Com S 476/576 Lab 1

Introduction to Robot Operating System (ROS)

Yuechuan Xue
yuechuan@iastate.edu

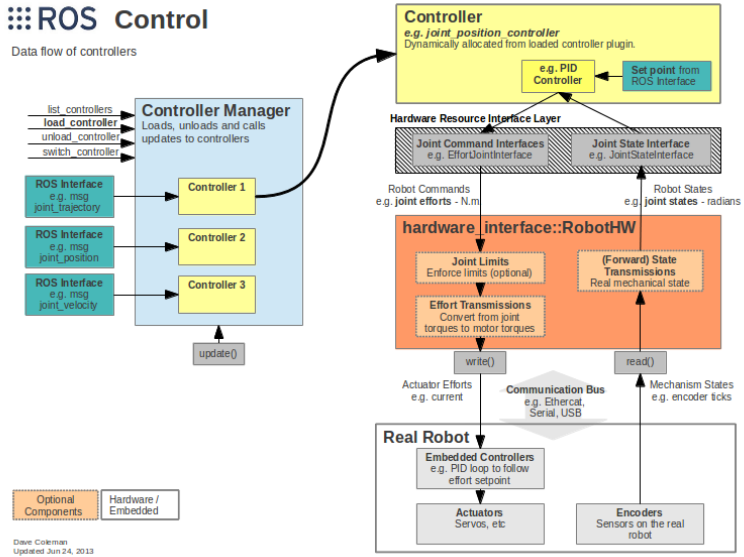
Com S 476/576, Spring 2021

ROS Control

`ros_control`

- A ROS package to make controllers generic to all robots
- Takes joint state data from your robot's actuator's encoders and an input set point
- Uses a generic control loop feedback mechanism (typically a PID controller) to control
- Sends output (typically effort) to your actuators

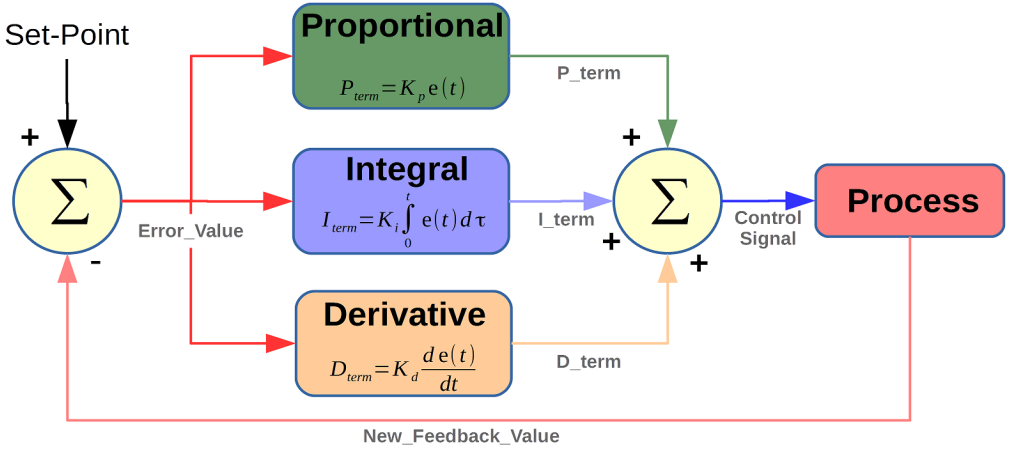
ROS Control Overview



Controllers

- effort_controllers
  Command a desired force/torque to joints.
  - joint_effort_controller
    force/torque input is directly output to the joint
  - joint_position_controller
    position input goes into a PID controller that outputs force/torque to the joint
  - joint_velocity_controller
    velocity input goes into a PID controller that outputs force/torque to the joint
- joint_group_position_controller
  Set one or multiple joint positions at once.
- joint_trajectory_controllers
  Extra functionality for splining an entire trajectory.
- ...

