Linia *<remap from="/scan_2D" to="/scan"/>*, która została dodana wcześniej do pliku *my_tb_gazebo_office.launch*, spowoduje zamianę nazwy kanału */scan_2D* na */scan*, tj. na identyczną, jakiej używał poprzednio symulowany skaner (*Fake laser*). Dzięki temu zabiegowi będzie możliwe poprawne działanie obu wersji robota (z *Fake laser* oraz z nowym skanerem 360 stopni).

Zapisz plik na dysku.

Szczegółowy opis użycia wirtualnych sensorów w Gazebo znajduje się pod adresem: http://gazebosim.org/tutorials?tut=ros_gzplugins

20. Dodaj do pliku *my_tb_gazebo_office.launch* w twoim pakiecie *my_tb* fragment uruchamiający węzeł publikujący relację między nowym układem odniesienia *xy_laser_scanner* (związanym z nowym sensorem) a układem współrzędnych *plate_bottom_link* (związanym ze środkiem dolnej płyty montażowej):

```
<node pkg="tf" type="static_transform_publisher" name="laser_to_plate_publisher"
        args="0 0 0.02 0 0 0 plate_bottom_link xy_laser_scanner 100"  >
 </node>
```

Zapisz plik na dysku.

Celem przypomnienia: w poprzednich ćwiczeniach uruchamiane były podobne węzły ale z linii komend:

rosrun tf static_transform_publisher x y z yaw pitch roll frame_id child_frame_id period_in_ms

W tym przypadku równoważna komenda do powyższego elementu *<node>…</node>* wyglądałaby następująco:

rosrun tf static_transform_publisher 0 0 0.2 0 0 0 plate_bottom_link xy_laser_scanner 100

21. Uruchom symulator *Gazebo* skrótem *my_tb_gaz*.

Uruchom program diagnostyczny *rqt*, a w nim wtyczkę *Visualization>TF Tree*.

Powiększ fragment z widoczną relacją między układami *xy_laser_scanner* a *plate_bottom_link*.

Wykonaj kopię ekranu do sprawozdania.

22. Uruchom kolejne składniki swojego systemu ROS skrótami: *tb_gmap, my_tb_rviz*.

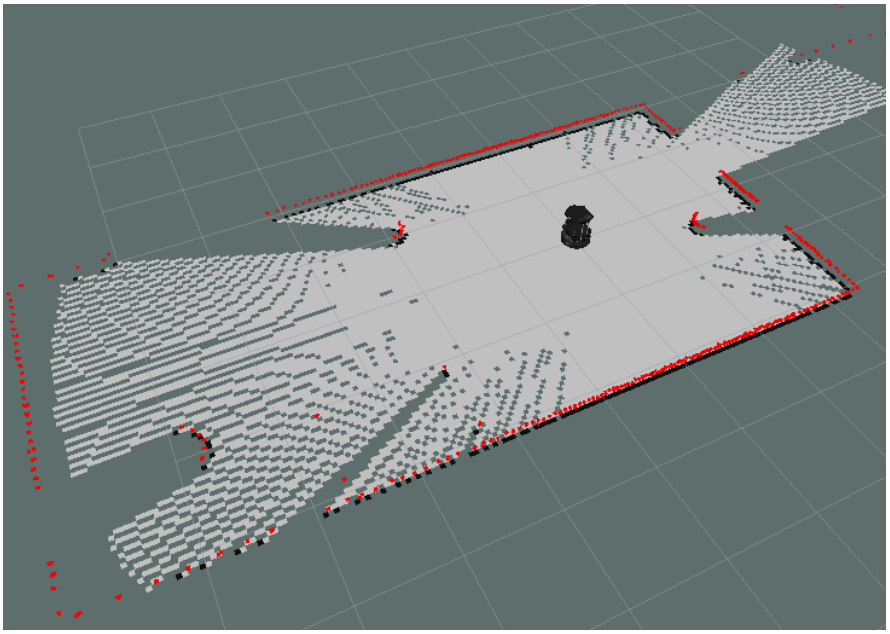
W oknie *RViz* widoczne będą wyniki pomiarów odległości z nowego skanera laserowego (obiekt *LaserScan*) zgodne z „zarysem” ścian i obiektów wokół robota (rys. 2.)

23. Użyj klawiatury do przemieszczania robota po całym modelu mieszkaniu w celu utworzenia kompletnej mapy dostępności. Kolory pikseli oznaczają:

- jasnoszary – obszar dozwolony dla ruchu robota (wolna przestrzeń),
- czarny – obszar zabroniony (pewna kolizja),
- ciemnoszary – obszar nieznany (algorytm planowania trasy będzie go traktował z reguły jako obszar zabroniony).

Postaraj się, aby twoja mapa miała jak najmniej obszarów nieznanych (ciemnoszarych).

W przypadku, gdy tworzona mapa znacznie odbiega od rzeczywistego układu pomieszczeń, przerwij mapowanie i rozpocznij od początku (zrestartuj węzły) i podążaj przez mieszkanie inną trasą.



Rys. 2.

24. Po ukończeniu mapowania (NIE KOŃCZ JESZCZE PRACY SYSTEMU ROS!) wykonaj komendy:
    cd ~/catkin_ws/src/my_tb/maps (zmiana bieżącego katalogu)
    rosrun map_server map_saver -f my_map
    Druga komenda utworzy w katalogu bieżącym dwa pliki:
    my_map.yaml – z informacjami o podstawowych parametrach mapy dostępności,
    my_map.pgm – plik graficzny (bitmapa) z właściwą mapą dostępności.
    Otwórz oba pliki i przeanalizuj ich zawartość.
25. Wykonaj kopię ekranu z widocznymi oknami *Gazebo* oraz *RViz* z ukończoną mapą dostępności.

Pytania sprawdzające

- Rozwiń skrót algorytmu *SLAM* i objaśnij jego przeznaczenie.