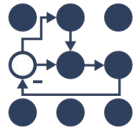
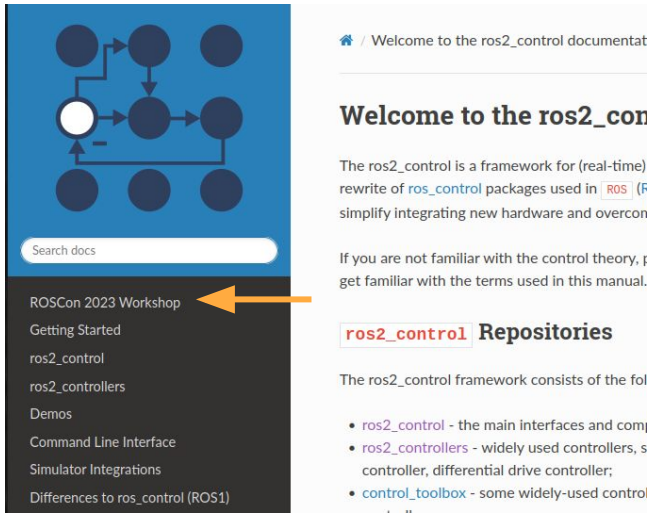
Preparations



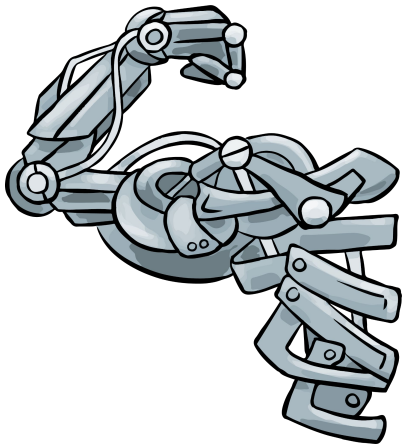
Open https://control.ros.org/master/doc/roscon2023_workshop.html

Or https://control.ros.org




```
docker pull bmagyar/roscon2023_workshop:latest
```

```
mkdir -p ws/src

cd ws/src

git clone
https://github.com/ros-controls/roscon2023_control_workshop

vcs import --input
roscon2023_control_workshop/roscon2023_control_workshop.repos
.
```



ros2_control on Steroids

$whoarewe

Bence Magyar – [Bent’seh]

- PhD in Robotics
- Principal Software Engineer at Locus Robotics
- ros_control and ros2_control maintainer

Denis Štogl – [Denis]

- PhD, Control Engineer and Roboticist
- Robotics Consultant at Stogl Robotics Consulting
- ros2_control maintainer

History

pr2_controller_manager

(pr2_mechanism)

