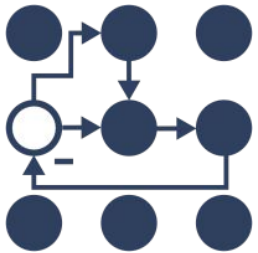
What is new in the best (and only) control framework for ROS2 - ros2_control



denis.stogl@stoglrobotics.de

Denis Štogl

- PhD in Robotics from KIT (Karlsruhe, Germany)
- Founder and CEO of Stogl Robotics Consulting
- ros2_control maintainer
- 80% of daily work is ros2_control-related

Presentation outline

1. Present outline ← We are here!
2. Short history and basic concepts
3. How is ros2_control different from ros_control?
4. I want to use ros2_control! Where to start?
    - About robot description, hardware drivers and controllers
5. But my hardware is complex…
6. Panic! My controllers are getting too convoluted…
7. And what if I have multiple robots?
8. Resources and persons behind ros2_control

What & where

pr2_controller_manager

(pr2_mechanism)

2009

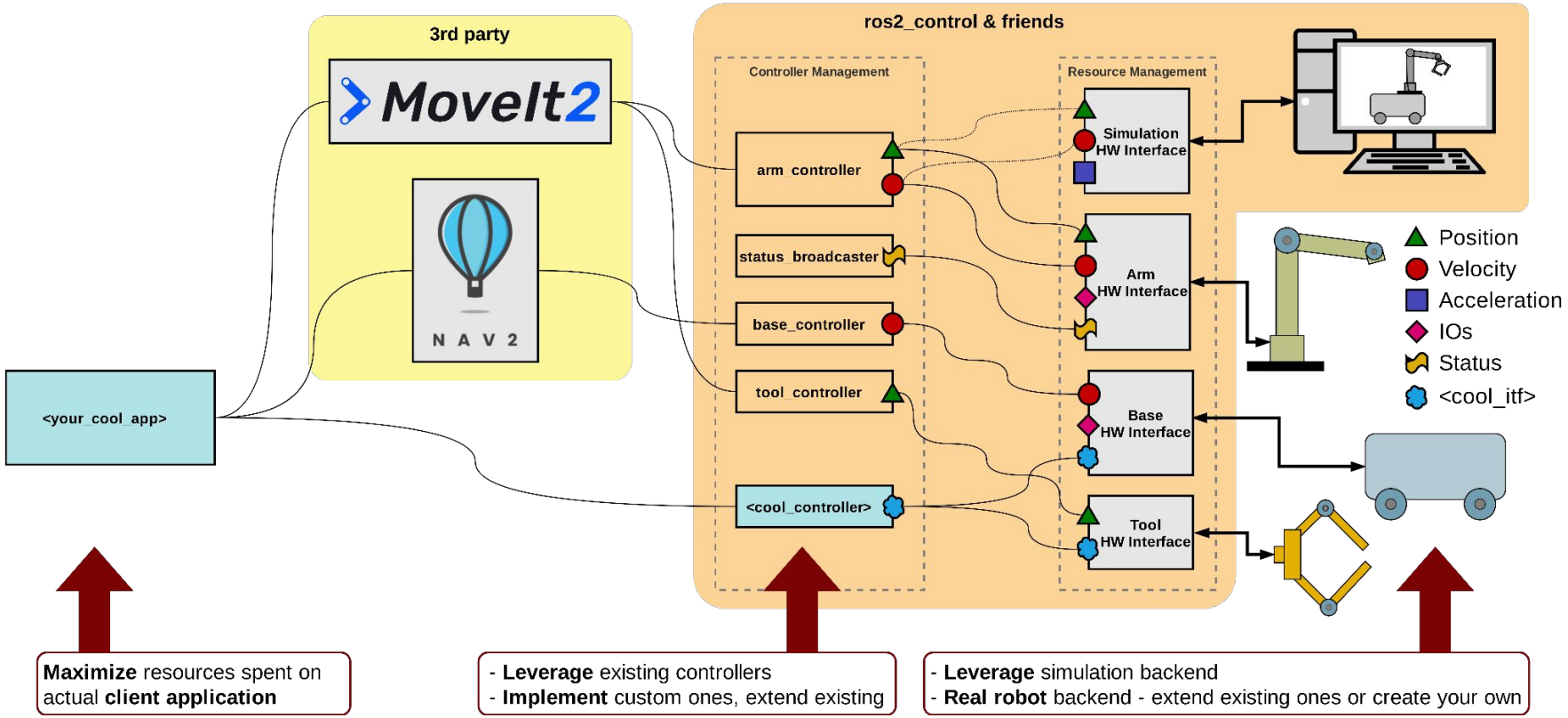
ros_control

2012/2013

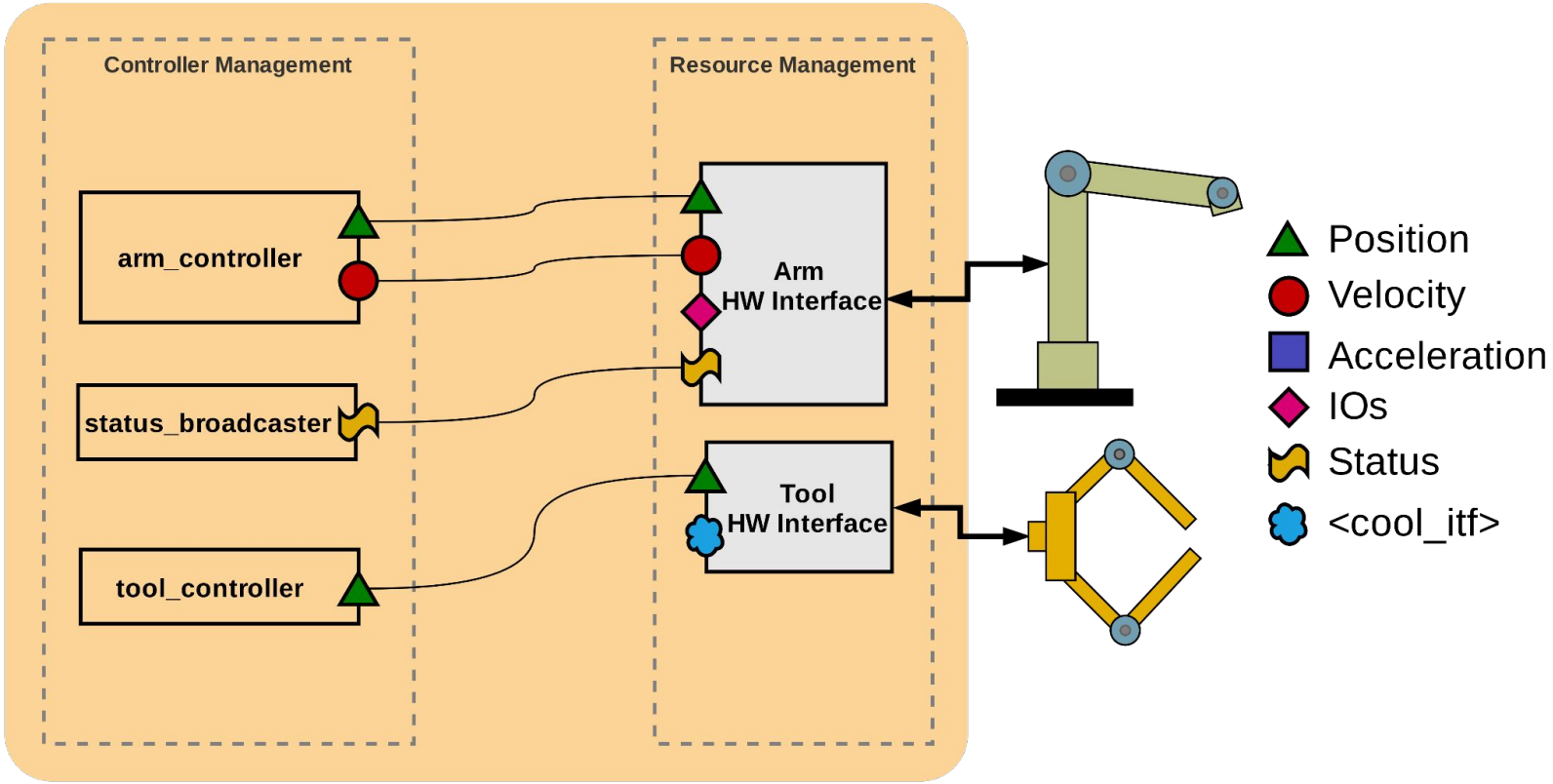
ros2_control

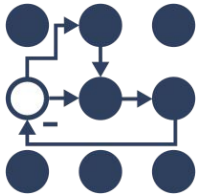
2017/2020

https://control.ros.org/master/doc/supported_robots/supported_robots.html

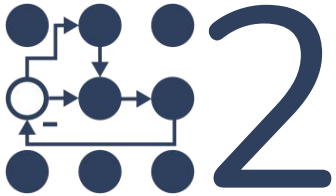

3rd party
MoveIt2
N A V 2
ros2_control & friends
Controller Management
Resource Management
arm_controller
status_broadcaster
base_controller
tool_controller
<cool_controller>
Simulation HW Interface
Arm HW Interface
Base HW Interface
Tool HW Interface
<your_cool_app>
Position
Velocity
Acceleration
IOs
Status
<cool_itf>
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





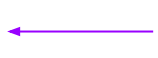
- General, robot-agnostic framework
