General Remarks - Addendum

Using custom messages and services, remember to:

- add the necessary headers in nodes using custom messages and services
- in the CMakeLists.txt and package.xml of packages contaning nodes using custom messages and/or services, add the dependency from the package where custom messages are defined *(es. turtlebot controller)*
- In the CmakeLists.txt of packages contaning nodes using custom messages and/or services, add *(es. turtlebot controller)*:

  add_dependencies(<node_name> ${catkin_EXPORTED_TARGETS})

- If a node using custom messages and/or services is built in the same package where custom messages and services are defined, in the CMakeLists.txt of the package add *(es. my_srv):*

  add_dependencies(<node_name> ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})

General Remarks - Addendum

As a general remark, before submitting your code, please be sure that the packages will be built without errors.

How can you do that?

Practical hint: - remove the directory build and devel of your ros workspace.

- build again the workspace (catkin_make)

If it works without any error, everything is fine. If not, something should be fixed in the CMakeLists.txt of your packages.

Research Track I – First assignment

Carmine Tommaso Recchiuto, Phd

First assignment

- Download the assignment by running:
  - git clone https://github.com/CarmineD8/assignment1.git in the src folder of your ROS workspace
  - catkin_make in the root folder of your ROS workspace
  - rospack profile

- The assignment requires controlling a holonomic robot in a 2d space with a simple 2d simulator, Stage. The simulator can be launched by executing the command:

  rosrun stage_ros stageros $(rospack find assignment1)/world/exercise.world

Only if you are not using the Docker image and you do not have stage_ros installed, you can install it with: sudo apt-get install ros-<your_ros_version>-stage-ros

Expected Behaviour

1. The robot asks for a random target, with both coordinates in the interval (-6.0, 6.0).

2. The robot reaches the target.

3. Go to step 1.

Requirements

- A new ROS packages should be created.
- Two processes (ROS nodes) should be developed.
- You may use cpp or python for writing your code.
- The first process will be in charge of:
  - calling a service for receiving a random target;
  - making the robot reach the target.
- The second process will act as a Service Server, by replying to the client with a random target, having x and y in the interval (-6.0, 6.0).
- A target is considered reached when the distance between the robot and the target is below 0.1.

Requirements – how to submit the assignment

- The link of a github repository containing the developed ROS package should be given;

- The repo should have a README.md file with:
  - Description of the content of the package (nodes, custom messages or services (if any)).
  - Computational graph of the system (how do nodes communicate? You may use rqt_graph).
  - Instructions about how to run the code.

- Functions and source files should be documented (optional: you can create a docs folder with DoxyGen documentation).

Hints

- The first node should implement a ROS publisher (cmd_vel, for setting the robot speed), a ROS subscriber (odom, for knowing the actual robot position) and a ROS client (for receiving the random target)

- The second node should implement a ROS server (for setting the random target)

- The position of the robot is given in the topic **odom**, by using a **nav_msgs/Odometry** (Odometry message defined in the package nav_msgs) message. This means that the x and y position of the robot may be retrieved by reading the pose.pose.position.x and pose.pose.position.y fields of the message received by the callback associated with the subscriber.

- For the robot control, at first, check if the position has been reached. If yes, you should call the service to retrieve the new target position. If not, you should set **cmd_vel** (a **geometry_msgs/Twist** message) depending on the difference between the target (xt) and the robot position (x) (e.g., vel_x = k* (xt - x))

Deadline

A soft deadline is set to the 25th November and the next Research Track class will be on November 26th. This means that:

- You can actually submit your assignment at any time, however 10 days before the oral discussion. However I encourage you to finish the assignment before 25th November.

- Indeed, I strongly advise you to start working on the assignment in the next weeks. In the last 9 hours of the course, we are going to work on more complex robotic simulations, which require a good knowledge of ROS. Doing the assignment will be a good way to understand what it is still unclear and catch up with the course.

- In the next two weeks, you can contact me even outside class hours for clarifications about the assignment and the course subjects in general.

- No bonus/malus will be associated with the respect of the deadline