2009

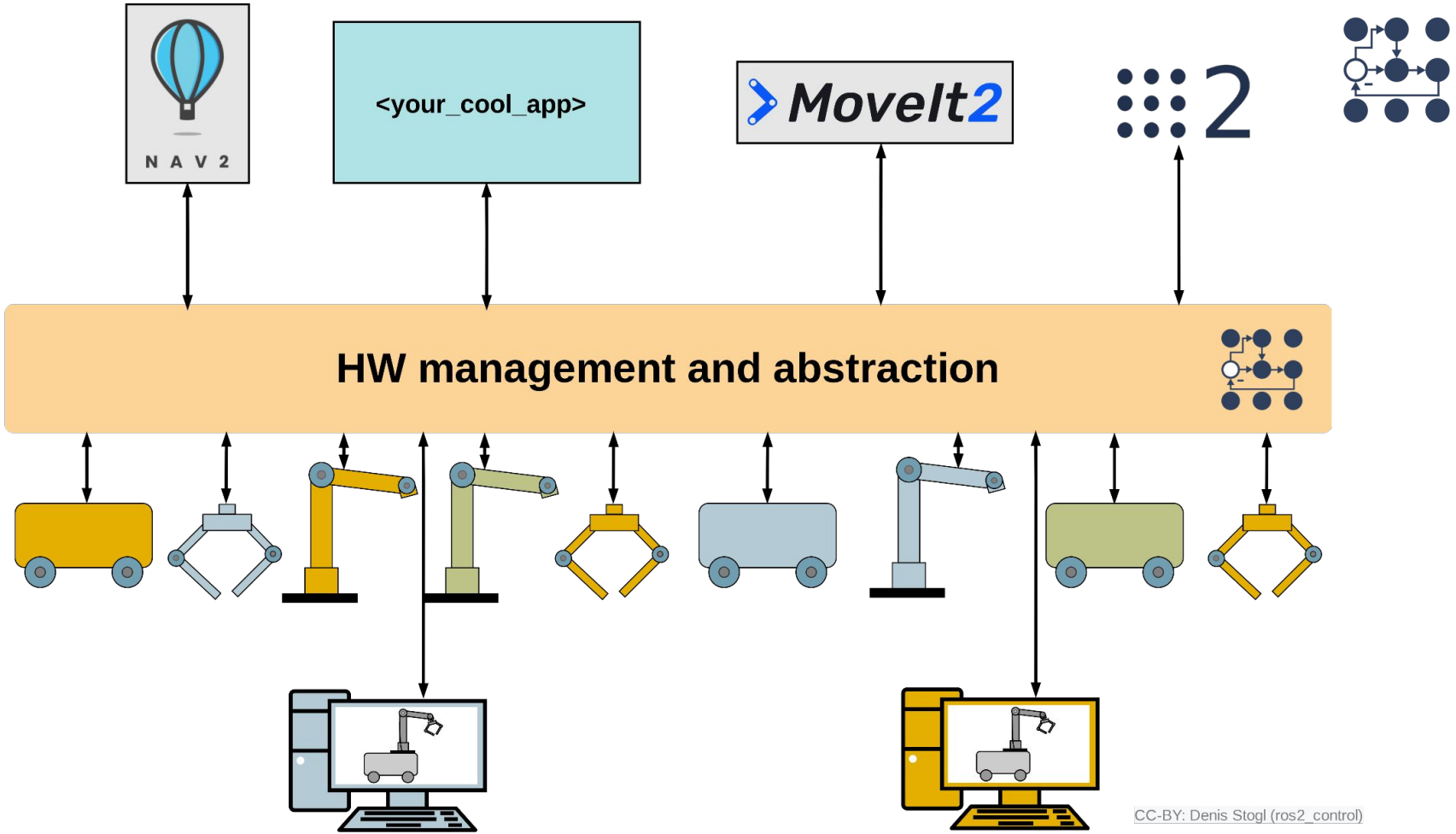
ros_control

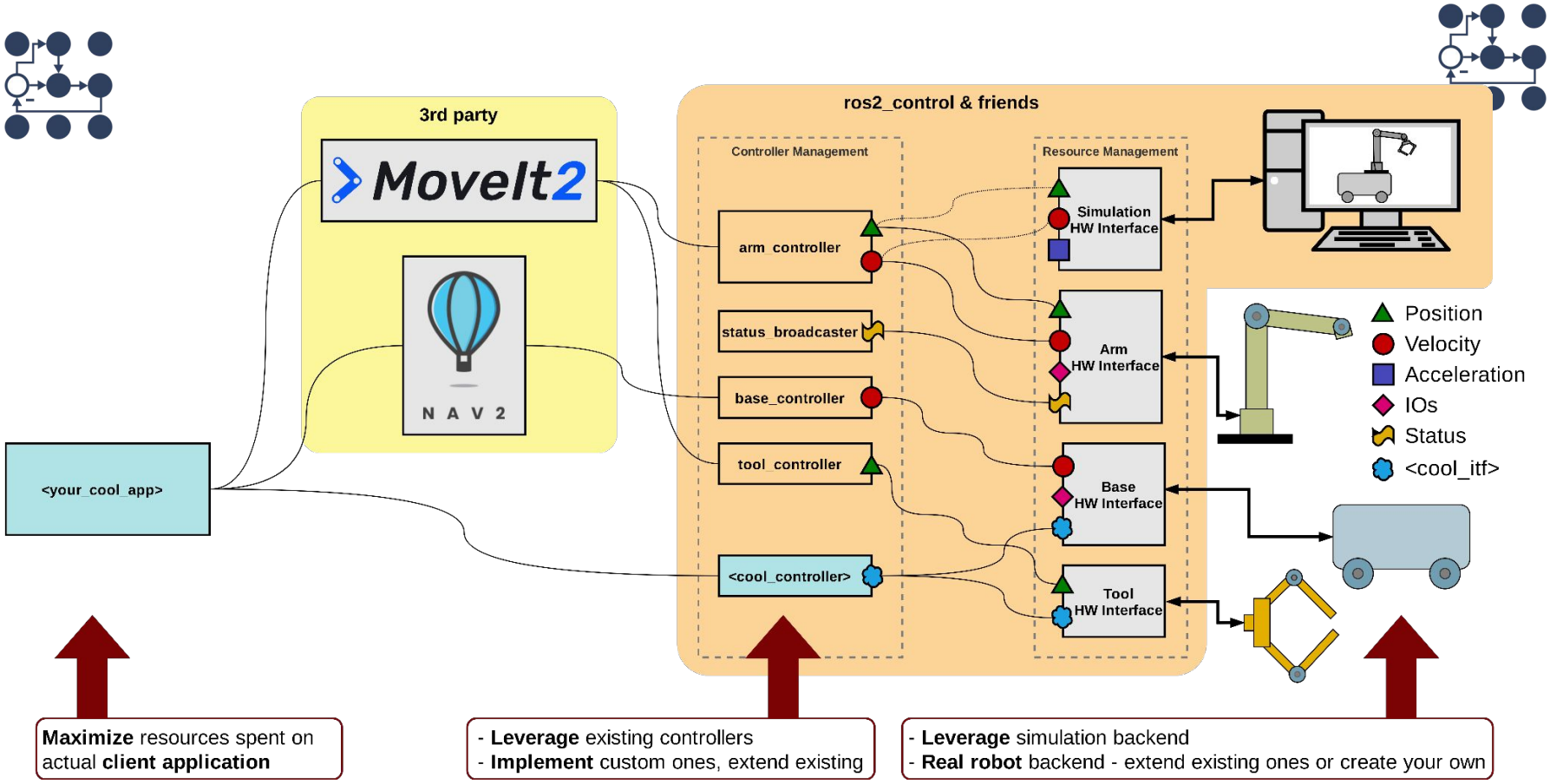
2012/2017

ros2_control

2017/2023

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



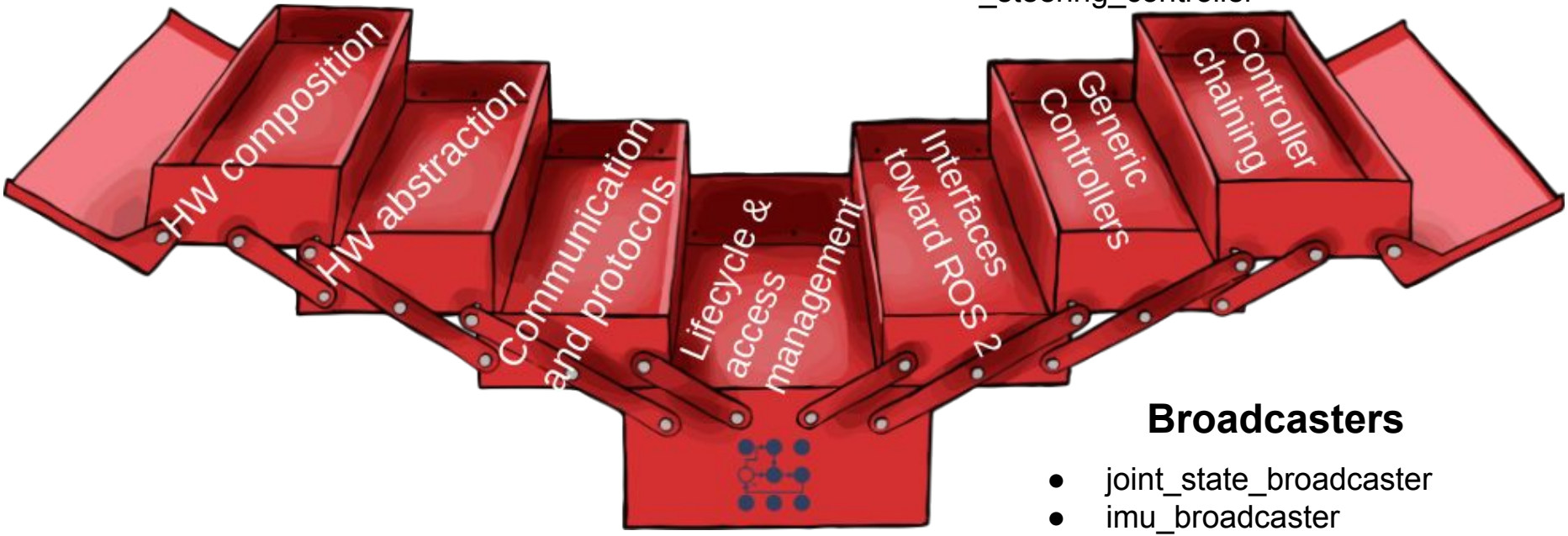

3rd party
MoveIt2
NAV 2
<your_cool_app>
ros2_control & friends
Controller Management
Resource Management
arm_controller
status_broadcaster
base_controller
tool_controller
<cool_controller>
Simulation HW Interface
Arm HW Interface
Base HW Interface
Tool HW Interface
Position
Velocity
Acceleration
IOs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

Hardware components

- SystemComponent
- SensorComponent
- ActuatorComponent

Controllers

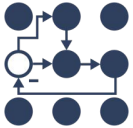
- joint_trajectory_controller
- diff_drive_controller
- forwarding controllers
- gripper_controllers
- *_steering_controller



Broadcasters

- joint_state_broadcaster
- imu_broadcaster
- force_torque_broadcaster

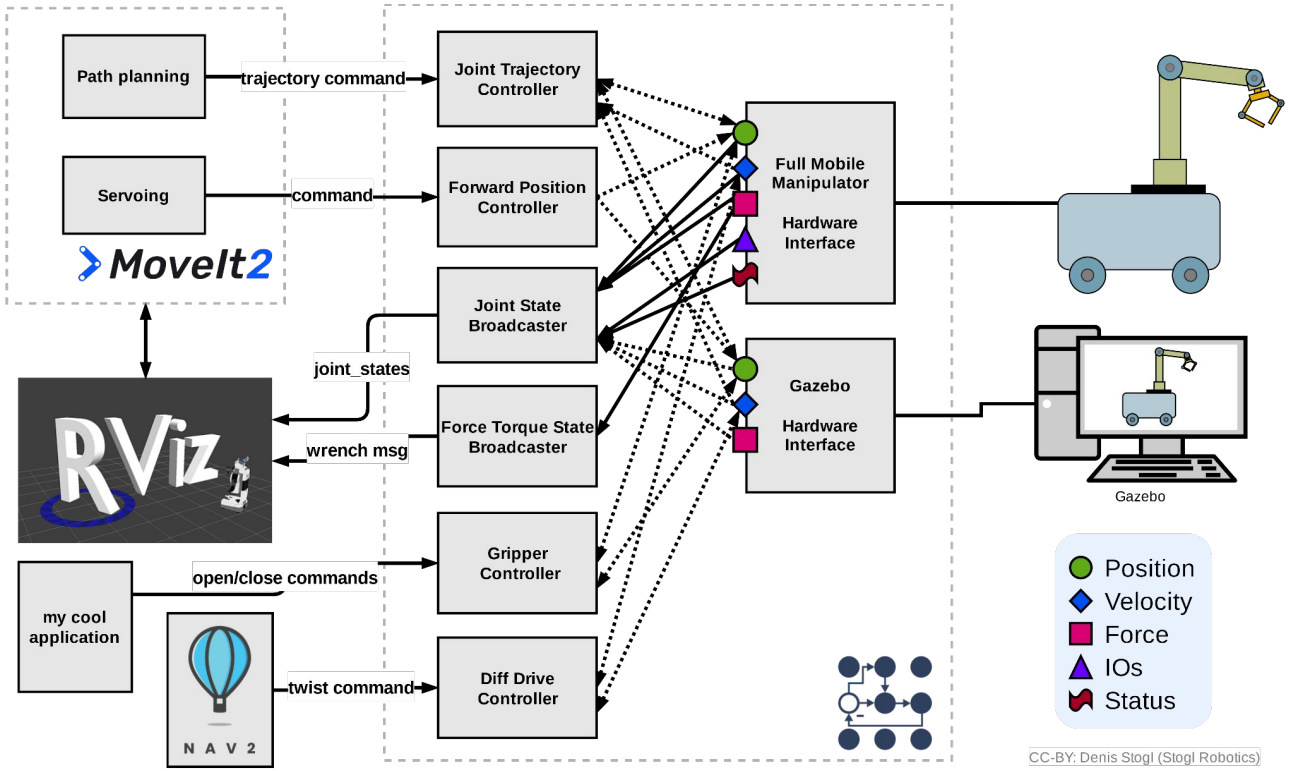
Outline REMOVE IT EVENTUALLY



- Introduction and quick rundown [15 minutes]
- Hardware modularization [10 pres. + 10 + 10 minutes]
- Controller chaining [15 pres. + 60 minutes]
- BREAK [10 minutes] — 2 hours mark
- ~~Parameter injection [15 minutes]~~
- Multi-robot architectures [15 pres. + 45 minutes]
- Debugging of complex systems [45 minutes]

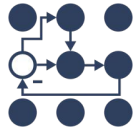
Configuring standard controllers

```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
    ...
```



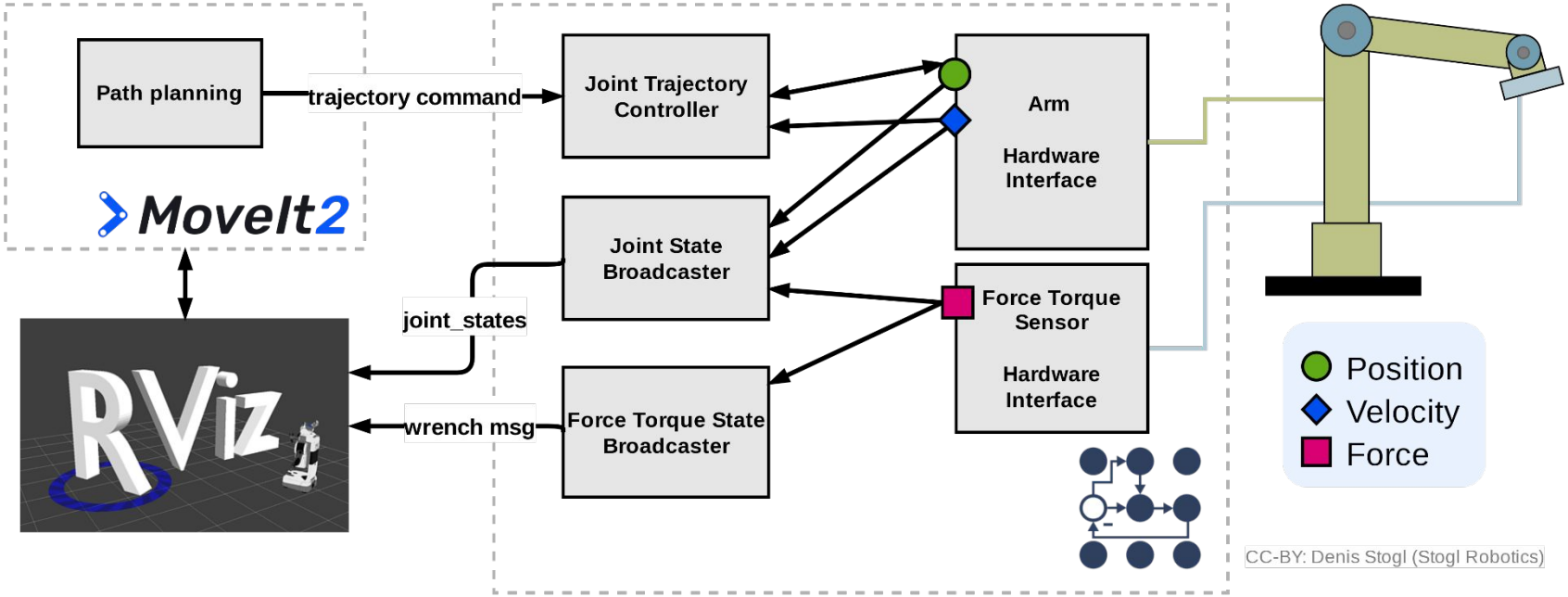
URDF extension with <ros2_control>-tag

```
<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>
```



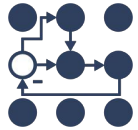
```
<ros2_control name="robot" type="system">

  <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>
```