- Collection of official controllers, defining de-facto standard ROS interfaces to 3rd party
- Off-the-shelf Gazebo integration
- Stability
- Supported joint interfaces: position, velocity, effort
- Code complexity high, lots of templating and inheritance
- Controller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



- ✅
- ✅
- ✅
- Stay tuned!
- 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
- Asynchronous controllers
- Joint limiting plugin
- Emergency stop handler plugin

Presentation outline

1. Present outline
2. Short history and basic concepts
3. How is ros2_control different from ros_control?
4. I want to use ros2_control! Where to start?
    - About robot description, hardware drivers and controllers
5. But my hardware is complex…
6. Panic! My controllers are getting too convoluted…
7. And what if I have multiple robots?
8. Resources and persons behind ros2_control



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>

  . . .

  <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>
```

```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_inteface"/>
    <param name="sensor_parameter">another_value</param>
  </sensor>

  <gpio name="flange_IOs">
    <command_interface name="digital_output" data_type="bool" size="8" />
    <state_interface name="digital_output" data_type="bool" size="8" />
    <command_interface name="analog_output" data_type="double" size="2" />
    <state_interface name="analog_output" data_type="double" size="2" />

    <state_interface name="digital_input" data_type="bool" size="4" />
    <state_interface name="analog_input" data_type="double" size="4" />
  </gpio>

