

Project Intelligent Robotics

Assignment #4

Task 4.1 Implementing Interfaces:

In addition to ROS *Topic* interfaces, which you have both published to and subscribed on in the previous assignments, you can also implement ROS *Services* and *Actions*.

You already called services, e.g., to reset the turtle's pose in turtlesim, but implementing a service server yourself quite simple too. You can find tutorials here:

C++: <http://wiki.ros.org/ROS/Tutorials/WritingServiceClient> (python)

Python: <http://wiki.ros.org/ROS/Tutorials/WritingServiceClient> (c++)

When you planned and executed robot motions through MoveIt, you already used Actions too, even if you only accessed MoveIt's helper classes that abstract away the full action interface.

C++: http://wiki.ros.org/actionlib_tutorials/Tutorials/SimpleActionServer (ExecuteCallbackMethod)

Py: <https://www.theconstructsim.com/ros-5-mins-033-create-ros-action-server/>

Look at the respective tutorials and understand how to implement either communication interface.

Task 4.2 Count collaboratively: The main goal of this task is to foster successful group interaction and discussion. The supervisors will not get involved unless asked for assistance.

Implement a software system to count from 0 to 20.

- You have to use some ROS interface(s) to achieve this behavior.
- The system must include code written by all team members.
Push your individual components to the tutorial repository.
- Nobody is allowed to edit code of other participants. Communicate instead.
- At the end of this session, everyone should be able to run the full system on their own computer.

4.2.1: Gather as a group and work on a concept to reach this goal. Plan your communication.

4.2.2: Define common interfaces yourself for everyone to use, e.g., a new *Message type*, *Service* or *Action definition*.

4.2.3: To simplify startup, create a roslaunch file where all components can be started together. You can find the relevant documentation here: <http://wiki.ros.org/roslaunch>