Hardware modularization – Arm + Sensor




```
git checkout hardware-modularization-ex5-start   (from the "src" folder run: rosdep install -y -i
--from-paths .)
cb && s
ros2 launch ros2_control_demo_example_5 rrbot_system_with_external_sensor.launch.py

ros2 launch ros2_control_demo_example_5 test_forward_position_controller.launch.py
```

Hardware modularization – Arm + Sensor



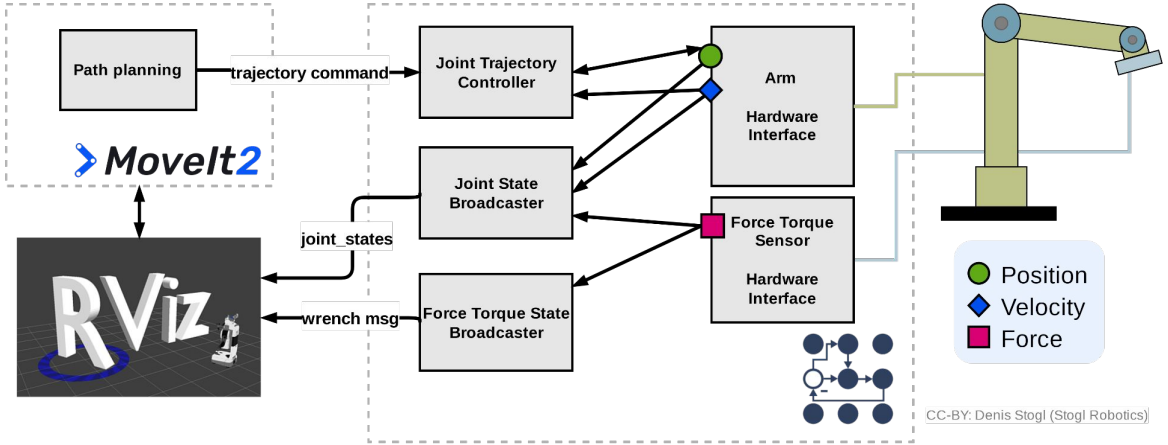
```
git checkout hardware-modularization-ex5-start    (from the “src” folder
run: rosdep install -y -i  --from-paths .)
cb && s
ros2 launch ros2_control_demo_example_5
rrbot_system_with_external_sensor.launch.py

ros2 launch ros2_control_demo_example_5
test_forward_position_controller.launch.py
```

Open 2 more terminals in `tmux` by using `CTRL+B` and `"` and `CTRL+B` and `%`.

You can navigate in tmux using `CTRL+B` and `ARROW` keys.

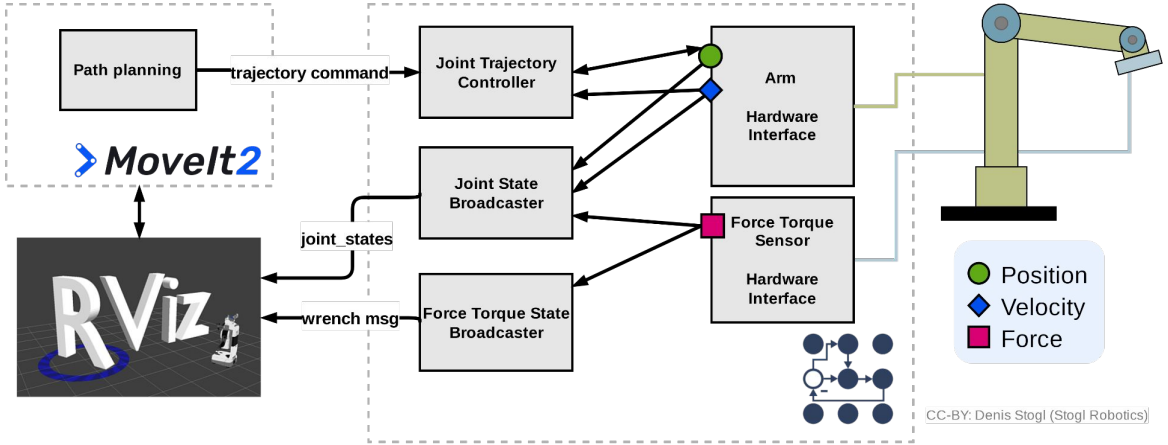
Task: Hardware modularization – Sensor in Arm

- Example 5 from ros2_control_demos repository
- Task:
  - Add sensor to be started together with the robot



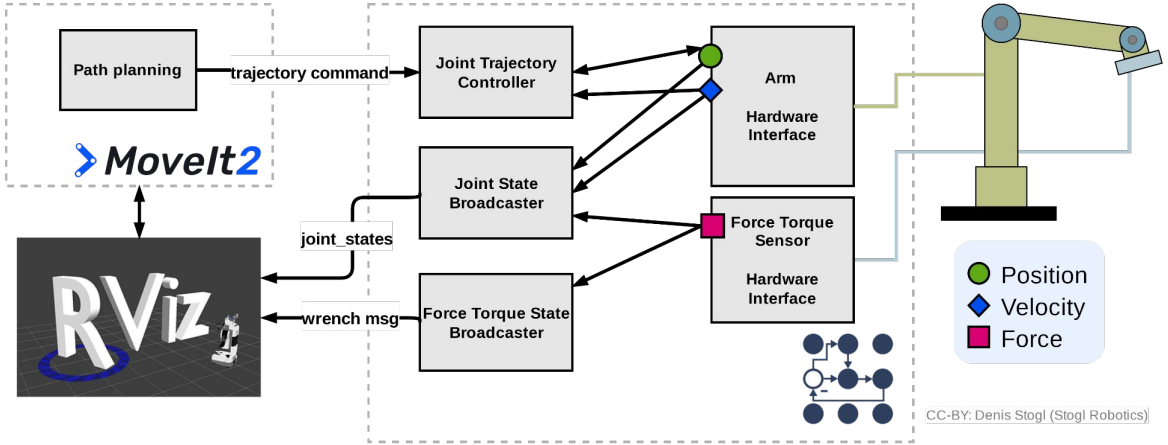
Task: Hardware modularization – Sensor in Arm

- Example 5 from ros2_control_demos repository
- Task:
  - Add sensor to be started together with the robot
    - Check plugins exported in the package
    - Update URDF of the setup

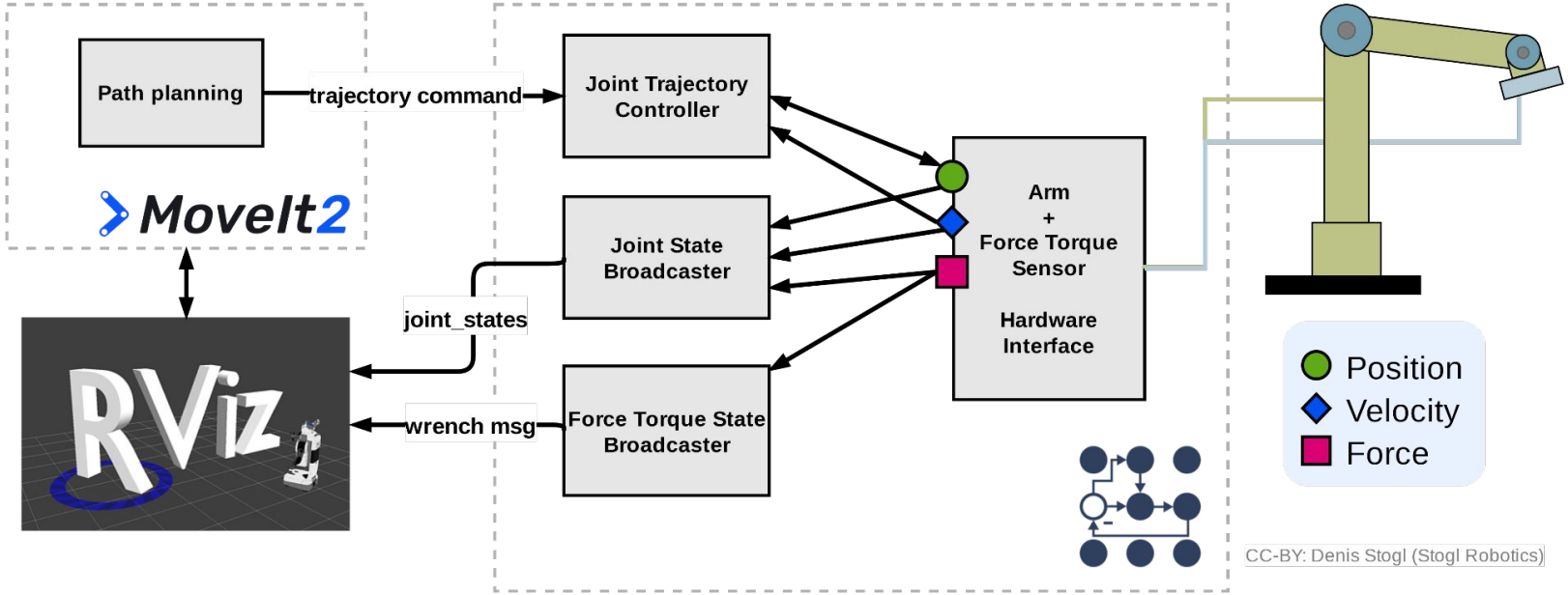


Task: Hardware modularization – Sensor in Arm

- `git checkout hardware-modularization-ex5-solution`
- `Check:`
  - `$(find ros2_control_demo_example_5)/description/urdf/rrbot_system_with_external_sensor.urdf.xacro`
  - `$(find ros2_control_demo_example_5)/ros2_control/external_rrbot_force_torque_sensor.ros2_control.xacro`