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

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

</ros2_control>
```

Implementing hardware interface (driver)

export_state_interfaces()
- Which states are available from HW?

export_command_interfaces()
- What can be commanded on HW?

on_init()
- read and process URDF parameters
- initialize all variables and containers

on_activate (previous_state)
- activate power of HW to enable movement

read()
- Fill states from HW readings

write()
- Write commands to HW

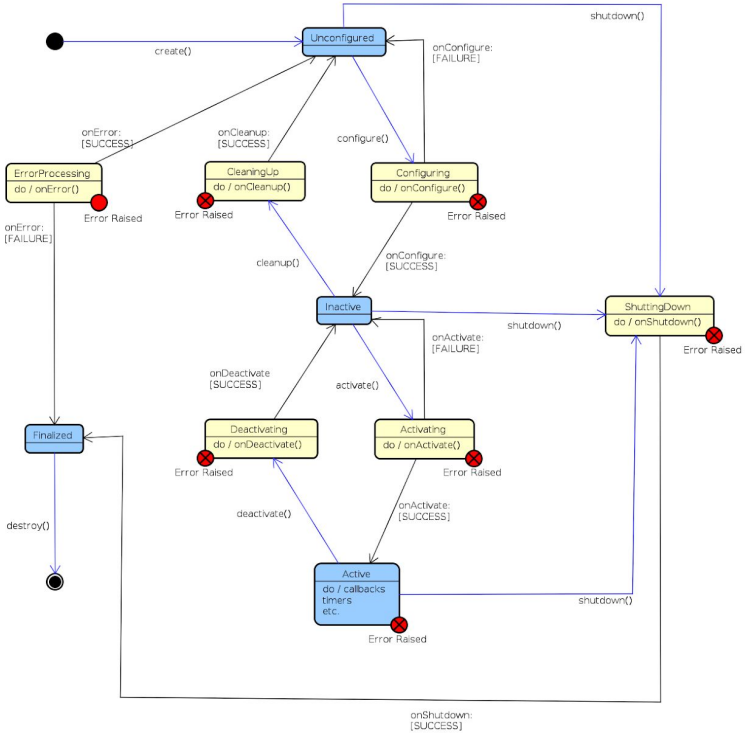
on_configure (previous_state)
- initiate communication with the HW
- be sure HW states can be read

on_deactivate (previous_state)
- disable HW movement

on_cleanup (previous_state)
- disable communication

on_error (previous_state)
- process and mitigate any errors
- it can happen in any state
- catching errors during read/write

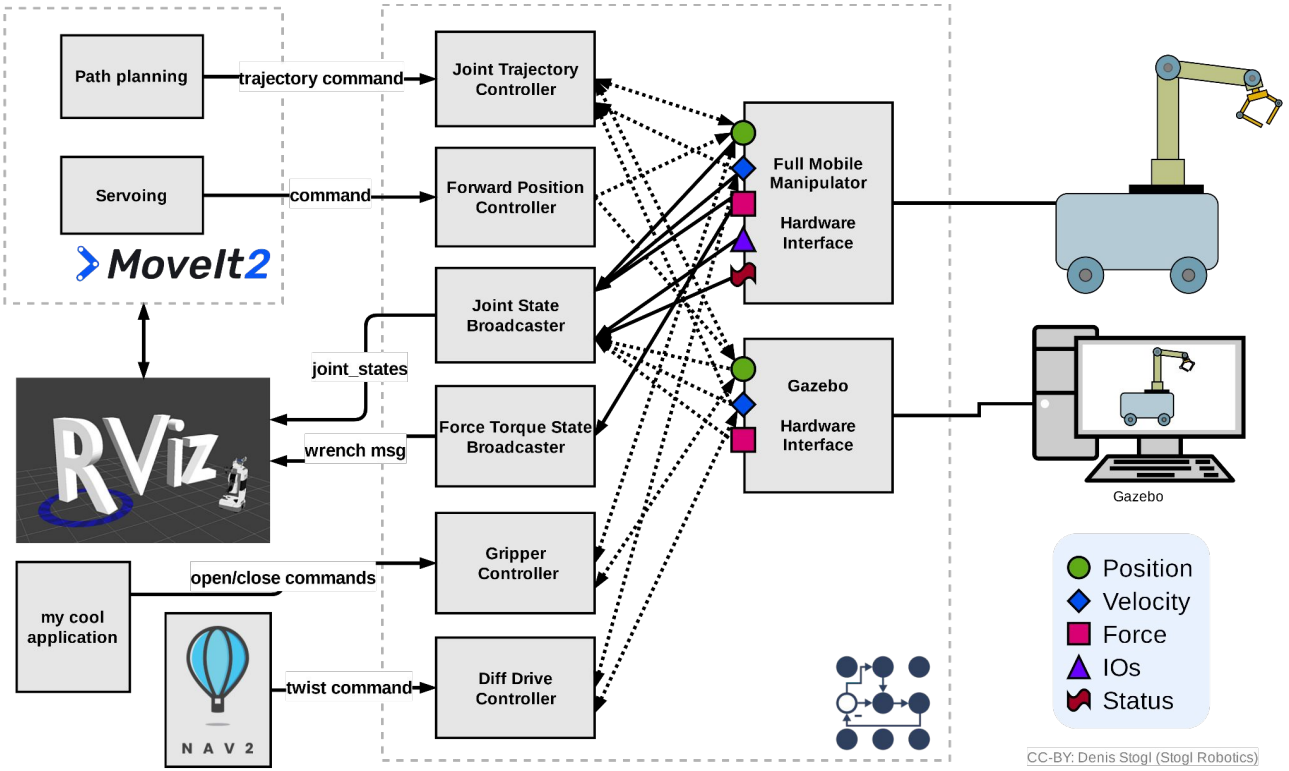
on_shutdown (previous_state)
- initiate HW shutdown sequence
- can be called from any state



https://design.ros2.org/articles/node_lifecycle.html

Configuring standard controllers

```yaml
controller_manager:
  update_rate: 500  # Hz

  joint_trajectory_controller:
    type: joint_trajectory_controller/JointTrajectoryController

  forward_position_controller:
    type: position_controllers/JointGroupPositionController

  joint_state_broadcaster:
    type: joint_state_broadcaster/JointStateBroadcaster

  force_torque_sensor_broadcaster:
    type: force_torque_sensor_broadcaster/ForceTorqueStateBroadcaster

  gripper_controller:
    type: position_controllers/GripperActionController

  diff_drive_controller:
    type: diff_drive_controller/DiffDriveController


joint_trajectory_controller:
  joints:
    - joint1
    - ...
  command_interfaces:
    - position
  state_interfaces:
    - position
    - velocity


forward_position_controller:
  joints:
    - joint1
    - ...


force_torque_sensor_broadcaster:
  sensor_name: tcp_fts_sensor
  frame_id: tool0
  topic_name: ft_data