PID Controller



Code Repositories

ros-controls
github.com/ros-controls

- `control_msgs`
  messages and actions useful for controlling robots
- `realtime_tools`
  tools that can be used from a hard realtime thread
- `control_toolbox`
  tools useful for writing controllers and robot abstractions
- `ros_control`
  generic and basic controller framework for ROS
- `ros_controllers`
  generic robot controllers for `ros_control`

URDF Transmission

```
<joint name="shoulder_pan_joint" type="revolute" >
    <parent link="base_link"/>
    <child link="shoulder_link"/>
    <origin rpy="0.0 0.0 0.0" xyz="0.0 0.0 0.1273"/>
    <axis xyz="0 0 1"/>
    <limit effort="330.0" lower="-6.28" upper="6.28" velocity="2.16"/>
    <dynamics damping="0.0" friction="0.0"/>
</joint>

<transmission name="shoulder_pan_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="shoulder_pan_joint">
        <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
    </joint>
    <actuator name="shoulder_pan_motor">
        <mechanicalReduction>1</mechanicalReduction>
    </actuator>
</transmission>
```

Add the gazebo_ros_control plugin

- In addition to the transmission tags, a Gazebo plugin needs to be added to your URDF that actually parses the transmission tags and loads the appropriate hardware interfaces and controller manager.
- By default the `gazebo_ros_control` plugin is very simple, though it is also extensible via an additional plugin architecture to allow power users to create their own custom robot hardware interfaces between `ros_control` and Gazebo.

```
<gazebo>
  <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
    <robotNamespace>/MYROBOT</robotNamespace>
  </plugin>
</gazebo>
```

The `gazebo_ros_control` ⟨plugin⟩ tag also has the following optional child elements:

- ⟨`robotNameSpace`⟩: The ROS namespace to be used for this instance of the plugin, defaults to robot name in URDF/SDF

Launch a Gazebo world

```
<launch>
  <arg name="limited" default="false" />
  <arg name="transmission_hw_interface" default="hardware_interface/EffortJointInterface" />

  <!-- startup simulated world -->
  <include file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="world_name" default="worlds/empty.world"/>
    <arg name="paused" value="false"/>
    <arg name="gui" value="true"/>
  </include>

  <param name="robot_description"
         command="$(find xacro)/xacro --inorder
         '$(find ur10_description)/urdf/ur10_robot.urdf.xacro'
         transmission_hw_interface:=$(arg transmission_hw_interface)" />

  <!-- push robot_description to factory and spawn robot in gazebo -->
  <node name="spawn_gazebo_model" pkg="gazebo_ros" type="spawn_model"
        args="-urdf -param robot_description -model robot -z 0.1"
        respawn="false" output="screen" />

  ...
```

Launch a Gazebo world

```
...

<!-- Robot state publisher -->
<node name="robot_state_publisher"
      pkg="robot_state_publisher" type="robot_state_publisher">
  <param name="publish_frequency" type="double" value="50.0" />
  <param name="tf_prefix" type="string" value="" />
</node>

<!-- Fake Calibration -->
<node pkg="rostopic" type="rostopic" name="fake_joint_calibration"
      args="pub /calibrated std_msgs/Bool true" />

<!-- joint_state_controller -->
<rosparam file="$(find ur10_controller)/controller/joint_state_controller.yaml"
          command="load"/>
<node name="joint_state_controller_spawner"
      pkg="controller_manager" type="controller_manager"
      args="spawn joint_state_controller" respawn="false" output="screen"/>

...
```

Launch a Gazebo world

```
  <!-- start this controller -->
  <rosparam file="$(find ur10_controller)/controller/arm_controller_ur10.yaml"
            command="load"/>
  <node name="arm_controller_spawner"
        pkg="controller_manager" type="spawner"
        args="shoulder_pan_joint_position_controller
              shoulder_lift_joint_position_controller
              elbow_joint_position_controller
              wrist_1_joint_position_controller
              wrist_2_joint_position_controller
              wrist_3_joint_position_controller"
        respawn="false" output="screen"/>
</launch>
```

Controller Params

In the `arm_controller_ur10.yaml`, we define the PID parameters for the arm controllers

```yaml
# Position Controllers ---------------------------------------
shoulder_pan_joint_position_controller:
  type: effort_controllers/JointPositionController
  joint: shoulder_pan_joint
  pid: {p: 1000.0, i: 0.01, d: 10.0}
shoulder_lift_joint_position_controller:
  type: effort_controllers/JointPositionController
  joint: shoulder_lift_joint
  pid: {p: 1000.0, i: 0.01, d: 10.0}
elbow_joint_position_controller:
  type: effort_controllers/JointPositionController
  joint: elbow_joint
  pid: {p: 1000.0, i: 0.01, d: 10.0}

...
```

Send a Message to the Controller

Use `ros_topic` to publish a one-time message to a `ros_control` topic

```
rostopic pub /elbow_joint_position_controller/command std_msgs/Float64 "data: 0.5" -1
```