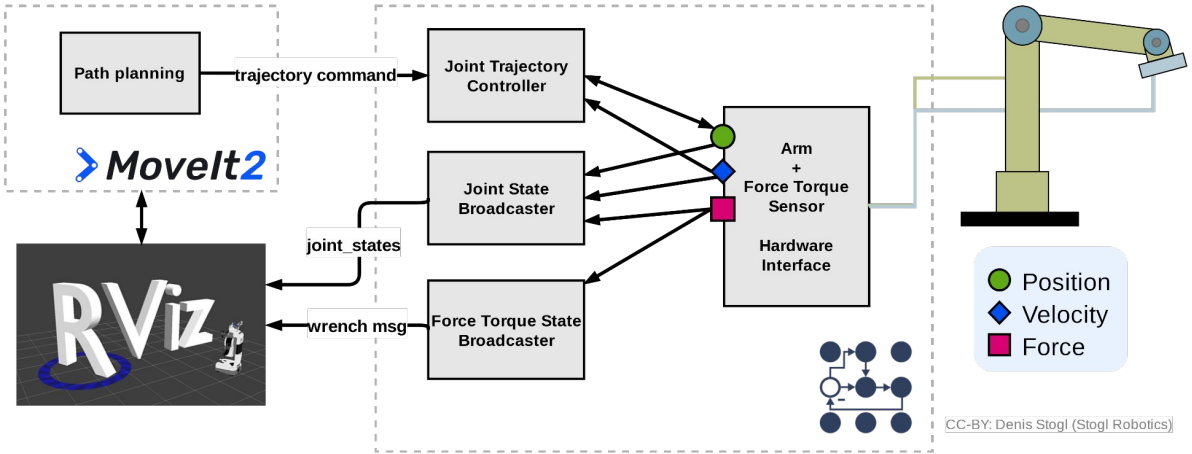
Hardware modularization – Sensor in Arm



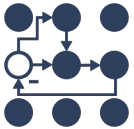

```
git checkout rolling

ros2 launch ros2_control_demo_example_4 rrbot_system_with_sensor.launch.py

ros2 launch ros2_control_demo_example_4 test_forward_position_controller.launch.py
```

Task: Hardware modularization – Sensor in Arm

- Example 4 from ros2_control_demos repository
- Task:
  - Start Mock Hardware instead of the “real” hardware and use mocked sensor values
  - Mock Sensor data:
    - Use: “ros2 control list_hardware_interfaces” CLI
    - Check rrbot_system_with_sensor.urdf.xacro <ros2_control> tag
    - Check launch file



Solution: Hardware modularization – Sensor in Arm



- `git checkout hardware-modularization-ex4-solution`

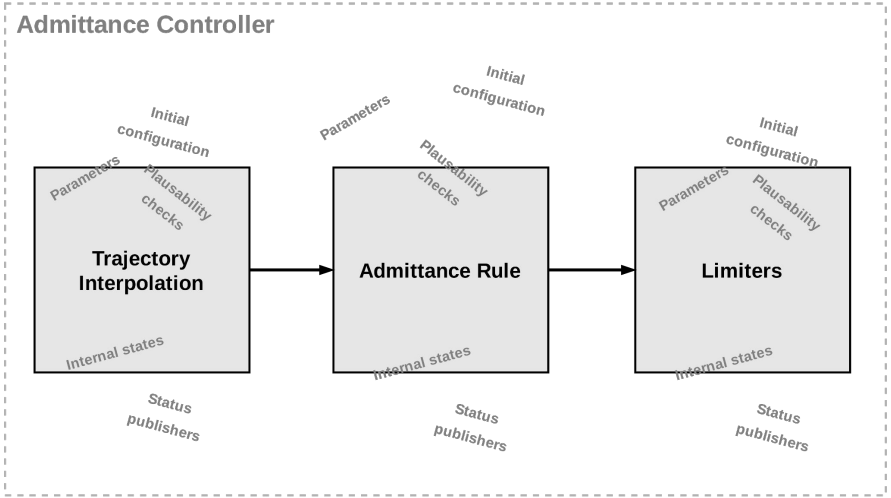
Change rrbot_system with sensor.launch.py and set values use_mock_hardware and mock_sensor_commands to “true”

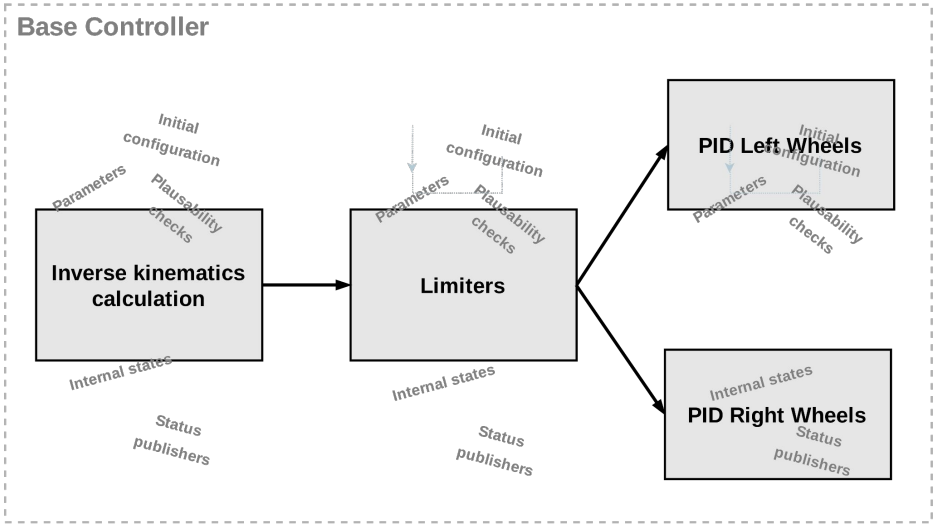
Check the updated controllers file.

Publish:

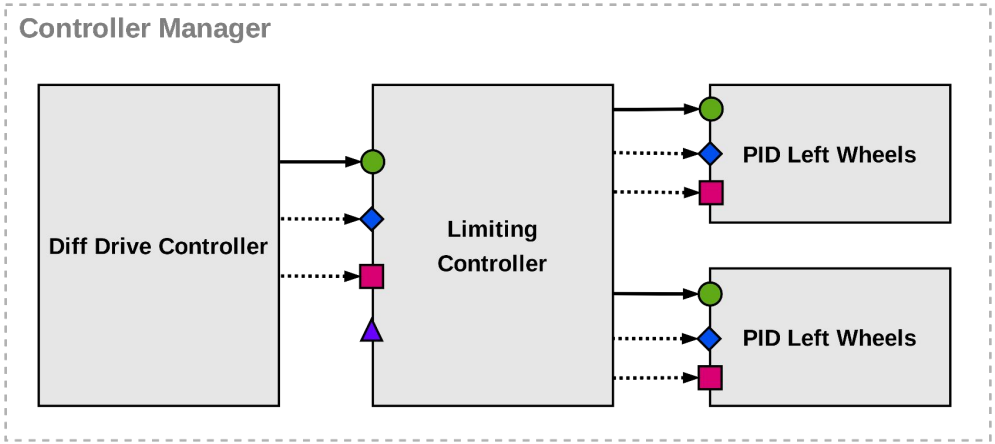
```
ros2 topic pub /mock_sensor_commands_forward_controller/command std_msgs/msg/Float64Array {0.7, 3.4}
```

This can end-up pretty convoluted…





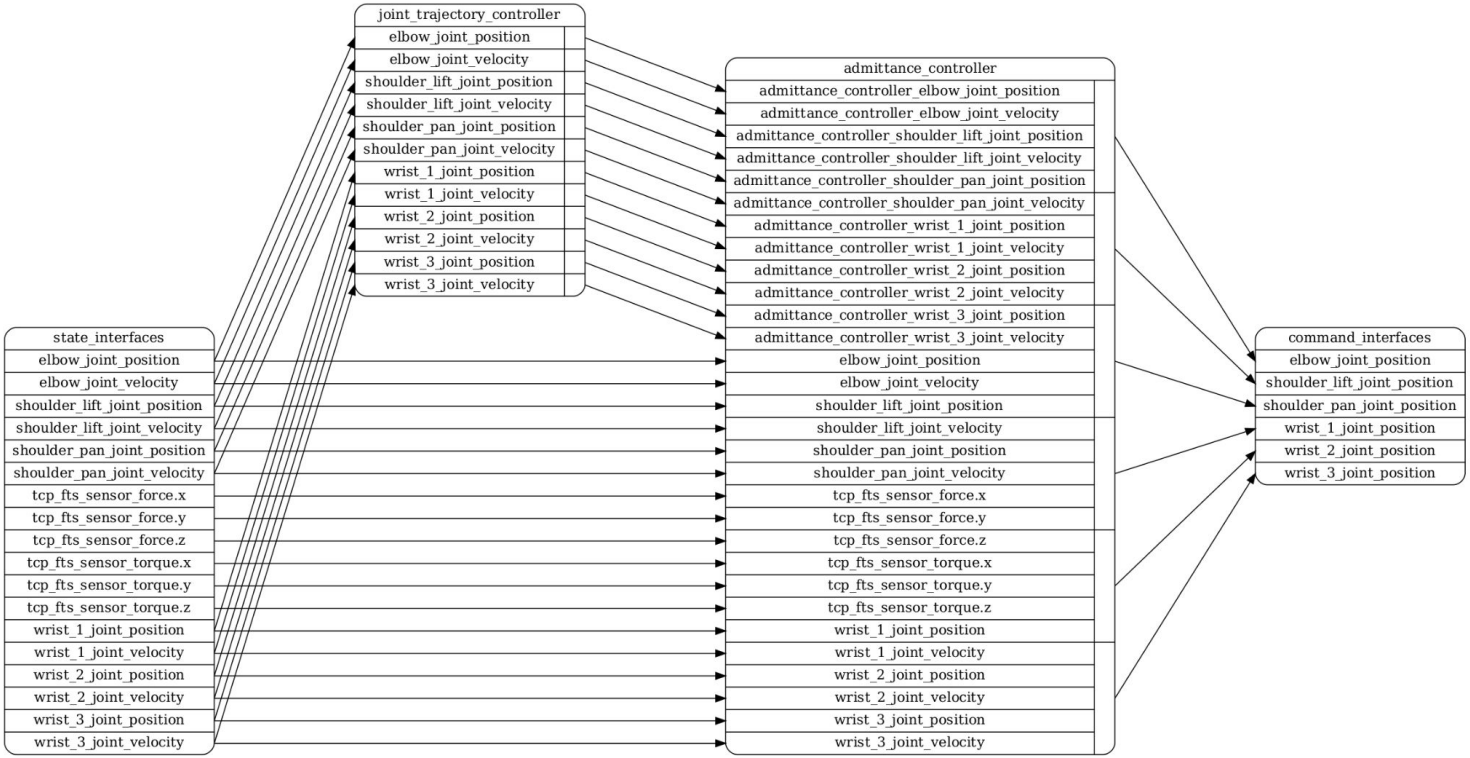
Controller chaining to the rescue!