gripper_controller:
  joints:
    - gripper_joint
  command_interface: position


diff_drive_controller:
  left_weel_names:
    - left_wheel_1
    ...
```



Using different controllers for control modes

export_state_interfaces()
- Which states are available from HW?

export_command_interfaces()
- What can be commanded on HW?

on_init()
- read and process URDF parameters
- initiallize all variables and containers

on_activate (previous_state)
- activate power of HW to enable movement

read()
- Fill states from HW readings

write()
- Write commands to HW

on_configure (previous_state)
- initiate communication with the HW

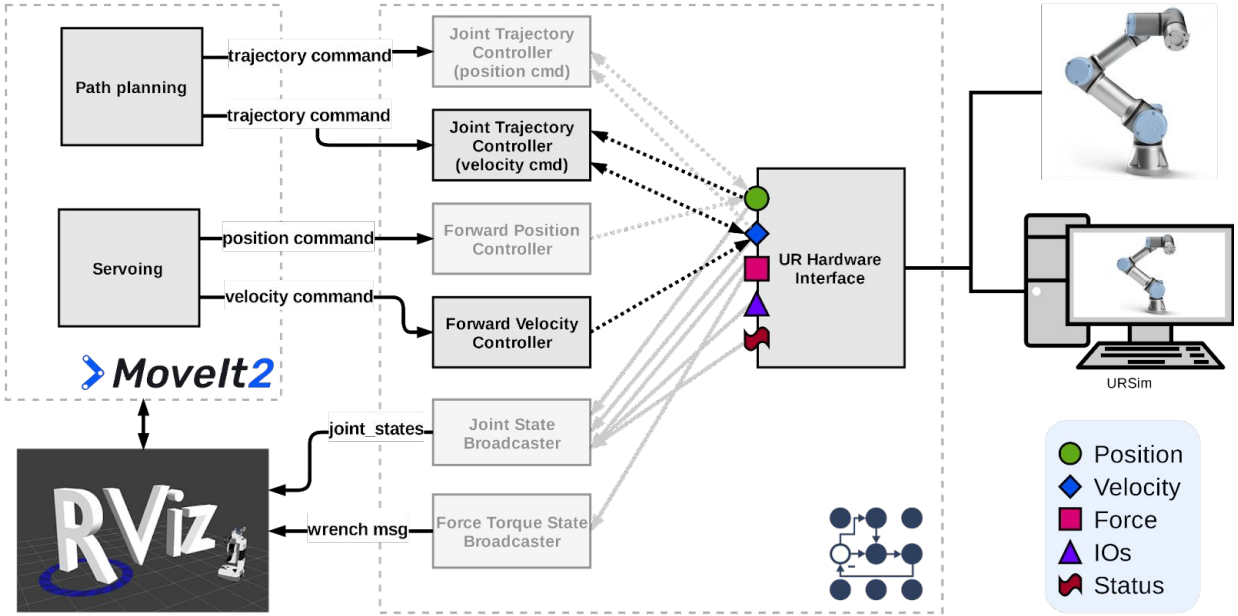
prepare_command_mode_switch (stop_interfaces, start_interfaces)
- Check if mode switch is possible w.r.t. given interfaces
- Only command interfaces are relevant
- Prepare robot for switching (initialize additional variables, etc.)

perform_command_mode_switch (stop_interfaces, start_interfaces)
- perform switching of the hardware
- set/reset internal variables for new/old control mode

on_shutdown (previous_state)
- initiate HW shoutdown sequence
- can be called from any state

Add controllers for other control-mode

- Forwarding controller
- Joint Trajectory controller with different set of command interfaces

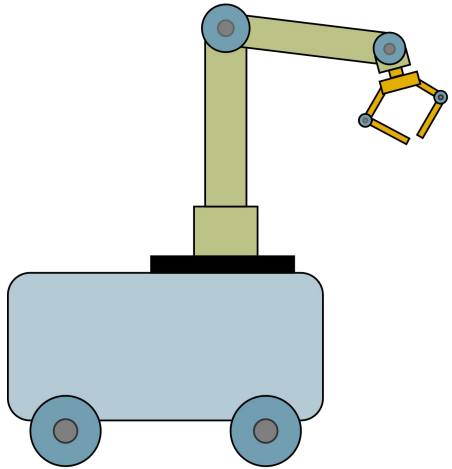


Presentation outline

1. Present outline
2. Short history and basic concepts
3. How is ros2_control different from ros_control?
4. I want to use ros2_control! Where to start?
    - About robot description, hardware drivers and controllers
5. But my hardware is complex…
6. Panic! My controllers are getting too convoluted…
7. And what if I have multiple robots?
8. Resources and persons behind ros2_control

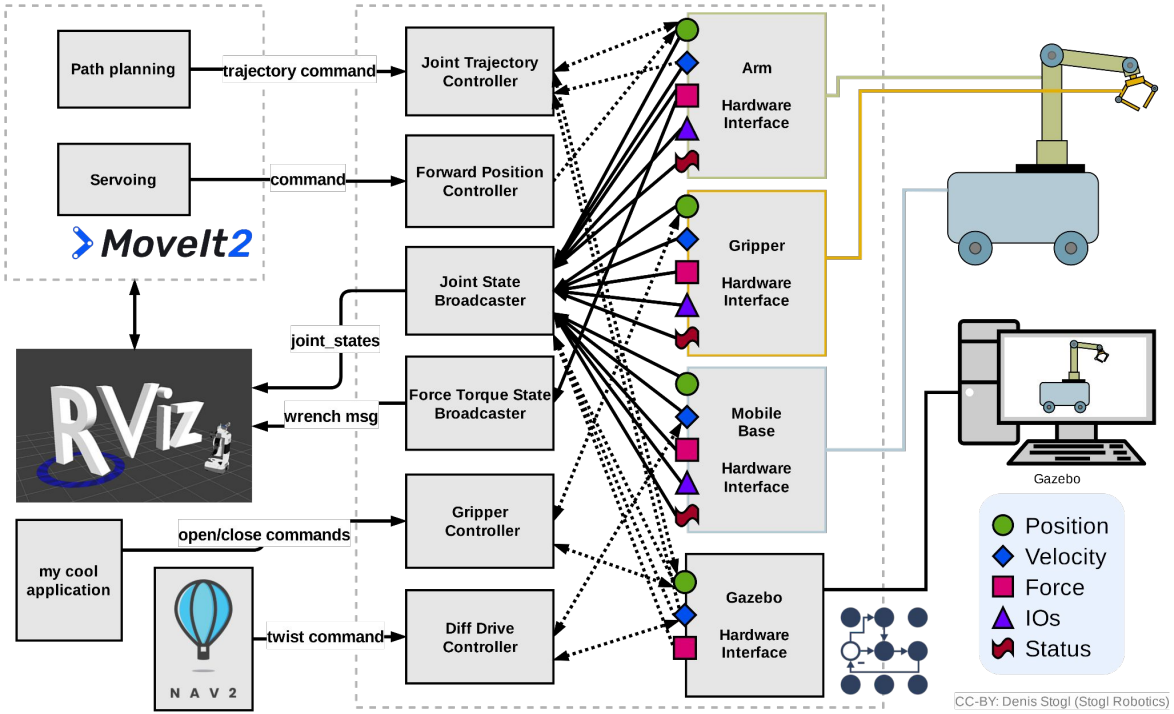
We are here!

About hardware modelling



- Choose hardware interface architecture to your needs
  - *Guideline*: one communication path – one hardware interface
- Check ros2_control_demos repository for different architecture examples
- Profit from modularity of hardware interfaces – “implement only one time”

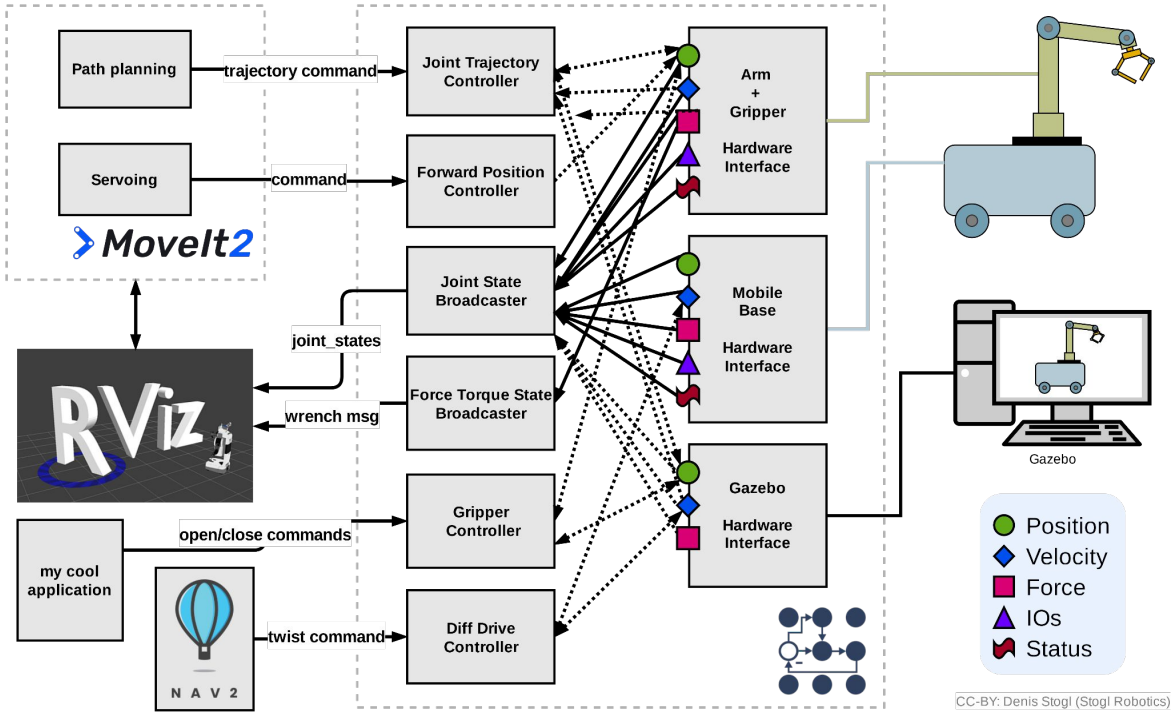
Modelling complex hardware – individual components




