# Ros Controllers

⇒ The ros_control package provide a set of controller plugins in order to interact in different ways with the joints of your robot.

1. effort_controllers
   ↳ joint you want to control accept effort commands.
   - Joint_effort_controller
     ↳ accepts effort as input value
   - joint_position_controller
     ↳ accepts position as input value
   - joint_velocity_controller
     ↳ accepts velocity as input.
2. Position_controllers
   ↳ joint you want to control accepts position commands.
   - joint_position_controller
3. Velocity_controllers
   ↳ joint you want to control accepts velocity commands.
   - joint_velocity controller
4. joint_state_controller
   ↳ This plugin provides the state of the joints, Publishing them into a topic called /joint_states.
   - joint_state_controller

## * Configration file

robotname_control.yaml

```
robotname:
    joint_state_controller:
        type: joint_state_controller/JointStateController
        publish_rate: 50

    joint1_position_controller:
        type: effort_controllers/JointPositionController
        joint: Jointname
        pid: {p:100.0, i:0.01, d: 10.0}

    ⋮   ⋮   ⋮   ⋮   ⋮
```

## * Launch file

```
<launch>
    <include file="$(find my_package)/launch/Robotname
                                    _world.launch"/>
    <rosparam file="$(find my-package)/config /
                   Robotname_gazebo_control.yaml"
                   command="load"/>
    <node name="controller_spawner" pkg="controller_manager"
        type="spawner" respawn="false" output="screen"
        ns="/Robotname args='joint_state controller
                joint1_position_controller
                    ⋮   ⋮   ⋮   ⋮
                          />
```

```
<node name ="robot_state_publisher" pkg ="robot_state_publisher"
    type = "robot_state_publisher"
    respawn = "false"  Output ="Screen">
    <remap from="/joint_states" to="/robotname/joint_states"/>
</node>

</launch>
```

---

★ <u>Transmission tag</u>

- Attribute name (required).
- Elements
  - <type> {Specifies the transmission type}
  - <joint>
    - Attribute name
      {Joint the transmission is connected to}
    - <hardwareInterface>
      {Specify a supported joint-space hardware interface}
  - <actuator>
    {The actuator the transmission is connected to}

*

→ Attribute name

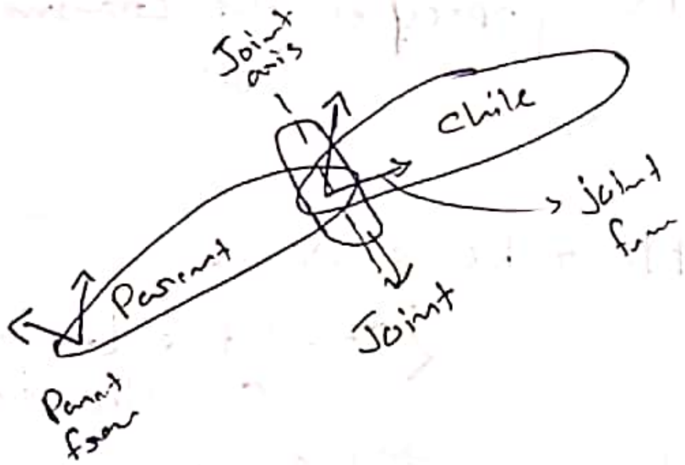
{name of the actuator}

→ <mechanical Reduction>

{Specifies a mechanical reduction at the joint/actuator transmission

## ★ Joint tag

⇒ The joint describe the Kinematics and dynamics of the joint and also specifies the Safety limits of the joint.



