2nd Test of ROS Package by FZI (2020/10/08, Yamaguchi)

The **second trial** of **Universal_Robots_ROS_Driver** developed in collaboration between Universal Robots and the FZI Research Center for Information Technology.

In the first trial, the driver did not support the velocity control, and it was complicated to use velocity control with that driver. Thus, we gave up on using the driver.
Now, the velocity control seems to be implemented, so we try this driver again.

Setup (PC & UR pendant)

Install ROS driver

$ sudo apt-get -f install ros-kinetic-industrial-robot-status-interface
$ cd ~/catkin_ws/src/
$ git clone https://github.com/UniversalRobots/Universal_Robots_ROS_Driver.git
$ git clone -b calibration_devel https://github.com/fmauch/universal_robot.git
(If an older version of Universal_Robots_ROS_Driver or ur_modern_driver is already installed, move them to somewhere else, and remove catkin_ws/{build,devel})
$ cd ..
$ catkin_make

Setting up a UR robot for ur_robot_driver

1. Download externalcontrol-1.0.4.urcap
2. CB: setup a CB3 robot, e-series: setup an e-Series robot.
    a. Install the URCaps file (external control).
    b. Change the host IP.
        i. Welcome screen -> Program Robot -> Installation tab -> External Control
        ii. Host IP: IP of a PC (e.g. 192.168.1.41)
    c. Create a new program and insert the External Control program node into the program tree.
    d. Save as [**External_Control.urp**]
3. tool communication on an e-Series robot: tool communication setup guide.

Prepare the ROS PC

1. Extract calibration information
    $ roslaunch ur_calibration calibration_correction.launch robot_ip:=**ur3ea** target_filename:="${HOME}/**ur3ea_calibration**.yaml"

Test (Get State, Joint Trajectory Control)

Test run:
(On pendant)

1. Power on the robot.
2. Local Control/Remote Control selection: Local.
3. Execute [**External_Control.urp**] by opening and pressing the play button.
    a. Before this, launching the ROS driver is necessary on PC (see below).

(On PC)
($ roslaunch ur_robot_driver <robot_type>_bringup.launch robot_ip:=<robot_ip>)
$ roslaunch ur_robot_driver **ur3e**_bringup.launch robot_ip:=**ur3ea**
($ roslaunch ur_robot_driver **ur3e**_bringup.launch robot_ip:=**ur3ea** kinematics_config:="${HOME}/**ur3ea_calibration**.yaml")
$ rviz
Worked.

Passing the calibration data:
$ roslaunch ur_robot_driver <robot_type>_bringup.launch robot_ip:=<robot_ip> \
 kinematics_config:=$(rospack find ur_calibration)/etc/ur10_example_calibration.yaml

The next goal is executing the test codes: ay_test/ros/py_ros/ur2.
$ cd ~/prg/ay_test/ros/py_ros/ur2/
$ ./kdl_test1.py
$ ./kdl_test2.py
Worked without changes from ay_test/ros/py_ros/ur.

Since the **joint name order is changed**
from ['shoulder_pan_joint', 'shoulder_lift_joint', 'elbow_joint', 'wrist_1_joint', 'wrist_2_joint', 'wrist_3_joint']
to an alphabetical order ['elbow_joint', 'shoulder_lift_joint', 'shoulder_pan_joint', 'wrist_1_joint', 'wrist_2_joint', 'wrist_3_joint']
we need a remapping of joint angles.
It was due to ros_control.
e.g. **/joint_states topic uses the alphabetical order**.
$ ./get_q1.py
$ ./get_q2.py
Worked.

In the following trajectory code, we needed to modify the action server name
from /follow_joint_trajectory
to **/scaled_pos_joint_traj_controller/follow_joint_trajectory** (This topic name is also changed from the previous test of Universal_Robots_ROS_Driver).
$ ./follow_q_traj2.py
Worked. It also uses a reordering version of GetState defined in get_q1.py.

Test (Velocity Control)

Related discussions on GitHub:
- https://github.com/UniversalRobots/Universal_Robots_ROS_Driver/issues/231
    - This Python script may be useful.
- https://github.com/UniversalRobots/Universal_Robots_ROS_Driver/issues/267

Topics:
- /joint_group_vel_controller/command

- Type: std_msgs/Float64MultiArray
- NOTE: The order of this array is from the root joint to the wrist joint. Unlike the joint_state topic, it is not alphabetical order of joint names.

The script from the issue#231 worked:
Entering to a velocity control mode:
$ rosservice call /controller_manager/switch_controller "stop_controllers: ['scaled_pos_joint_traj_controller']"
$ rosservice call /controller_manager/switch_controller "start_controllers: ['joint_group_vel_controller']"
Sending topics to the /joint_group_vel_controller/command message:
$ ./velctrl2.py -0.1
Press Ctrl+C to quit.

In this velocity control mode, the joint trajectory controller does not work.
Exiting the velocity control mode:
$ rosservice call /controller_manager/switch_controller "stop_controllers: ['joint_group_vel_controller']"
$ rosservice call /controller_manager/switch_controller "start_controllers: ['scaled_pos_joint_traj_controller']"

This script includes the control switching, i.e. you do not need to call /controller_manager/switch_controller.
$ ./velctrl3.py
Press Ctrl+C to quit.

ISSUE#277: The robot moves slightly after switching the controller from velocity to trajectory controllers

The order of values in **/joint_group_vel_controller/command** topic:
The order of this array is from the root joint to the wrist joint. Unlike the joint_state topic, it is not alphabetical order of joint names.
In order to confirm it, use velctrl3.py with specifying the moving joint index.
$ ./velctrl3.py 3
Press Ctrl+C to quit.

This script operates the robot with a keyboard. It tests switching the trajectory and velocity controllers.
$ ./keyctrl1.py
Result: It seems that we can change controllers smoothly when the target velocity is zero.
However, when the target velocity of the velocity controller is not zero, the ISSUE#277 seems to happen.

Test (UR3)

All above tests worked both on UR3e(a) and UR3(a).
I did not find outstanding differences.
Note that the ISSUE#277 happened in both cases.