

# Politechnika Łódzka Instytut Automatyki

# Laboratorium Robotów Usługowych

Instrukcja 3 Processing - ROS

# Wstęp

Celem instrukcji jest połączenie utworzonego interfejsu dla robota z wykorzystaniem języka Processing z systemem ROS. Zapoznać się z załączonym do instrukcji przykładem *ros\_processing*. Przed uruchomieniem przykładu należy uruchomić symulację z poprzednich zajęć. W terminalu nalezy wpisać:

roslaunch pkg\_rob\_usl start\_sim.launch

## Zadania

#### Zadanie na 3

Rozbudowa interfejsu (instrukcja 1 zadanie na 3) do obsługi 1 robota o możliwość wysłania po ROS w zależności od wybranego robota na topicu /nazwa\_robota/move\_base\_simple/goal współrzędnych punktu (co najmniej 1 z 3), do którego dojechać ma robot.

### Zadanie na 4

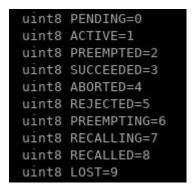
Rozbudowa interfejsu (instrukcja 1 zadanie na 4) o możliwość wysłania do 1 z 3 robotów po ROS informacji o celu, do którego ma dojechać robot oraz podłączenie do ROS kontrolek sterujących prędkościami postępowymi i obrotowymi dla robota. Prędkości należy wysłać na topicu /nazwa\_robota/cmd\_vel. Prędkość postępową należy wysłać w wiadomości w polu linear.x, a obrotową na angular.z.

#### 7adanie na 5

Do tego zadania należy uruchomić dodatkowo skrypt "task\_result" dołączony do instrukcji 3. Biblioteka ROSprocessing obsługuje tylko niektóre typy wiadomości ROS, dlatego do przekazywania informacji o statusach będzie wykorzystywana wiadomość typu Vector3 z biblioteki geometry\_msgs.

Należy rozbudować interfejs (instrukcja 1 zadanie na 5) o kontrolki odbierające informację na topicu **/robots\_task\_status**. W polach x, y, z kolejno przekazane są statusy po dojechaniu do punktu dla robotów tb3\_0, tb3\_1, tb3\_2. Należy wyświetlać odebrany status w zależności od stanu robota.

Znaczenie statusów:



Do skonwertowania odebranej wiadomości i wyświetlenia tekstu po stronie interfejsu można użyć funkcji:

**Double.toString**(zmienna)