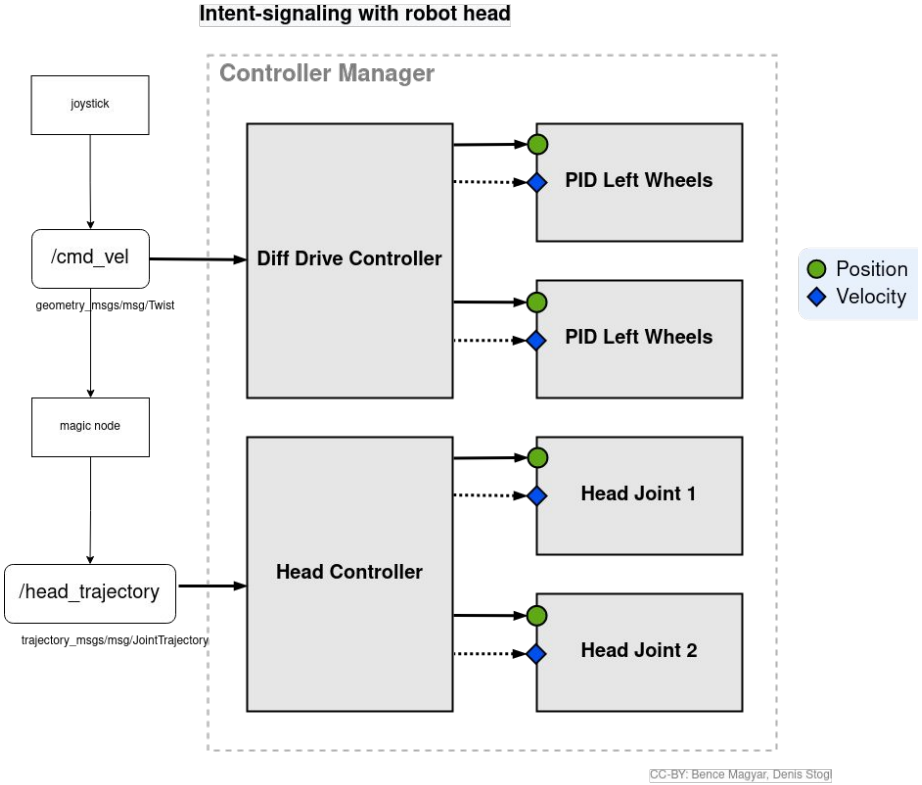

```yaml
controller_manager:
  update_rate: 500  # Hz

  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
    ...
```

CLI extra

$ ros2 control view_controller_chains



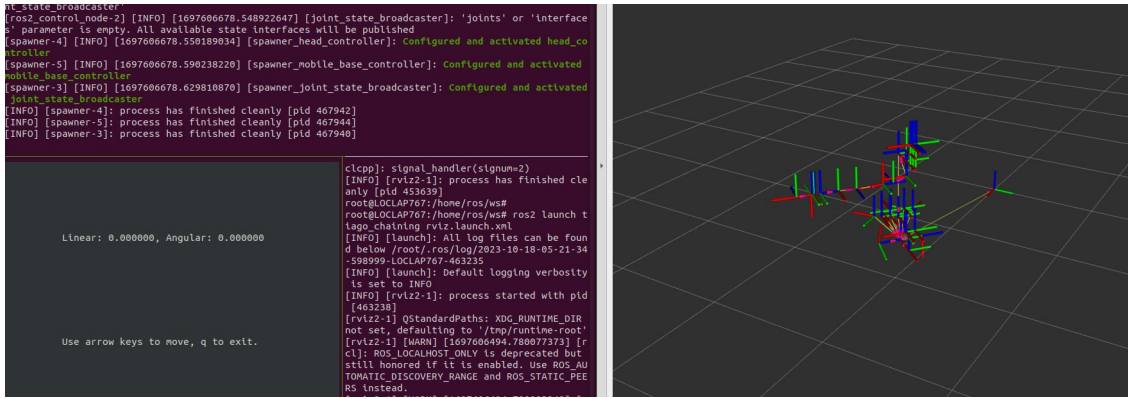
Controller chaining task



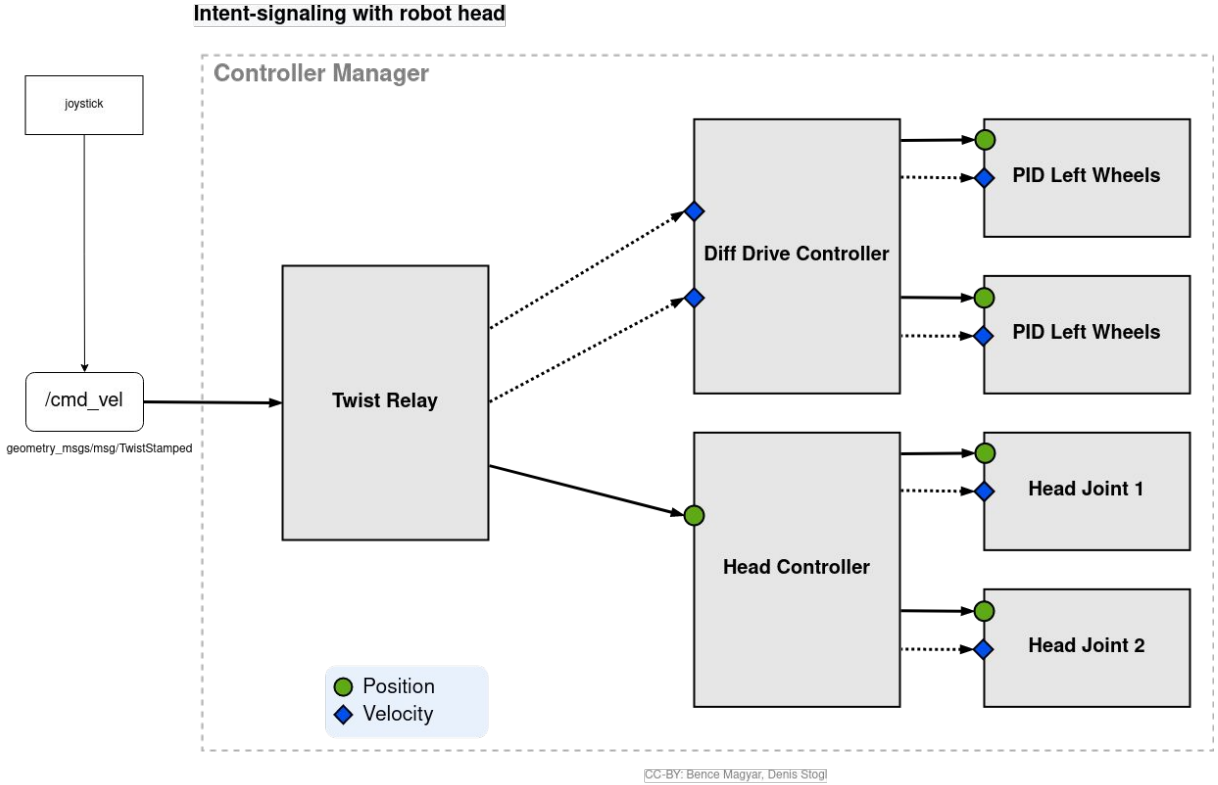
Controller chaining task - warmup

- git checkout chaining
- cb && s

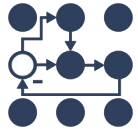
- ros2 launch tiago_chaining tiago_warmup.launch.xml
- ros2 control list_hardware_interfaces
- ros2 topic list
- ros2 node info /mobile_base_controller
- ros2 launch tiago_chaining rviz.launch.xml
- ros2 run key_teleop key_teleop key_vel:=/mobile_base_controller/cmd_vel_unstamped



Controller chaining task

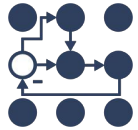


Meet twist_relay_controller/RelayController



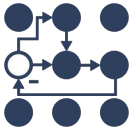
```cpp
#ifndef TWIST_RELAY_CONTROLLER__TWIST_RELAY_CONTROLLER_HPP_
#define TWIST_RELAY_CONTROLLER__TWIST_RELAY_CONTROLLER_HPP_

#include <memory>

#include <realtime_tools/realtime_box.h>
#include <controller_interface/controller_interface.hpp>
#include <geometry_msgs/msg/twist_stamped.hpp>

// auto-generated by generate_parameter_library
#include "twist_relay_controller_parameters.hpp"

namespace twist_relay_controller
{
using Twist = geometry_msgs::msg::TwistStamped;

class RelayController : public controller_interface::ControllerInterface
{
public:
  RelayController();

  controller_interface::InterfaceConfiguration command_interface_configuration() const override;

  controller_interface::InterfaceConfiguration state_interface_configuration() const override;

  controller_interface::return_type update(
    const rclcpp::Time & time, const rclcpp::Duration & period) override;

  controller_interface::CallbackReturn on_init() override;

  controller_interface::CallbackReturn on_configure(
    const rclcpp_lifecycle::State & previous_state) override;

protected:
  rclcpp::Subscription<Twist>::SharedPtr twist_subscriber_ = nullptr;
  realtime_tools::RealtimeBox<std::shared_ptr<Twist>> last_msg_ptr_{nullptr};

  std::shared_ptr<ParamListener> param_listener_;
  Params params_;
};
}  // namespace twist_relay_controller

#endif  // TWIST_RELAY_CONTROLLER__TWIST_RELAY_CONTROLLER_HPP_
```

```xml
<library path="twist_relay_controller">
  <class name="twist_relay_controller/RelayController"
         type="twist_relay_controller::RelayController"
         base_class_type="controller_interface::ControllerInterface">
  <description>
    Controller relaying parts of a twist message
  </description>
  </class>
</library>
```



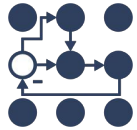
class DiffDriveController : public controller_interface::**ControllerInterface**

Meet the new diff_drive_controller



```cpp
namespace diff_drive_controller
{
class DiffDriveController : public controller_interface::ChainableControllerInterface
{
  using Twist = geometry_msgs::msg::TwistStamped;

public:
  DIFF_DRIVE_CONTROLLER_PUBLIC
  DiffDriveController();

  DIFF_DRIVE_CONTROLLER_PUBLIC
  controller_interface::InterfaceConfiguration command_interface_configuration() const override;

  DIFF_DRIVE_CONTROLLER_PUBLIC
  controller_interface::InterfaceConfiguration state_interface_configuration() const override;

  DIFF_DRIVE_CONTROLLER_PUBLIC controller_interface::return_type
  update_reference_from_subscribers(
    const rclcpp::Time & time, const rclcpp::Duration & period) override;

  DIFF_DRIVE_CONTROLLER_PUBLIC controller_interface::return_type
  update_and_write_commands(const rclcpp::Time & time, const rclcpp::Duration & period) override;
```

