ros2_control
The future of ros_control
What & where

pr2_controller_manager
(pr2_mechanism)
2009

ros_control
2012/2013

ros2_control
2017/2021
Maximize resources spent on the actual **client app**

- Leverage existing controllers
  - Implement **custom ones**
- Leverage simulation backend
  - **Real robot** backends
Mobile manipulator example

- Head controller / gaze controller
- End-effector controller
- Arm controller / arm-torso controller
- Mobile-base controller
- State broadcaster(s)
Standard controllers

- joint_state_broadcaster
- diff_drive_controller
- joint_trajectory_controller
- gripper_controllers

Forwarding controllers for (groups of) joints
  - position_controllers
  - velocity_controllers
  - effort_controllers
- General, robot-agnostic framework
- Supported joint interfaces: position, velocity, effort
- Code complexity high, lots of templating and inheritance
- Controller lifecycle inspired by Orocos, custom
- Unclear semantics: everything is in the RobotHW or controller
- Opt-in Hardware Composition
- RobotHW and boilerplate code
- Collection of official controllers, defining de-facto standard ROS interfaces to 3rd party
- Off-the-shelf Gazebo integration
- Stability
- General, robot-agnostic framework
- Supported joint interfaces: position, velocity, effort
- Collection of official controllers, defining de-facto standard ROS interfaces to 3rd party
- Off-the-shelf Gazebo integration
- Code complexity high, lots of templating and inheritance
- Controller lifecycle inspired by Orocos, custom
- Unclear semantics: everything is the RobotHW or controller
- Opt-in Hardware Composition
- RobotHW and boilerplate code
- Stability

- ✔️ Supported joint interfaces: no limitations
- ✔️
- ✔️
- Code leaner, more modern C++
- Controller lifecycle via ROS2 LifecycleNode
- [System|Actuator|Sensor]Component, Controller and Broadcaster
- Hardware Composition is first class citizen
- Default ros2_control_node
- Hardware lifecycle
- Synchronous but variable rate for controllers
- Stay tuned!
● General, robot-agnostic framework
● Supported joint interfaces: no limitations
● Collection of official controllers, defining de-facto standard ROS interfaces to 3rd party
● Off-the-shelf Gazebo integration
● Lean and extendable structure using modern C++
● Controller lifecycle via ROS2 LifecycleNode
● [System|Actuator|Sensor]Component, Controller and Broadcaster
● Full Hardware Composition support
● Default ros2_control_node
● Hardware lifecycle
● Synchronous but variable rate for controllers
● Asynchronous controllers
● Joint limiting plugin
● Emergency stop handler plugin
Maximize resources spent on the actual **client app**

- **Leverage** existing controllers
- Implement **custom ones**
- **Leverage** simulation backend
- **Real robot** backends
Maximize resources spent on actual client application

- Leverage existing controllers
- Implement custom ones, extend existing

- Leverage simulation backend
- Real robot backend - extend existing ones or create your own
Command and state interfaces
URDF extension with `<ros2_control>`-tag

```xml
<ros2_control name="robot" type="system">
    <hardware>
        <plugin>robot_package/Robot</plugin>
        <param name="hardware_parameter">some_value</param>
    </hardware>

    <joint name="joint_first">
        <command_interface name="position"/>
        <state_interface name="acceleration"/>
    </joint>

    <joint name="joint_last">
        <command_interface name="velocity">
            <param name="min">-1</param>
            <param name="max">1</param>
        </command_interface>
        <state_interface name="temperature"/>
    </joint>

    <sensor name="tcp_sensor">
        <state_interface name="sensing_interface">
            <param name="sensor_parameter">another_value</param>
        </state_interface>
    </sensor>

    <gpio name="rrbot_status">
        <state_interface name="mode" data_type="int"/>
        <state_interface name="bit" data_type="bool" size="4"/>
    </gpio>

</ros2_control>

<ros2_control name="tool" type="actuator">
    <hardware>
        <plugin>tool_package/Tool</plugin>
        <param name="hardware_parameter">some_value</param>
    </hardware>

    <joint name="tool">
        <command_interface name="command"/>
    </joint>

</ros2_control>
```
Syntax cookies: semantic components
Lifecycle for controllers and hardware

- Managed nodes - interface
  - Hardware components
  - Controllers

https://design.ros2.org/articles/node_lifecycle.html
## ros2_control CLI

Integrated with ROS2 CLI

<table>
<thead>
<tr>
<th>Command</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>$ ros2 control list_controllers</code></td>
<td>base_controller[DiffDriveController] active</td>
</tr>
</tbody>
</table>
| `$ ros2 control list_controller_types` | diff_drive_controller/DiffDriveController controller_interface::ControllerInterface  
joint_state_controller/JointStateController controller_interface::ControllerInterface  
joint_trajectory_controller/JointTrajectoryController controller_interface::ControllerInterface |
| `$ ros2 control list_hardware_interfaces` | command interfaces  
 joint1/position [unclaimed]  
 joint2/position [unclaimed]  
 state interfaces  
 joint1/position  
 joint2/position |
References

- ros_control paper in the Journal of Open Source Software
- ros2_control resources
  - https://control.ros.org
  - https://github.com/ros-controllers/ros2_control
  - https://github.com/ros-controllers/ros2_controllers
  - https://github.com/ros-controllers/ros2_control_demos
  - https://github.com/ros-controllers/roadmap/blob/master/documentation_resources.md
Thank you!

Karsten Knese, Victor Lopez, Jordan Palacios, Olivier Stasse, Mathias Arbo, Jaron Lundwall, Colin MacKenzie, Matthew Reynolds, Andy Zelenak, Lovro Ivanov, Jafar Abdi, Tyler Weaver, Márk Szitanics, Anas Abou Allaban, Yutaka Kondo, Mateus Amarante, Auguste Bourgois and many more!