# Wprowadzenie do autonomicznej nawigacji robotami mobilnymi w środowisku ROS

Instrukcja do ćwiczenia

Michał Drwięga

15 marca 2016

## 1 Wstęp

Celem ćwiczenia jest zapoznanie się ze środowiskiem programistycznym *ROS* (*Robot Operating System*), jego strukturą oraz możliwościami. Dodatkowo, przedstawiony zostanie sposób wykorzystania symulatora *Stage* z poziomu *ROS'a* w wersji *Hydro*.

## 2 Przygotowanie do ćwiczenia

Aby rozpocząć ćwiczenie należy uruchomić system Ubuntu oraz pobrać katalog z repozytorium SVN (adres: svn://diablo.ict.pwr.wroc.pl/roboty_spoleczne/ros, użytkownik: ros, hasło: ros), np. za pomocą poniższego polecenia.

```
svn co –username ros –password ros svn://diablo.ict.pwr.wroc.pl/roboty_spoleczne/ros
```

W katalogu tym oprócz kopii niniejszej instrukcji znajduje się wstępnie skonfigurowana paczka środowiska ROS o nazwie: pioneer_p3dx_demo .

## 3 Utworzenie nowego workspace'a w ROS'ie

Przed założeniem workspace'a należy załadować zmienne środowiskowe ROS'a poleceniem:

```
source /opt/ros/hydro/setup.bash .
```

Następnie, w katalogu domowym należy utworzyć katalog proj_zesp_ws/src oraz wygenerować plik *CMakeLists.txt*, wykorzystując poniższe komendy.

```
mkdir -p ∼/proj_zesp_ws/src
cd ∼/rob_spol_ws/src
catkin_init_workspace
```

W kolejnym kroku workspace zostanie zbudowany, co spowoduje utworzenie struktury katalogów oraz plików konfiguracyjnych.

```
cd ∼/proj_zesp_ws/
catkin_make
```

Wystarczy teraz załadować zmienne środowiskowe workspace'a, który będzie po tym gotowy.

```
source devel/setup.bash
```

Warto zaznaczyć, że po otworzeniu nowego terminala należy w nim również ustawić zmienne środowiskowe, wykonując powyższe polecenie z używanego workspace'a.

Można również zautomatyzować ustawianie zmiennych środowiska w kolejnych terminalach, przez dodanie odpowiedniego wpisu do pliku ~/.bashrc.

# 4 Opis dostarczonego pakietu i sposobu kompilacji

Poniżej przedstawiona została struktura katalogów oraz zawartość dostarczonego pakietu.

**pioneer3dx_demo/**

- **package.xml** – podstawowe informacje o pakiecie,
- **CMakeLists.txt** – plik konfiguracyjny systemu budowania pakietów*CMake*,
- **launch/** – pliki uruchomieniowe,
  - **pioneer3dx_urdf.launch** - uruchomienie węzła publikującego dane o robocie P3-DX na podstawie pliku p3dx.urdf.
  - **stage_p3dx_sick.launch** - uruchomienie symulatora Stage z modelem robota oraz skanerem laserowym Hokuyo.
  - **stage-worlds/** - pliki symulatora *Stage*.
  - **p3dx.rviz** - konfiguracja narzędzia *rviz*.
- **scripts/** – skrypty w Pythonie,
  - **key_teleop.py** - skrypt umożliwiający sterowanie robotem za pomocą klawiatury.

Katalog z pakietem należy przenieść do utworzonego workspace'a, do katalogu src. Następnie należy skompilować pakiety workspace'a, np. za pomocą poniższego polecenia.

```
cd ~/proj_zesp_ws && catkin_make && rosdep install pioneer3dx_demo
```

W przypadku, gdy w workspace'ie znajduje się więcej pakietów, w celu skompilowania wybranego można skorzystać z parametru - -pkg. Polecenie ma następującą postać.

```
catkin_make  - -pkg pioneer3dx_demo
```

# 5 Uruchomienie symulacji w środowisku *Stage* z poziomu *ROS'a*

W celu uruchomienia symulacji można wykorzystać dostarczony plik stage_pioneer3dx_sick.launch oraz narzędzie *ROS'a* do uruchamiania plików tego typu, czyli roslaunch (http://wiki.ros.org/roslaunch).

```
roslaunch pioneer3dx_demo stage_p3dx_hokuyo.launch
```

Wykonanie powyższego polecenia powinno spowodować pojawienie się okna symulatora z planszą i modelem robota *Pioneer P3-DX*.

Aby sterować robotem za pomocą klawiatury można wykorzystać dołączony skrypt w pythonie key_teleop.py. W tym celu w nowym terminalu należy wywołać poniższe polecenie.

```
python ~/proj_zesp_ws/scripts/key_teleop.py
```

Można wykorzystać również dostarczony program wall_follower, który spowoduje autonomiczne poruszanie się robota wzdłuż ścian.

```
rosrun pioneer3dx_demo wall_follower
```

Uruchomienie algorytmu *SLAM* w środowisku *ROS* wraz z wizualizacją uzyskiwanych wyników można przeprowadzić w poniższy sposób.

```
rosrun gmapping slam_gmapping scan:=base_scan _particles:=100 _linearUpdate:=0.1 _angularUpdate:=0.03
rosrun rviz rviz -d $(rospack find pioneer3dx_demo)/launch/p3dx.rviz
```

Do wizualizacji mapy uzyskiwanej metodą SLAM wykorzystano narzędzie *rviz*. Umożliwia ono również wizualizację wielu innych danych, między innymi wyników odometrii i odczytów skanera laserowego. Rodzaj wyświetlanych danych można wybrać z poziomu uruchomionego okna.

# 6 Zadanie do wykonania

W ramach ćwiczenia należy uruchomić symulację oraz zapoznać się z działaniem systemu. Do analizy działania uruchomionego oprogramowania na platformie *ROS* mogą być przydatne poniższe polecenia.

`rostopic list` – wyświetla listę uruchomionych tematów, http://wiki.ros.org/rostopic

`rosnode list` – wyświetla listę uruchomionych węzłów, http://wiki.ros.org/rosnode

`rosrun rqt_graph rqt_graph` – wyświetla graf, na którym znajdują się uruchomione węzły, tematy oraz połączenia między nimi, http://wiki.ros.org/rqt_graph

# Wykorzystane materiały

[1] ROS Tutorials. http://wiki.ros.org