Meet the new diff_drive_controller



```
std::vector<hardware_interface::CommandInterface>
DiffDriveController::on_export_reference_interfaces()
{
  const int nr_ref_itfs = 2;
  reference_interfaces_.resize(nr_ref_itfs, std::numeric_limits<double>::quiet_NaN());
  std::vector<hardware_interface::CommandInterface> reference_interfaces;
  reference_interfaces.reserve(nr_ref_itfs);

  reference_interfaces.push_back(hardware_interface::CommandInterface(
    get_node()->get_name(), std::string("linear/") + hardware_interface::HW_IF_VELOCITY,
    &reference_interfaces_[0]));

  reference_interfaces.push_back(hardware_interface::CommandInterface(
    get_node()->get_name(), std::string("angular/") + hardware_interface::HW_IF_VELOCITY,
    &reference_interfaces_[1]));

  return reference_interfaces;
}
```

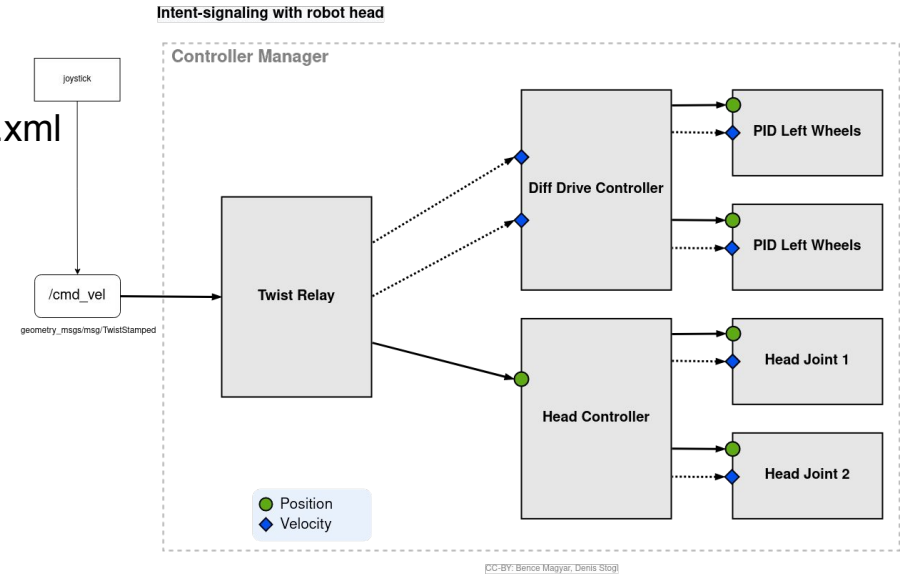
Controller chaining

- git checkout chaining
- cb && s

- ros2 launch tiago_chaining tiago_chaining.launch.xml
- ros2 control list_hardware_interfaces
- ros2 control list_controllers

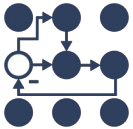
What's missing? Check:

- tiago_chaining/launch/tiago_chaining.launch.xml
- tiago_chaining/config/controllers.yaml
- twist_relay code for TODO notes

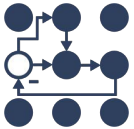


- ros2 launch tiago_chaining rviz.launch.xml
- ros2 run key_teleop key_teleop key_vel:=/cmd_vel -ros-args -p "twist_stamped_enabled:=True"

Happy?



Chainable Twist Relayer!



- ros2 launch tiago_chaining tiago_chaining.launch.xml
- ros2 control list_hardware_interfaces

Todo list:

- twist_relay_controller.hpp /.cpp / controller_plugins.xml
  - ControllerInterface -> ChainableControllerInterface
- Add / implement
  - update_reference_from_subscribers(...)
  - update_and_write_commands(...)
  - on_set_chained_mode(...)
  - on_export_reference_interfaces(...)

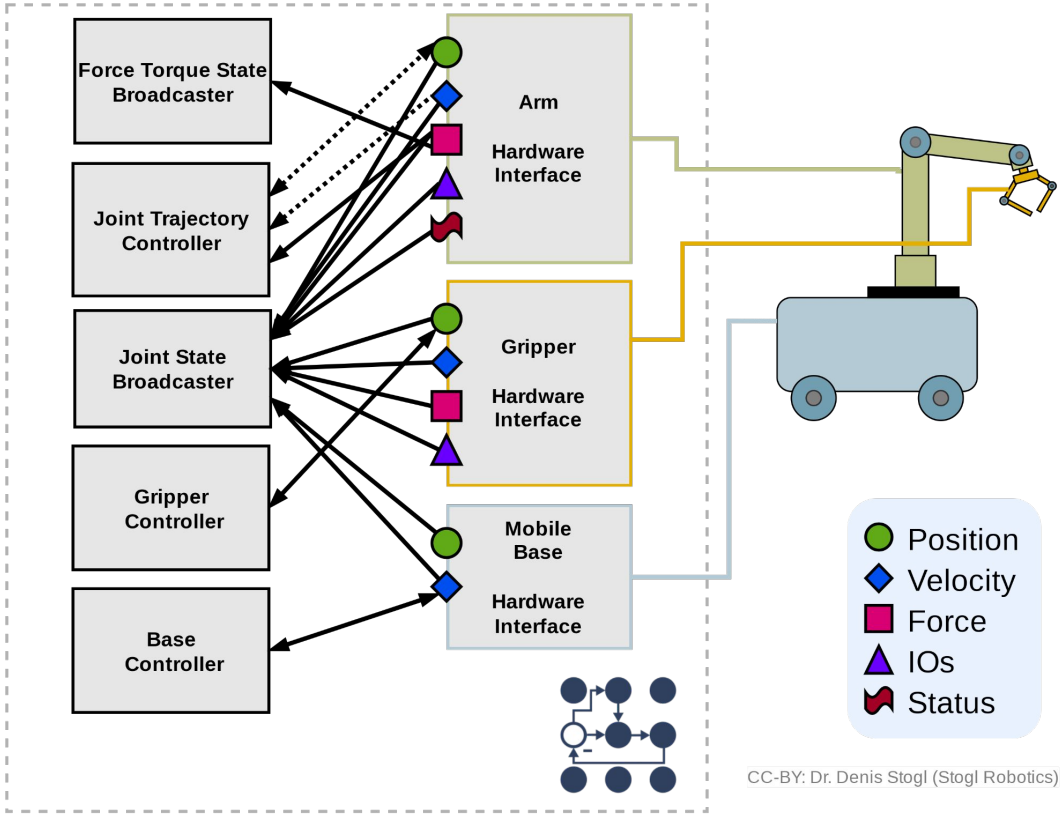
Multi-robot architectures

- `git checkout multi-robot-architectures-begin`

- Inspection of Hercules example:

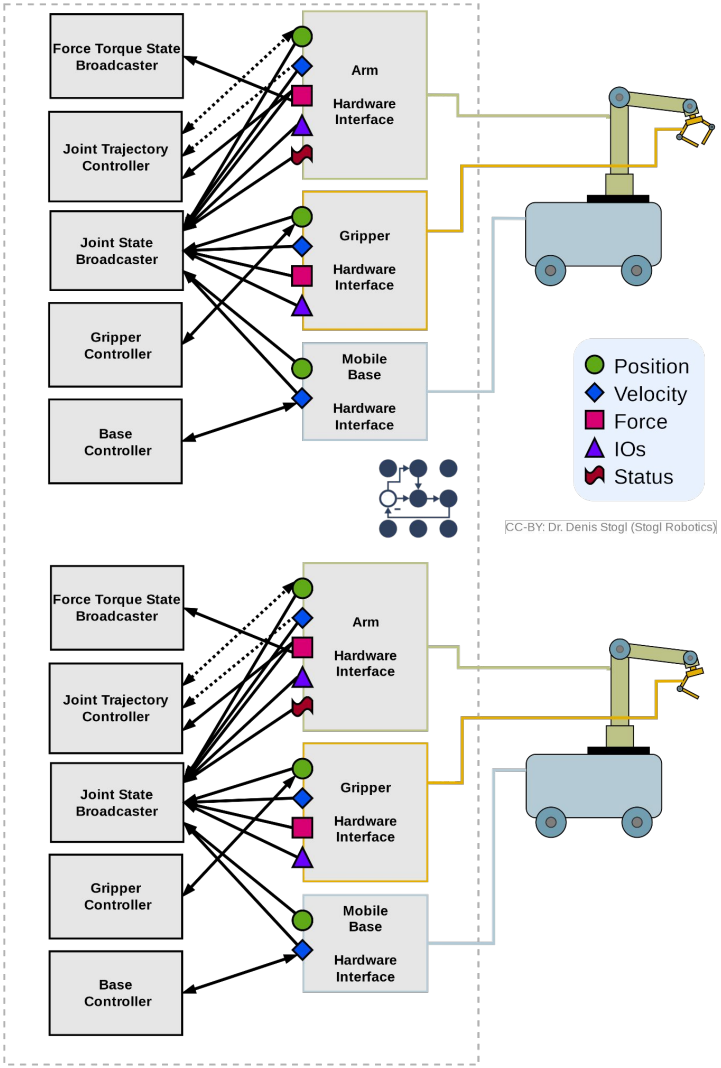
- `ros2 launch hercules_description hercules_sim_control.launch.py`

- `ros2 launch hercules_description test_hercules_controllers.launch.py`

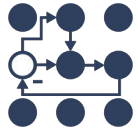


Two Hercules' under one controller_manager

- What do we need?
  - Which Controllers?
  - What Hardware?
  - How to add this into the description?
  - Where to add controllers?



