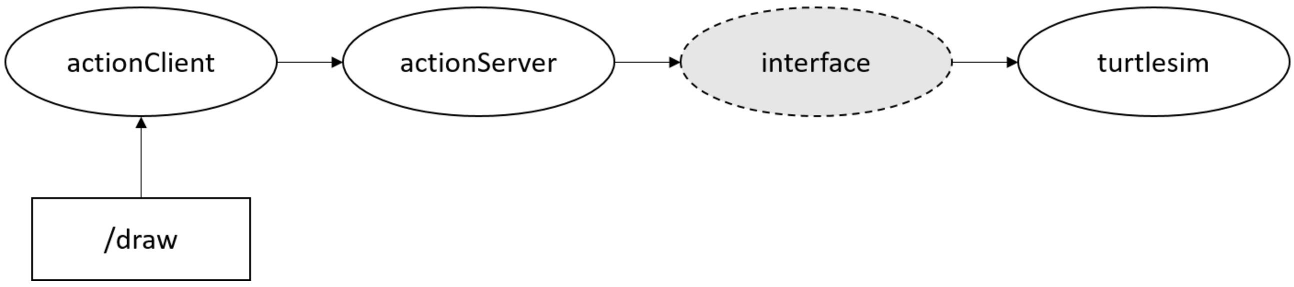
Robotics Homework 2016/2017 – Part B.01

Gianluca Bardaro & Matteo Matteucci

Exercise n°3 – actionlib

The aim of this exercise is to test your ability in implementing a simple interaction between an actionlib client and an actionlib server. What you need to implement is the following:

- a simple action client which reads a topic and uses the information provided to trigger an action
- a simple action server which manages the action and provides feedback and result
- a ROS interface between the action server and the robot



The Robot

For this exercise, you will have to use as your robotic platform the ROS turtlesim (http://wiki.ros.org/turtlesim). You are not allowed to modify the node in any way, any extra feature needed has to be implemented outside it.

The action

Final goal of the turtlesim is to draw a regular polygon on the screen (see picture below). How you are going to do that is up to you, but the action has the following structure:

- **goal**: length of the sides of the figure and number of the sides
- **feedback**: current number of sides already drawn
- **result**: boolean value indicating if the drawing was successful or not

A drawing is unsuccessful if the robot hits the border of the world when trying to draw a figure.

The action client

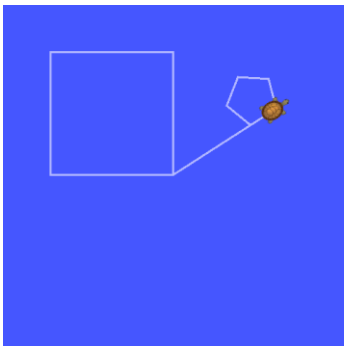
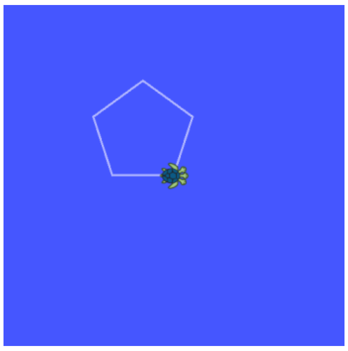
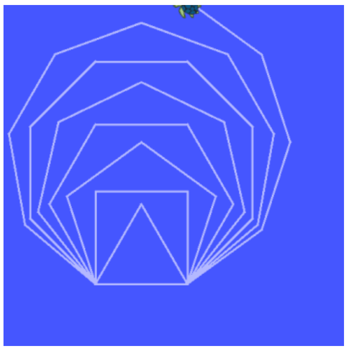
The action client receives the goal from a topic called `/draw` using the following message:

```
draw_figures_msgs/Draw.msg
uint32 sides  #number of the sides
uint32 length #length of each side
```

You have to create the package containing this message in your workspace, but **you do not need to include** it in your solution. Any included message will be ignored, and we will use our own client to test your actionlib exercise.

The interface

It is up to you how to implement the interface between the actionServer and the turtlesim. The method used to draw is irrelevant. The expected result is shown in the following pictures.