```
<ros2_control name="MobileBase" type="system">
  <hardware>
    <plugin>ros2_contro_robot/Mobile_Base</plugin>
    ...
  </hardware>
  <joint name="wheel1_joint">
    <command_interface name="velocity"/>
    <state_interface name="position"/>
    <state_interface name="velocity"/>
    <state_interface name="status"/>
  </joint>
  ...
</ros2_control>

<ros2_control name="Arm" type="system">
  <hardware>
    <plugin>ros2_contro_robot/Arm</plugin>
    ...
  </hardware>
  <joint name="joint1">
    <command_interface name="position"/>
    <state_interface name="position"/>
    <state_interface name="velocity"/>
    <state_interface name="status"/>
  </joint>
  ...
</ros2_control>

<ros2_control name="Gripper" type="actuator">
  <hardware>
    <plugin>ros2_contro_robot/Gripper</plugin>
    ...
  </hardware>
  <joint name="gripper_joint">
    <command_interface name="position"/>
    <state_interface name="position"/>
    <state_interface name="velocity"/>
    <state_interface name="status"/>
  </joint>
  ...
</ros2_control>
```

ros2_control_demos Example: "Modular Robots with separate communication to each actuator"

Modelling complex hardware – “bus through arm” + base

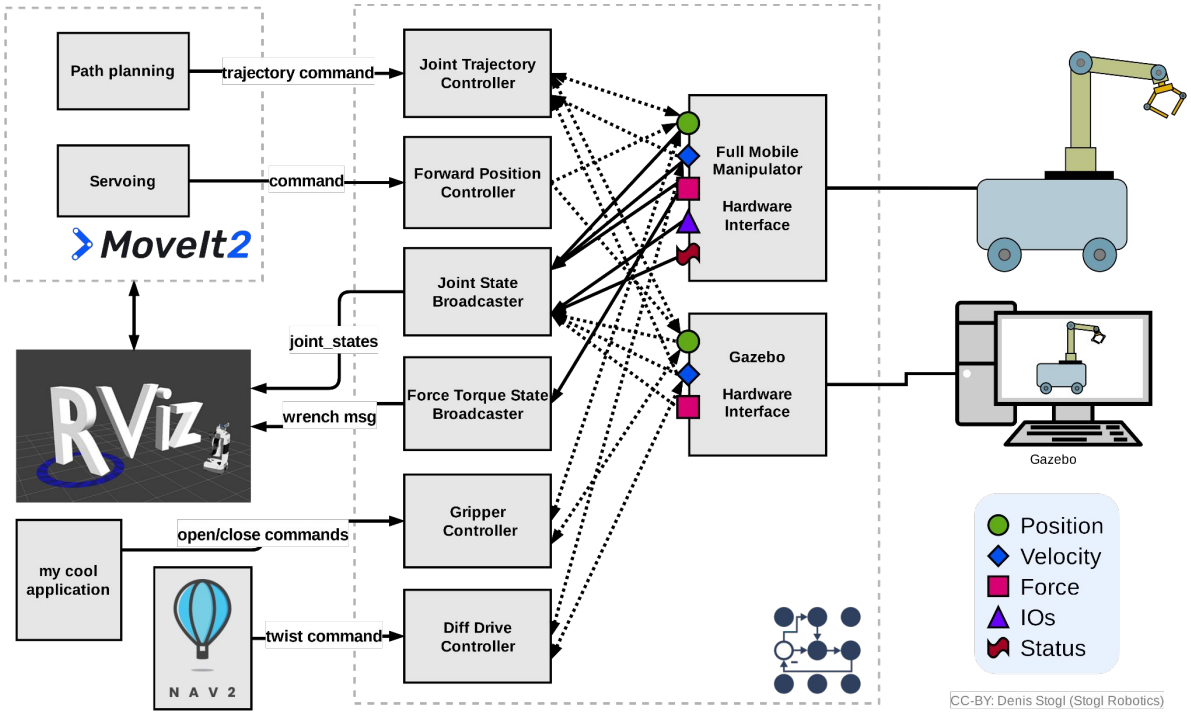