Multi-robot architectures – Solution

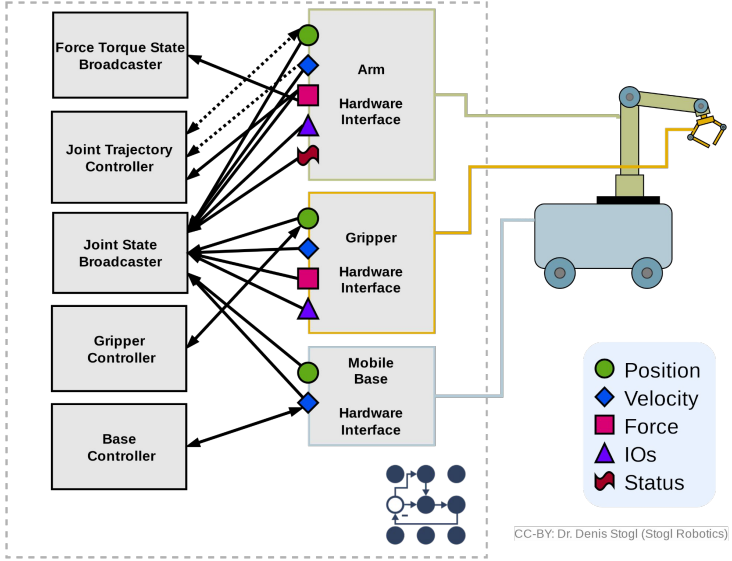


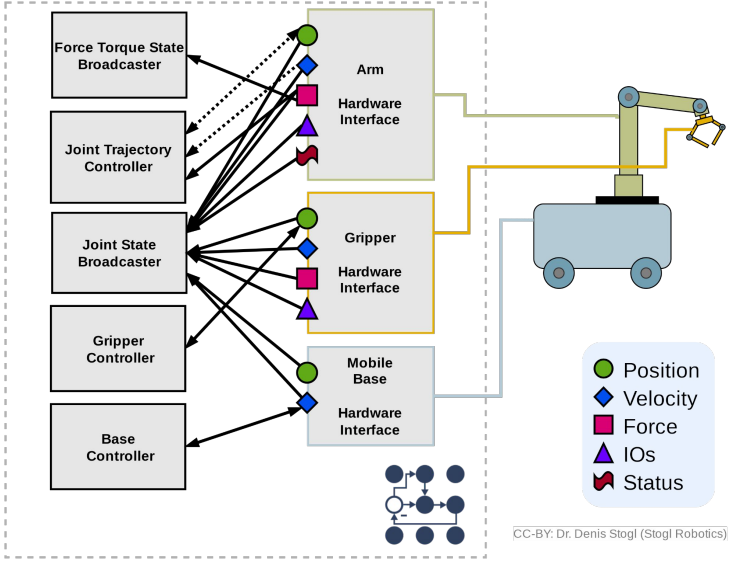
a. `ros2 launch hercules_description multi_hercules_sim_control.launch.py`

b. `ros2 launch hercules_description test_multi_hercules_controllers.lau nch.py`

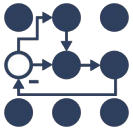
Two Hercules' under two controller_manager s

a. Where to duplicate?
b. What to duplicate?
c. What has to be unique in the system?
d. What can be duplicated?





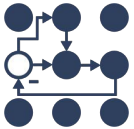
Multi-robot architectures – Solution



a. `ros2 launch hercules_description multi_cm_hercules_sim_control.launch.py`

b. `ros2 launch hercules_description test_multi_cm_hercules_controllers.  launch.py`

Summary: Debugging of complex systems



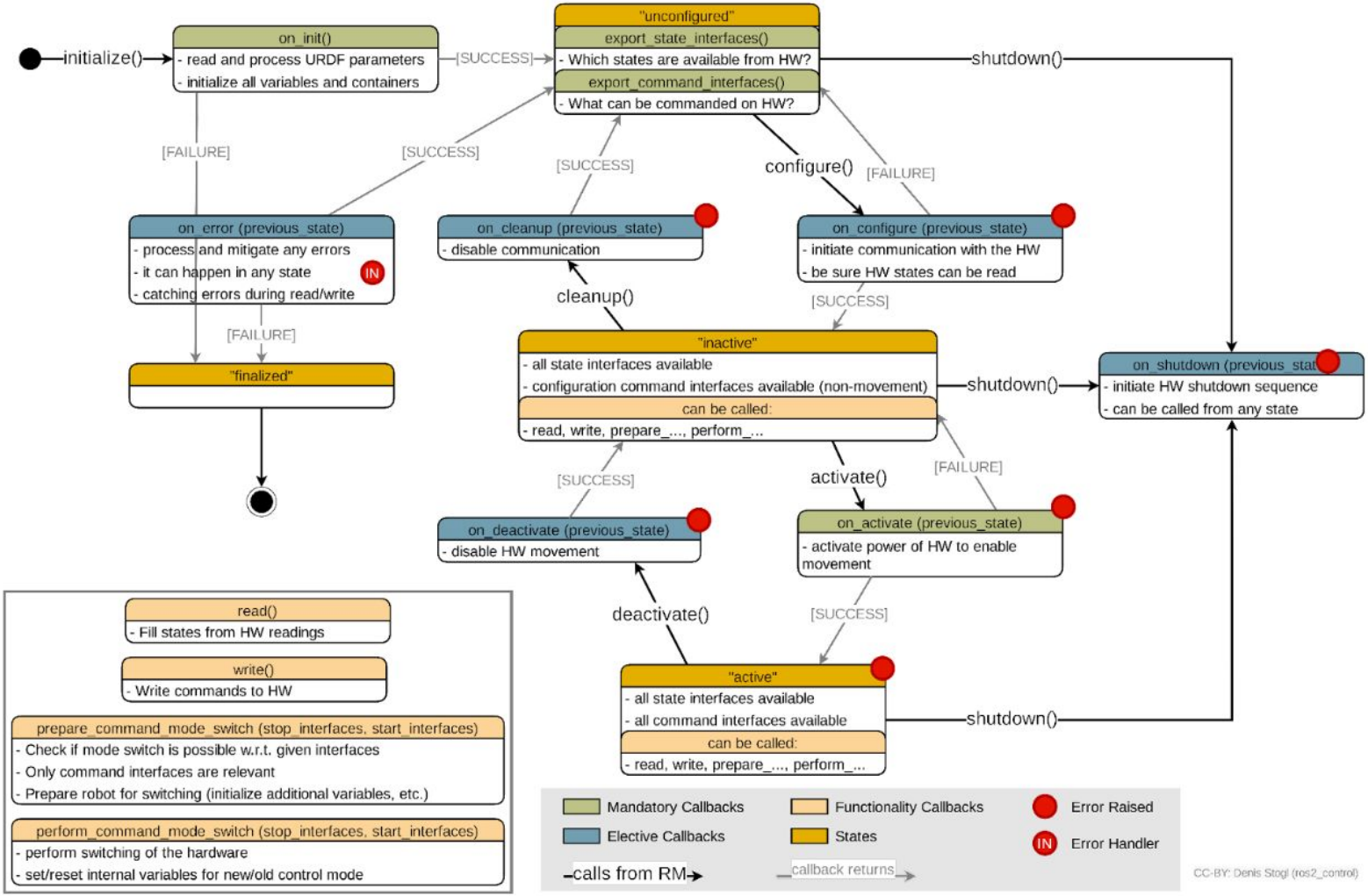
- Good practices for controllers “~/controller_state topic”
- Error handling
- Logging for problem triage: what to add/watch for long-term deployments
- Be aware for your <ros2_control>- tag in URDF → always search for “*.ros2_control.xacro”
- Use “xacro”

Introduce tooling one can use for debugging

- All the different CLI, all CLI really list verbose controllers too
- controller_state topic + plotjuggler/foxglove

Additional possibilities:

- Error handling in read/write – Felix’s PR
- Set logging level through services

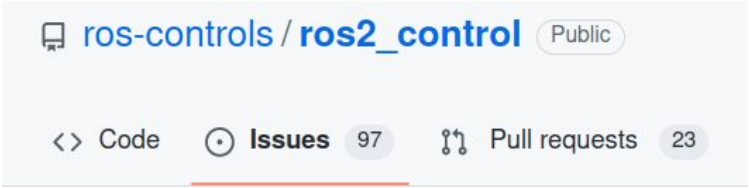

initialize()
on_init()
- read and process URDF parameters
- initialize all variables and containers
[SUCCESS]
"unconfigured"
export_state_interfaces()
- Which states are available from HW?
export_command_interfaces()
- What can be commanded on HW?
shutdown()
[FAILURE]
[SUCCESS]
[SUCCESS]
configure()
[FAILURE]
on_error (previous_state)
- process and mitigate any errors
- it can happen in any state
- catching errors during read/write
IN
on_cleanup (previous_state)
- disable communication
on_configure (previous_state)
- initiate communication with the HW
- be sure HW states can be read
cleanup()
[SUCCESS]
[FAILURE]
"finalized"
"inactive"
- all state interfaces available
- configuration command interfaces available (non-movement)
can be called:
- read, write, prepare_..., perform_...
shutdown()
on_shutdown (previous_state)
- initiate HW shutdown sequence
- can be called from any state
[SUCCESS]
activate()
[FAILURE]
on_deactivate (previous_state)
- disable HW movement
on_activate (previous_state)
- activate power of HW to enable movement
deactivate()
[SUCCESS]
"active"
- all state interfaces available
- all command interfaces available
can be called:
- read, write, prepare_..., perform_...
shutdown()
read()
- Fill states from HW readings
write()
- Write commands to 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
Mandatory Callbacks
Elective Callbacks
Functionality Callbacks
States
Error Raised
IN
Error Handler
calls from RM
callback returns
CC-BY: Denis Stogl (ros2_control)

Getting involved

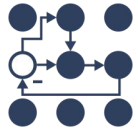
https://github.com/ros-controls

ros2_control Working Group

- Every second Wednesday 6PM UK time, 1PM Eastern Time
- Discourse announcements
- You are invited!