Exercise n°4 – ROS Navigation

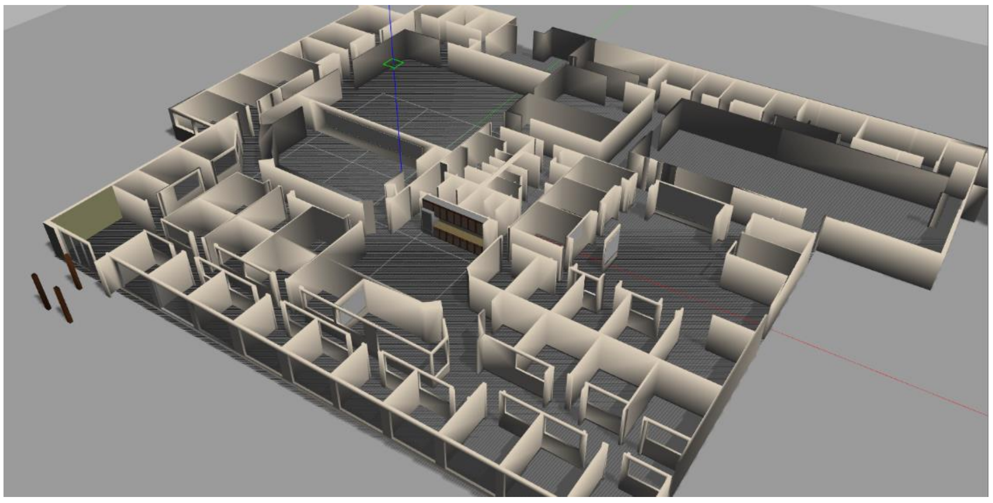
In this exercise, you need to prove your ability **in configuring and using** ROS Navigation (http://wiki.ros.org/navigation). First, you need to create a map of an environment using gmapping and later configure ROS Navigation to navigate the same environment autonomously, by sending goals using rviz graphical interface.

The environment

The reference environment is the willow garage world shipped with Gazebo, you can run it using the following command:

```
roslaunch gazebo_ros willowgarage_world.launch
```

This is the environment you need to map, and the one you will use for your final navigation trial. I suggest using a simpler environment when testing your robot.



The robot

The robot used for navigation is attached to the project. It is a simple differential drive robot equipped with a laser. You are not allowed to modify the robot in any way. Your configuration will be tested on the original robot, and any model provide by the studets will be ignored.

Resources

Here you will find the details on how to create a map:

http://wiki.ros.org/slam_gmapping/Tutorials/MappingFromLoggedData

Here a guideline on how to configure ROS Navigation:

http://wiki.ros.org/navigation/Tutorials/RobotSetup

The expected output is the set of configuration files (e.g., map, launch files, etc.) used to run a navigation experiment.

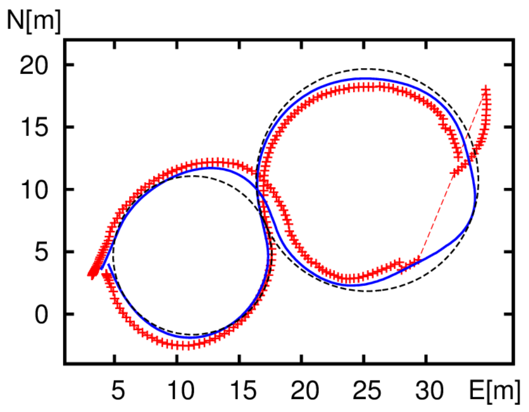
Exercise n°5 – ROS/Gazebo plugin

In this exercise, you need to implement a simple Gazebo plugin that is able to publish the measurement of a gazbo sensor on a ROS topic.

The sensor

You need to simulate a GPS with the following behaviors (please note that the GPS already exists, what you need to do is to publich it on ROS with the following characteristics):

- The frequency of the signal (topic) is 1 Hz
- Loss of signal: with a low probability (parameter) the GPS may stop working for a variable amount of second (i.e. between 5 and 10 seconds)
- multipath: with a really low probability (parameter) the measurements are translate by a random amount for a variable amount of second. See picture for a possible reference taken from a real GPS.



The topic

The measurements have to be published on a topic named `/fix` using a `sensor_msgs/NavSatFix` (http://docs.ros.org/kinetic/api/sensor_msgs/html/msg/NavSatFix.html) message.