06. Roslaunch, ROS paraméter szerver, Rosbag

Elmélet

Roslaunch

- Launch multiple nodes
- Also launches ROS master if not running
- Set parameters
- XML file format, `.launch` extension

Example launch file

```
    <!-- dvrk_server.launch -->
    <!-- Launch the irob dVRK high-level robot controller. After start, it will wait for
irob_msgs/Robot actions -->

    <launch>
      <group ns="saf">

        <arg name="arm_typ" default="PSM2"/>
        <arg name="arm_name" default="arm_1"/>
        <arg name="camera_registration_file" default="registration_psm1.yaml"/>

        <arg name="instrument_info_file" default="prograsp_forceps.yaml"/>

        <include file="$(find irob_robot)/config/dvrk_topic_names.xml" />

        <node name="robot_server_$(arg arm_typ)" pkg="irob_robot"
type="robot_server_dvrk"
                                                output="screen">

          <param name="arm_typ" type="string" value="$(arg arm_typ)" />
          <param name="arm_name" type="string" value="$(arg arm_name)" />
          <param name="home_joint_angles" type="yaml" value="[0.0, 0.0, 0.0, 0.0, 0.0,
0.0]" />

          <rosparam command="load"
                file="$(find irob_robot)/config/$(arg camera_registration_file)"/>

          <rosparam command="load"
                file="$(find irob_robot)/config/$(arg instrument_info_file)"/>

        </node>
      </group>
    </launch>
```

Usage

```
    roslaunch package_name file.launch
    roslaunch irob_robot dvrk_server.launch arm_typ:=PSM1
```

ROS Parameter Server

- Nodes can store and retrieve parameters at runtime
- Shared dictionary
- Best use for configuration
- ROS naming convention
- Private parameters (~)

- Available data types:
  - 32-bit integers
  - booleans
  - strings
  - doubles
  - iso8601 dates
  - lists
  - base64-encoded binary data
- Useful command: `rosparam`

Python API

```
# Call AFTER rospy.init_node()

# Getting parameters
global_name = rospy.get_param("/global_name")
relative_name = rospy.get_param("relative_name")
private_param = rospy.get_param('~private_name')
default_param = rospy.get_param('default_param', 'default_value')

# fetch a group (dictionary) of parameters
gains = rospy.get_param('gains')
p, i, d = gains['p'], gains['i'], gains['d']

# Setting parameters
# Using rospy and raw python objects
rospy.set_param('a_string', 'baz')
rospy.set_param('~private_int', 2)
rospy.set_param('list_of_floats', [1., 2., 3., 4.])
rospy.set_param('bool_True', True)
rospy.set_param('gains', {'p': 1, 'i': 2, 'd': 3})

# Using rosparam and yaml strings
rosparam.set_param('a_string', 'baz')
rosparam.set_param('~private_int', '2')
rosparam.set_param('list_of_floats', "[1., 2., 3., 4.]")
rosparam.set_param('bool_True', "true")
rosparam.set_param('gains', "{'p': 1, 'i': 2, 'd': 3}")

rospy.get_param('gains/p') #should return 1
```

Roslaunch API

```
<param name="arm_typ" type="string" value="ECM" />
<param name="publish_frequency" type="double" value="10.0" />
<rosparam command="load" file="FILENAME" />
```

YAML

- “A human friendly data serialization standard for all programming languages”

```
# registration_identity.yaml
t: [0.0, 0.0, 0.0]
R: [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]
```

Rosbag

- Record and playback ROS topics
- Command line tool
- API for C++ and Python

```
rosbag record <topic_name>
rosbag record --all
rosbag play <filename.bag>
```

Gyakorlat

Figyelem!

Az óra végén a forráskódokat mindenkinek fel kell tölteni Moodle-re egy zip archívumba csomagolva!

1: Marker: Körlap

1. Hozzuk létre a szokásos helyen a `dummy_cylinder.py` fájlt. Publikáljunk egy lapos, henger alakú markert (0.05, 0.05, -0.15) pozícióval és 0.1 m sugárral.

2: Launchfile és paraméterek a markerekhez

1. Hozzunk létre fájlt `dummy_markers.launch` névvel a `~catkin_ws/src/ros_course/launch` mappában. Írjunk launchfájlt, amely mind a két dummy marker publisher-t elindítja.

2. Módosítsuk a launchfájlt és a Python szkripteket úgy, hogy a dummy marker publisher-ek a marker pozícióját ROS paraméterként kapják meg, mely a roslaunch parancssori argumentumaként is módosítható. A markerek pozíciójának legyen default értéke is, gömb: (-0.05, 0.1, -0.12), körlap: (0.05, 0.05, -0.15).

3. Hozzunk létre YAML fájlt, amelyből a körlap marker mérete és színe kerül beolvasásra.

3: Navigáció a körlap pereme mentén

1. Hozzunk létre launchfájlt `psm_grasp.launch` névvel a `psm_grasp.py` szkripthez. A dt, sebesség és a pofák szögsebessége ROS paraméterként legyen állítható.

2. Futtassuk a `psm_grasp.launch`-ot különböző marker pozíciók mellett.

3. Módosítsuk a node-ot úgy, hogy a gömb marker megragadása előtt navigáljon körbe a korong alakú marker peremén.

4: Mentés rosbag-be

1. Az előző feladatban implementált program futása közben rögzítsük a topic-ok tartalmát egy rosbag fájlba.

```
rosbag record --all
```

2. Telepítsük az `rqt` csomagot.

```
sudo apt-get update
sudo apt-get install ros-noetic-rqt
sudo apt-get install ros-noetic-rqt-common-plugins
```

3. Játsszuk vissza a rosbag fájlt és jelenítsük meg a PSM végpontjának koordinátáit `rqt_plot` segítségével.

```
rosbag play <filename.bag>
rostopic echo /PSM1/measured_cp
```

Hasznos linkek

- Roslaunch

- ROS Parameter Server
- Python API for the ROS Parameter Server
- tag in roslaunch
- Rosparam YAML
- Rosbag
- rqt_plot