```xml
<ros2_control name="MobileBase" type="system">
  <hardware>
    <plugin>ros2_contro_robot/Mobile_Base</plugin>
    ...
  </hardware>
  <joint name="wheel1_joint">
    <command_interface name="velocity"/>
    <state_interface name="position"/>
    <state_interface name="velocity"/>
    <state_interface name="status"/>
  </joint>
  ...
</ros2_control>
```

```xml
<ros2_control name="ArmWithGripper" type="system">
  <hardware>
    <plugin>ros2_contro_robot/Arm_With_Gripper</plugin>
    ...
  </hardware>
  <joint name="joint1">
    <command_interface name="position"/>
    <state_interface name="position"/>
    <state_interface name="velocity"/>
    <state_interface name="status"/>
  </joint>
  ...
  <joint name="gripper_joint">
    <command_interface name="position"/>
    <state_interface name="position"/>
    <state_interface name="velocity"/>
    <state_interface name="status"/>
  </joint>
  ...
</ros2_control>
```

ros2_control_demos Example: "Industrial robot with integrated sensor"

Modelling complex hardware – monolithic components



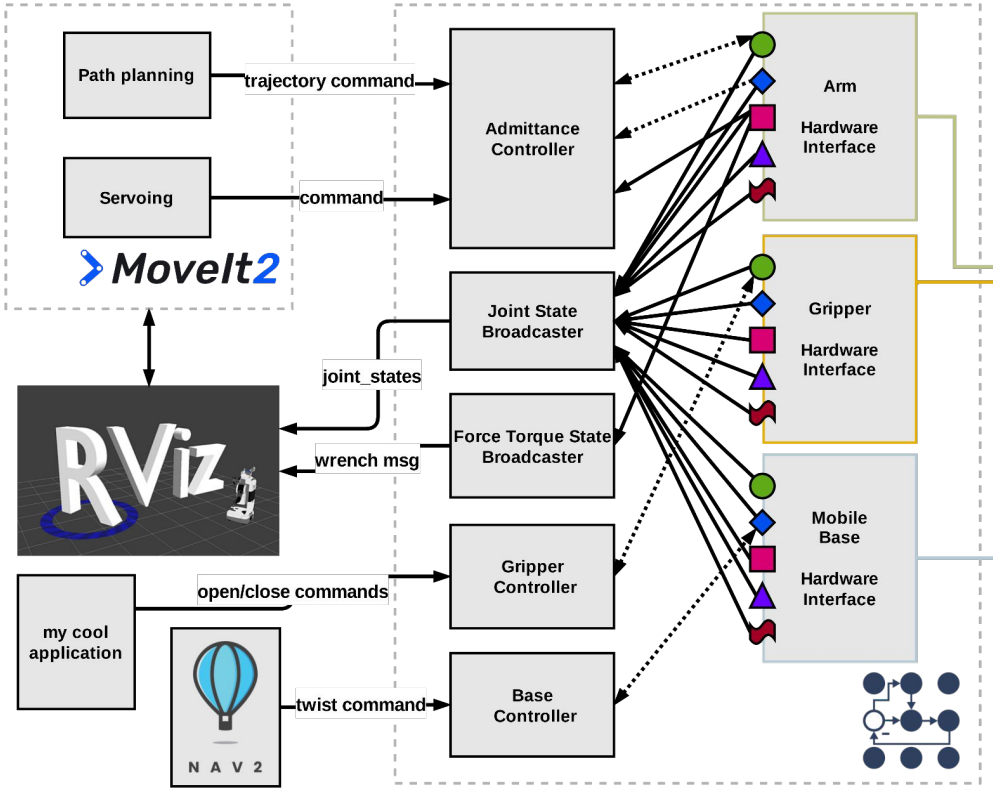

```
<ros2_control name="MobileManipulator" type="system">
  <hardware>
    <plugin>ros2_contro_robot/Full_Mobile_Manipulator</plugin>
    ...
  </hardware>
  <joint name="wheel1_joint">
    <command_interface name="velocity"/>
    <state_interface name="position"/>
    <state_interface name="velocity"/>
    <state_interface name="status"/>
  </joint>
  ...
  <joint name="joint1">
    <command_interface name="position"/>
    <state_interface name="position"/>
    <state_interface name="velocity"/>
    <state_interface name="status"/>
  </joint>
  ...
  <joint name="gripper_joint">
    <command_interface name="position"/>
    <state_interface name="position"/>
    <state_interface name="velocity"/>
    <state_interface name="status"/>
  </joint>
  ...
</ros2_control>
```

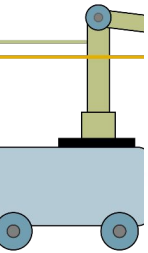
ros2_control_demos Example: "Robots with multiple interfaces"

Presentation outline

1. Present outline
2. Short history and basic concepts
3. How is ros2_control different from ros_control?
4. I want to use ros2_control! Where to start?
    - About robot description, hardware drivers and controllers
5. But my hardware is complex…
6. Panic! My controllers are getting too convoluted… ← We are here!
7. And what if I have multiple robots?
8. Resources and persons behind ros2_control

Let’s check an example