## # Attributes

→ name {Unique name for the joint}

→ type {Specifies the type of joint}

*

*
- ~~Reo~~ revolute {hinge joint with limited range}
- Continuous {hinge joint with no limit on range}
- Prismatic {Sliding joint that slides along the axis, and has a limited range}
- fixed {This is not really a joint because it cannot move}
- floating {Have all 6 DOF}
- Planar {Allows motion in a plane ⊥ to the axis}

# Elements

- <origin>
  - {This is the transform from the Parent link to the child link}
  - xyz {Represents xyz Offset)
  - rpy {Represents rotation about the corresponding axis.}
- <Parent>
  - link {Parent link name)
- <child>
  - link {child link name}
- * <axis> → xyz

- <calibration>

  {The reference positions of the joint, used to Calibrate the absolute Position of the joint}

  - rising

    {when the joint moves in a positive direction, this reference position will trigger a rising edge}

  - falling

    {When the joint movo in a positive direction, this reference position will trigger a falling edge}

- <dynamic>

  {for specifying physical properties of joint}

  - damping
  - friction

- <limit> (required for revolute and presmatic joint)

  - lower
  - upper
  - effort {for enforcing max joint effort
  - velocity {for enforcing max joint velocity

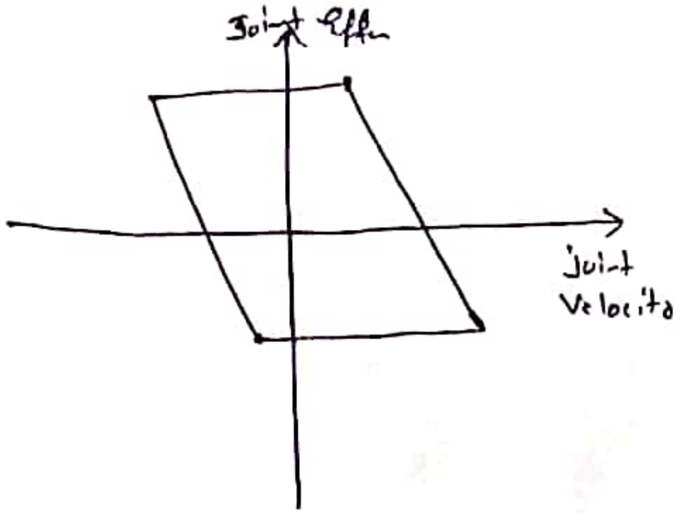
- <mimic>
  - Attribute joint (to mimic)
  - multiplier
  - Offset

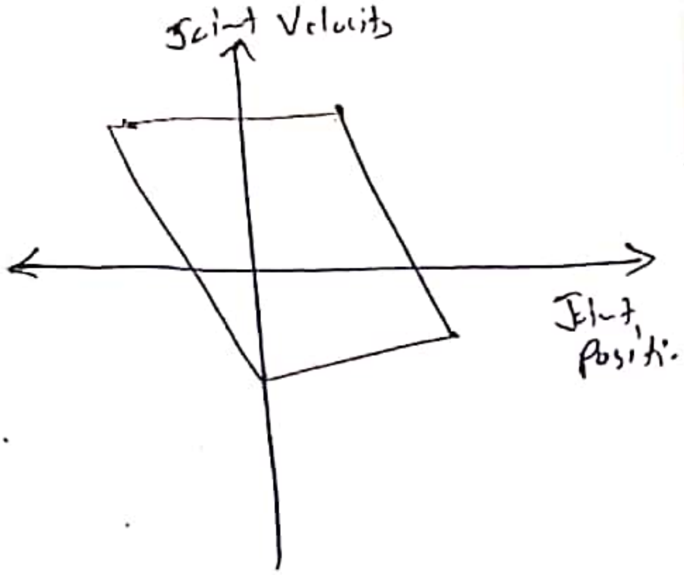
⇒ This tag is used to specify that the defined joint mimics another existing joint.

⇒ The value of the joint can be computed as

$$\left\{ \text{Value} = (\text{multiplier} \times \text{other\_joint\_Value}) + \text{offset} \right\}$$

- <Safety_Controller>
  - Soft_lower_limit
  - Soft_upper_limit
  - K_position
  - K_velocity


Joint Effort
Joint Velocity


Joint Velocity
Joint Position