Subscriber tutorial

Goal

The goal of this tutorial is to learn how to create a subscriber node in ROS 2 in C++. The subscriber will subscribe to the topic */joint_state*s and then print the joint names with the corresponding positions.

PKG repo: <link_here>
ROS 2 tutorial: link

Clone the tutorial

Navigate the source folder (agimus_ws/src) of your ROS 2 workspace and clone the following package to get started:

```
git clone
https://agimus-user:frpTR_--SSsbKWRJkK5V@gitlab.com/pal-robotics/agimus_winte
r_school/tutorials/agimus_subscriber_tutorial.git
```

Create a class

1.
First, in the header file, *joint_subscriber.hpp*, add the necessary dependencies, given below:

```
#include <memory>
#include <algorithm>
#include <iostream>

#include "rclcpp/rclcpp.hpp"
#include "sensor_msgs/msg/joint_state.hpp"
```

2.
Create a class, *JointSubscriber*, that inherits from the *rclcpp::Node* class. Use both the files *joint_subscriber.cpp* and *joint_subscriber.hpp* file. Create a simple constructor for this class. You will find a simple subscriber in the ROS2 tutorials.

3.
Register the class as a component node as done in the ROS 2 tutorial. The advantage of using component nodes is that the node does not require a main function to be started.

4.
Add the following to CmakeLists.txt following the public ROS 2 tutorial. Instead of *ament_cmake*, use *ament_cmake_auto*, this simplifies the structure of CmakeLists.txt.

- Create a library that contains *joint_subscriber.cpp*.
- Register the node as a component node in the previously created library.

5.
Add the required dependencies to the package.xml:

- rclcpp
- rclcpp_components
- sensor_msgs

Create functions

1.
To get more information about the topic interface run the following command in a terminal, this shows the required input for the action goal and the received result:

```
ros2 interface show sensor_msgs/msg/JointState
```

2.
Create the following two functions.

```
void joint_callback(const sensor_msgs::msg::JointState::SharedPtr msg);
void print_joint_states(
   const std::vector<std::string> & joint_names,
   const std::vector<double> & joint_positions);
```

The *joint_callback* is used by the subscriber to receive *joint_states* of the robot.

The function *print_joint_states* prints the joint states in the following format. Ensure to only select the joints of the torso and the arm.

```
joint_names: [torso_lift_joint, arm_1_joint, arm_2_joint, arm_3_joint, arm_6_joint,
arm_7_joint, arm_5_joint, arm_4_joint]
joint_positions: [0.14999, 0.2, -1.34, -0.199999, 1.37001, -2.0535e-06, -1.57,
1.93998]
```

3.
Ensure that the subscriber node exits after receiving one message.

Test subscriber

1.
Launch the simulation of TIAGo:

```
ros2 launch tiago_gazebo tiago_gazebo.launch.py
```

2.
Run the *joint_subscriber_node*:

```
ros2 run agimus_subscriber_tutorial joint_subscriber_node
```