```yaml
controller_manager:
  update_rate: 500  # Hz

  admittance_controller:
    type: ros2_control_mm/AdmittanceController

  forward_position_controller:
    type: position_controllers/JointGroupPositionController

  joint_state_broadcaster:
    type: joint_state_broadcaster/JointStateBroadcaster

  force_torque_sensor_broadcaster:
    type: force_torque_sensor_broadcaster/ForceTorqueStateBroadcaster

  gripper_controller:
    type: position_controllers/GripperActionController

  base_controller:
    type: ros2_control_mm/BaseControllers
```

```yaml
admittance_controller:
  joints:
    - joint1
    - ...
  command_interfaces:
    - position
  state_interfaces:
    - position
    - velocity
```

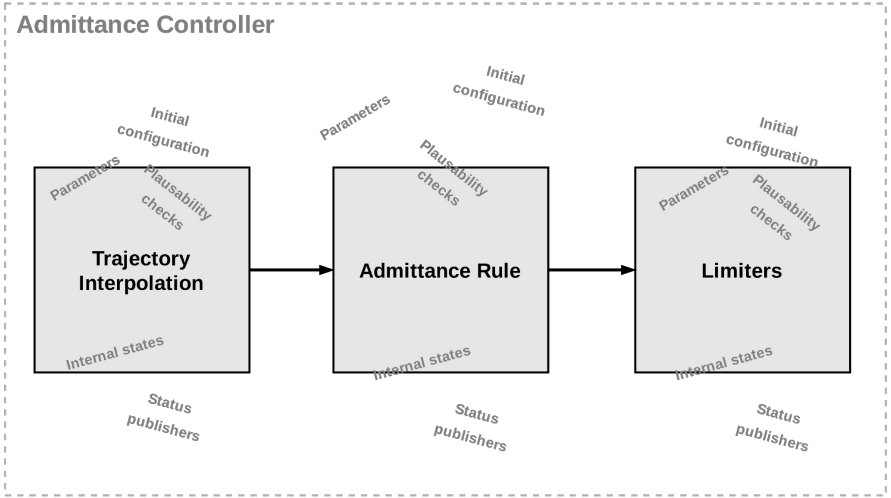
```yaml
forward_position_controller:
  joints:
    - joint1
    - ...
```

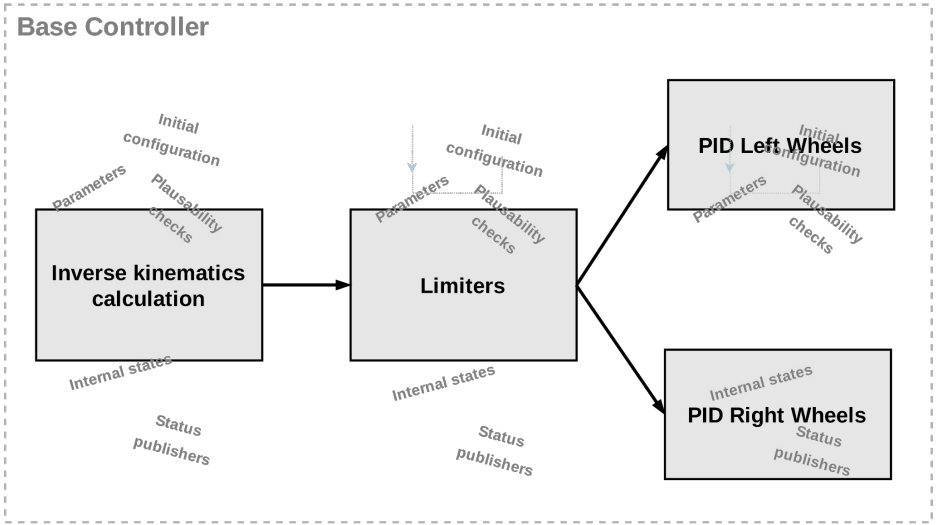
```yaml
force_torque_sensor_broadcaster:
  sensor_name: tcp_fts_sensor
  frame_id: tool0
  topic_name: ft_data
```

```yaml
gripper_controller:
  joints:
    - gripper_joint
  command_interface: position
```

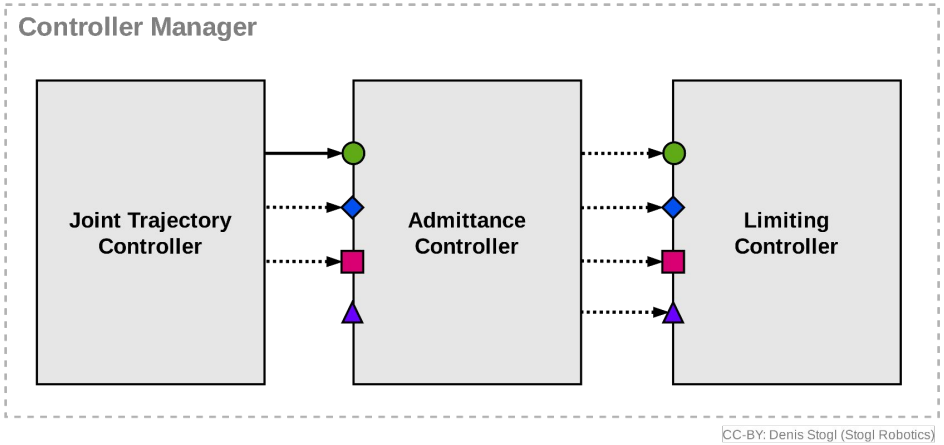
```yaml
base_controller:
  left_wheel_names:
    - left_wheel_1
    ...
```

This can end-up in convoluted and complex controllers…





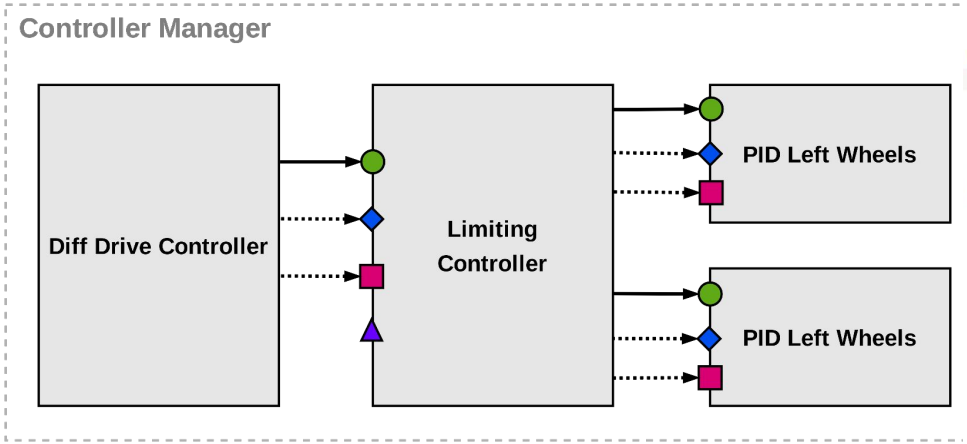
Using controller-chaining…




