ROS support package

Introduction

The ROS support package from Commonplace Robotics GmbH provides integration for CPR Mover and igus Robolink robots into the ROS ecosystem. It features:

- Classes that can be used to control robots using ROS topics.
- URDF robot definition files, that can for example be used to visualize the state of CPR robots in RViz or similar tools.
- A simple RViz plugin allowing to control CPR robots from within RViz.
- Running example targetting the igus robolink 4DOF, small version.
- Running example targetting the CPR Mover6.

Getting started

System Requirements

- The package has been developed and tested on Ubuntu 18.04 with ROS melodic using the GNU 8.3 C++ toolchain.
- A C++17 capable compiler will be required in any case.
- In order to physically connect robots you will need a USB to CAN adapter, testing was done with an adapter from PEAK System.
- We recommend setting your system locale to EN-US because some ROS components seem to have trouble parsing floating point numbers in XML files, if commas are used instead of points to separate the fractional part of floating point numbers by the system locale.
- We recommend having a good C++ editor of your choice at hands, the command line examples on this page will assume Microsoft Visual Studio Code is installed on your machine - it is what has been used during developement of the package in conjunction with the following plugins:
  - C/C++ IntelliSense, debugging, and code browsing
  - CMake langage support
  - Extended CMake support
  - Development support for Robot Operating System

Installation

1. Prerequisites

- Make sure to have ROS melodic (or later) properly installed and configured on your machine. The instructions provided from the ROS wiki might be helpful doing so.
- The package is designed to be built using catkin_make. You will need a properly configured catkin workspace. This tutorial may be helpful setting it up.

2. Clone and build the repository

The source code for the package can be downloaded from our github repository.

1. Clone the package sources by typing the following commands into a terminal from within the root folder of your catkin workspace:
   ```
   cd src
   git clone https://github.com/CPR-Robots/cpr_robot
   cd ..
   ```
2. It may be advisable to reconfigure the CMake cache and clean the build folder. To do so, from within the root folder of your catkin workspace, type the following into a terminal:
   ```
   catkin_make clean
   catkin_make rebuild_cache
   ```
3. Now it is time to actually build the package. From within the root folder of your catkin workspace type the following into a terminal:
   ```
   catkin_make
   ```
4. Finally let's install the package. From within the root folder of your catkin workspace type the following into a terminal:
   ```
   catkin_make install
   ```

3. Run the examples

1. Make sure the robot ist connected to your computer via your CAN to USB adapter and powered on.
2. The CAN interface on your computer needs to be started, whenever the CAN to USB adapter has been plugged in or the computer has started up. To do so, type the following into a terminal:
   ```
   sudo modprobe can_dev
   sudo modprobe can
   sudo modprobe can_raw
   sudo ip link set can0 type can bitrate 500000
   sudo ifconfig can0 up
   ```
   It may be advisable to have a small shellscript with these commands, typing them again each time the adapter hase been plugged in may be cumbersome after a while.
3. Run the RViz plugin with one of the example robot nodes.
   - To control an igus robolink 4DOF, small version robot type the following into a terminal:
     ```
     roslaunch cpr_robot igus_4DOF_SV.launch
     ```
   - If you want to control a CPR Mover6 robot, type the following into a terminal:
     ```
     roslaunch cpr_robot CPRMover6.launch
     ```

4. Take a look at the source code

1. Go to the package source folder:
   ```
   roscd cpr_robot
   ```
2. Open the folder using an editor (in this example Visual Studio Code is used):
   ```
   code .
   ```
3. Have fun playing around with it.

Overview

The Robot node

See the files **main_CPRMover6.cpp** or **main_igus_4DOF_SV.cpp** in order to have a working example on how to set up a ROS node that will allow to integrate robots from igus or CPR into ROS. The general way to do it, is to instantiate a class that has been derived from the abstract Robot base class, then within a standard ROS spinning loop subsequently call the Read, PublishState and Write methods:

```
cpr_robot::CPRMover6 robot;
robot.Init();
ros::Rate loop_rate(10);
while (ros::ok())
{
  robot.Read();
  robot.PublishState();
  robot.Write();
  ros::spinOnce();
  loop_rate.sleep();
```

```
}
```

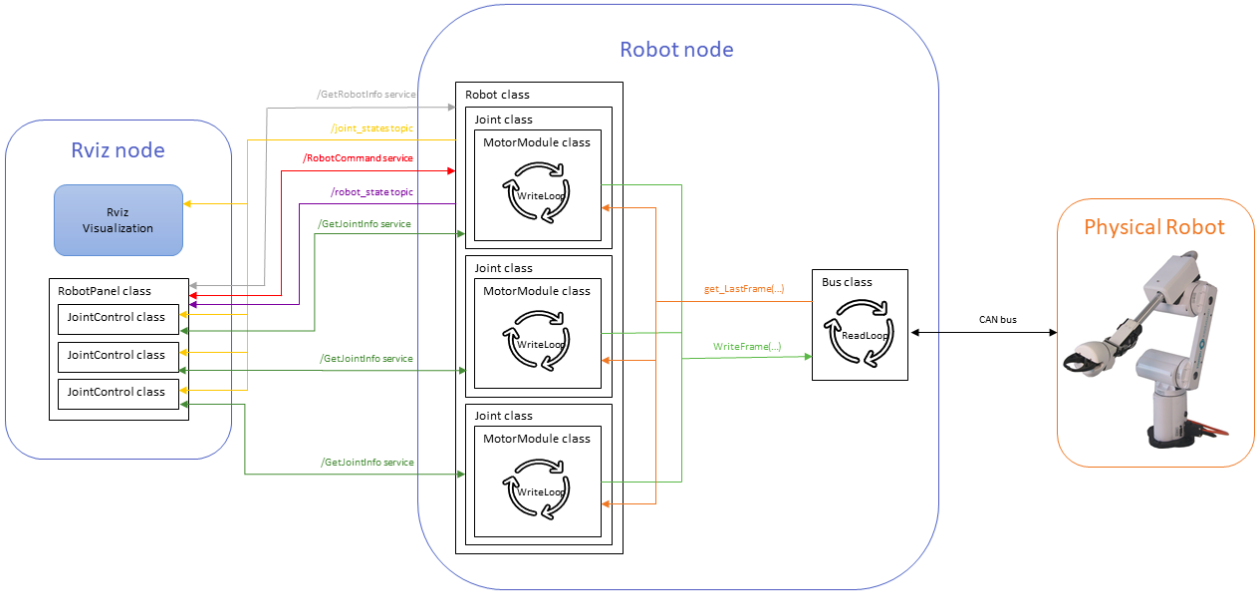
The RViz plugin

The complete source code for the RViz plugin is contained in the files **RobotPanel.cpp** and **JointControl.cpp** and their respective headers. You can use these a a starting point for the developement of your own plugin. Note that you will have to run

```
catkin_make install
```

for Panels (and other RViz plugins) to become accesible from within RViz.

Dataflow schema



---

cpr_robot
Author(s):
autogenerated on Thu Jun 13 2019 10:47:30