```yaml
controller_manager:
  update_rate: 500  # Hz

  joint_trajectory_controller:
    type: joint_trajectory_controller/JointTrajectoryController

  admittance_controller:
    type: admittance_controller/AdmittanceController

  limiting_controller:
    type: limiting_controllers/JointLimitingController


joint_trajectory_controller:
  joints:
    - joint1
    - ...
  command_joints:
    - admittance_controller/joint1
    - ...
  command_interfaces:
    - position
  state_interfaces:
    - position
    - velocity

# export reference interfaces: "<controller_name>/<joint_name>/<interface_name>"
admittance_controller:
  joints:
    - joint1
    - ...
  command_joints:
    - limiting_controller/joint1
    - ...
  command_interfaces:
    - position
  state_interfaces:
    - position
    - velocity


# export reference interfaces: "<controller_name>/<joint_name>/<interface_name>"
limiting_controller:
  joints:
    - joint1
    - ...
  interfaces:
    - position
limiting_controller:
  joints:
    - joint1
    - ...
  interfaces:
    - position
```

https://github.com/ros-controls/ros2_control_demos/pull/162
https://github.com/ros-controls/ros2_control/pull/667

Using controller-chaining…




```yaml
controller_manager:
  update_rate: 500  # Hz

  diff_drive_controller:
    type: diff_drive_controller/DiffDriveController

  limiting_controller:
    type: limiting_controllers/JointLimitingController

  pid_left_wheels:
    type: pid_controllers/PIDController

  pid_right_wheels:
    type: pid_controllers/PIDController


diff_drive_controller:
  left_weel_names:
    - left_wheel_1
    ...

# export reference interfaces: "<controller_name>/<joint_name>/<interface_name>"
limiting_controller:
  joints:
    - left_wheel_1
    - ...
  command_joints:
    - pid_left_wheels/joint1/velocity
    - ...
    - pid_right_wheels/joint1/velocity
    - ...
  interfaces:
    - velocity


# export reference interfaces: "<controller_name>/<joint_name>/<interface_name>"
pid_left_wheels:
  joints:
    - left_wheel_1
    ...


# export reference interfaces: "<controller_name>/<joint_name>/<interface_name>"
pid_right_wheels:
  joints:
    - right_wheel_1
    ...
```

https://github.com/ros-controls/ros2_control_demos/pull/162
https://github.com/ros-controls/ros2_control/pull/667

Presentation outline

1. Present outline
2. Short history and basic concepts
3. How is ros2_control different from ros_control?
4. I want to use ros2_control! Where to start?
    - About robot description, hardware drivers and controllers
5. But my hardware is complex…
6. Panic! My controllers are getting too convoluted…
7. And what if I have multiple robots?
8. Resources and persons behind ros2_control

We are here!

Multiple controller managers

1. Using one controller manager – when tight synchronization is needed
2. Using multiple controller managers – when robots are mainly independent

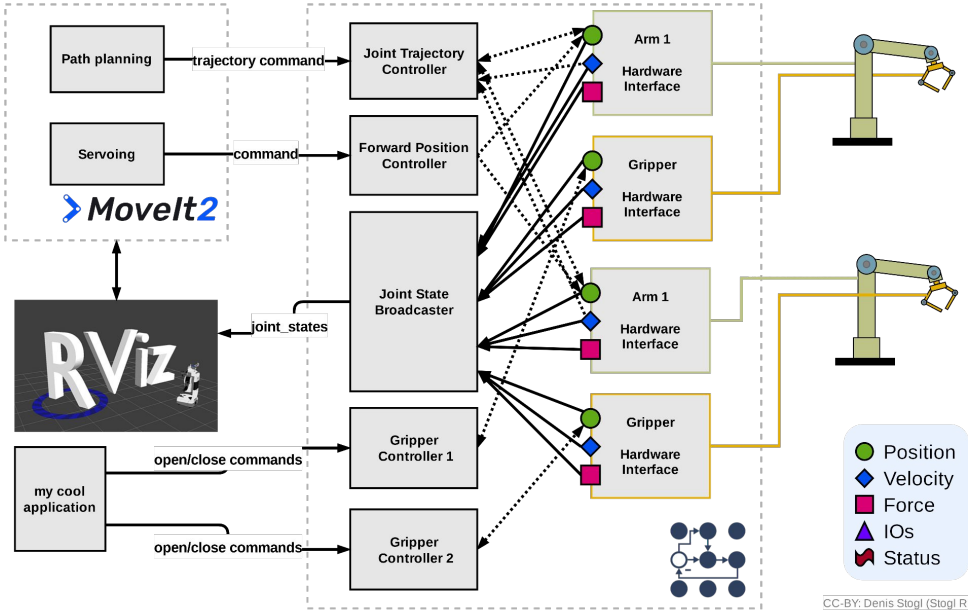
Multiple controller managers

- Robots are mainly independent—swarm robotics
- Uses:
  - Separate namespaces for *ros2_control_nodes* (controller_manager)
  - Prefixes for joints (hardware interface name also recommended)

Scenario showcase: “Using multiple controller managers on the same machine”
https://github.com/ros-controls/ros2_control_demos/pull/170

One controller manager

- Tight coupling and synchronization between robots needed, e.g., dual-arm
- Prefixes for hardware interfaces and joints
- Controllers for one or both

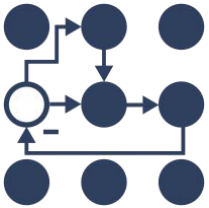


Presentation outline

1. Present outline
2. Short history and basic concepts
3. How is ros2_control different from ros_control?
4. I want to use ros2_control! Where to start?
    - About robot description, hardware drivers and controllers
5. But my hardware is complex…
6. Panic! My controllers are getting too convoluted…
7. And what if I have multiple robots?
8. Resources and persons behind ros2_control

We are here!

References



- ros_control paper in the Journal of Open Source Software

- ros2_control presentations
  - https://control.ros.org/master/doc/resources/resources.html

- ros2_control resources
  - https://ros-controls.github.io/control.ros.org/
  - https://github.com/ros-controls/ros2_control
  - https://github.com/ros-controls/ros2_controllers
  - https://github.com/ros-controls/ros2_control_demos
  - https://github.com/ros-controls/roadmap/blob/master/documentation_resources.md

Thank you!

Bence Magyar, Denis Štogl, 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, Michael Wiznitzer, Paul Gesel, Mateus Amarante, Auguste Bourgois and many more!