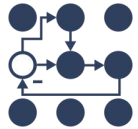
References



- https://control.ros.org

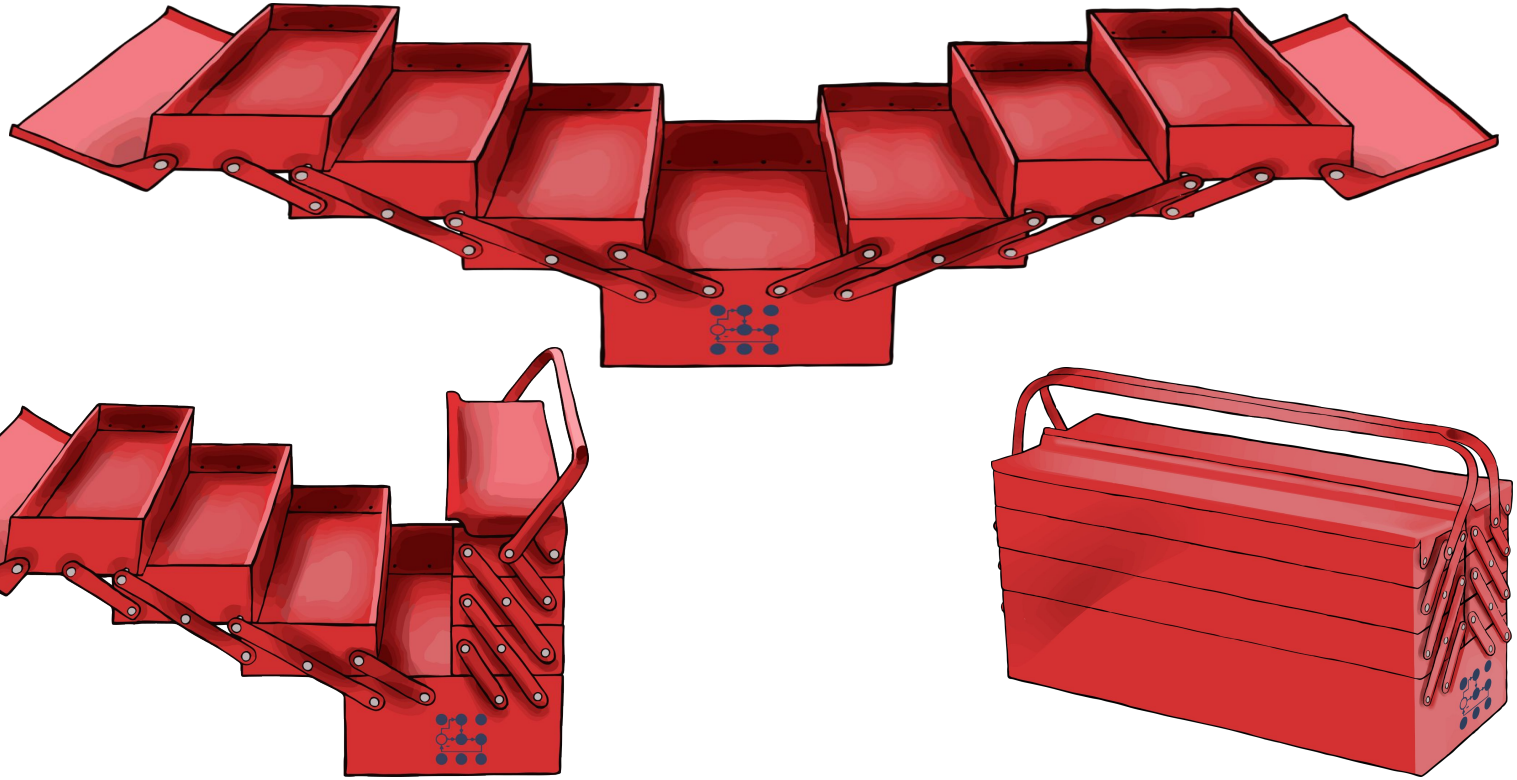
- ros_control paper in the Journal of Open Source Software
- ros2_control presentations
  - https://control.ros.org/master/doc/resources/resources.html
- ros2_control resources
  - https://ros-controls.github.io/control.ros.org/
  - https://github.com/ros-controls/ros2_control
  - https://github.com/ros-controls/ros2_controllers
  - https://github.com/ros-controls/ros2_control_demos
  - https://github.com/ros-controls/roadmap/blob/master/documentation_resources.md

Thank you!



Christoph Fröhlich, Denis Štogl, Bence Magyar, Sai Kishor Kothakota, Felix Exner, Manuel Muth, Alex Moriarty, Marq Rasmussen, Tyler Weaver, Olivier Stasse, Alejandro Hernández Cordero, Reza Kermani, Lovro Ivanov, Paul Gesel, Tony Najjar, Karsten Knese, Victor Lopez, Jordan Palacios, Márk Szitanics, Andy Zelenak, Noel Jiménez García, Jaron Lundwall, Tim Clephas, Erick G. Islas-Osuna, Abrar Rahman Protyasha and many more!

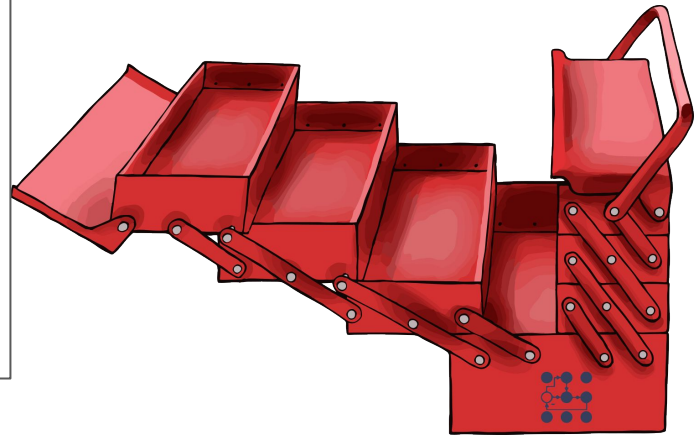
BACKUP SLIDES START HERE



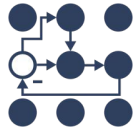
ros2_control CLI - Integrated with ROS2 CLI

$ ros2 control list_hardware_interfaces

```
command interfaces
        flange_gpios/digital_out_1 [available] [unclaimed]
        flange_gpios/digital_out_2 [available] [unclaimed]
        joint1/position [available] [claimed]
        joint1/velocity [available] [unclaimed]
        joint2/position [available] [claimed]
        joint2/velocity [available] [unclaimed]
state interfaces
        flange_gpios/digital_in_1
        flange_gpios/digital_in_2
        flange_gpios/digital_out_1
        flange_gpios/digital_out_2
        joint1/effort
        joint1/position
        joint1/velocity
        joint2/effort
        joint2/position
        joint2/velocity
```

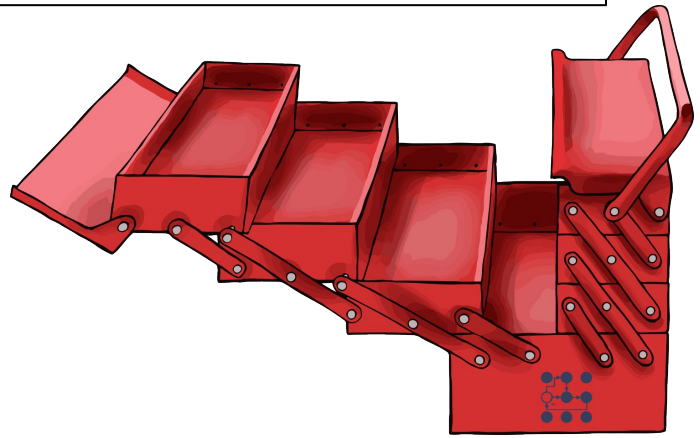


ros2_control CLI - Integrated with ROS2 CLI

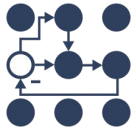


$ ros2 control list_controllers

```
joint_state_broadcaster[joint_state_broadcaster/JointStateBroadcaster] active
forward_position_controller[forward_command_controller/ForwardCommandController] active
joint_trajectory_controller[joint_trajectory_controller/JointTrajectoryController] inactive
```

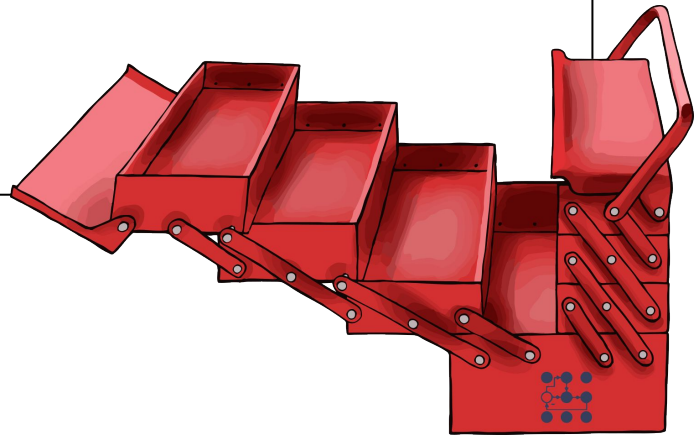


ros2_control CLI - Integrated with ROS2 CLI

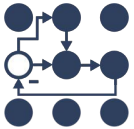


$ ros2 control list_controllers -v

```
...
forward_position_controller[forward_command_controller/ForwardCommandController] active
        claimed interfaces:
                joint1/position
                joint2/position
        required command interfaces:
                joint1/position
                joint2/position
        required state interfaces:
        chained to interfaces:
        exported reference interfaces:
...
```



What config files and where?



```yaml
controller_manager:
    ros__parameters:
        update_rate: 10  # Hz

        joint_state_broadcaster:
        type: joint_state_broadcaster/JointStateBroadcaster

        position_trajectory_controller:
        type: joint_trajectory_controller/JointTrajectoryController

position_trajectory_controller:
    ros__parameters:
        joints:
            - joint1
            - joint2

        command_interfaces:
            - position

        state_interfaces:
            - position

        state_publish_rate: 200.0 # Defaults to 50
        action_monitor_rate: 20.0 # Defaults to 20

        allow_partial_joints_goal: false # Defaults to false
        open_loop_control: true
        allow_integration_in_goal_trajectories: true
        constraints:
            stopped_velocity_tolerance: 0.01 # Defaults to 0.01
            goal_time: 0.0 # Defaults to 0.0 (start immediately)
```

```xml
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">

  <xacro:macro name="rrbot_ros2_control" params="name prefix">

    <ros2_control name="${name}" type="system">
      <hardware>
        <plugin>ros2_control_demo_hardware/RRBotSystemPositionOnlyHardware</plugin>
        <param name="example_param_hw_start_duration_sec">0</param>
        <param name="example_param_hw_stop_duration_sec">3.0</param>
        <param name="example_param_hw_slowdown">100</param>
      </hardware>

      <joint name="${prefix}joint1">
        <command_interface name="position">
          <param name="min">-1</param>
          <param name="max">1</param>
        </command_interface>
        <state_interface name="position"/>
      </joint>
      <joint name="${prefix}joint2">
        <command_interface name="position">
          <param name="min">-1</param>
          <param name="max">1</param>
        </command_interface>
        <state_interface name="position"/>
      </joint>
    </ros2_control>

  </xacro:macro>

</robot>
```

```python
control_node = Node(
    package="controller_manager",
    executable="ros2_control_node",
    parameters=[robot_description, robot_controllers],
    remappings=[
        (
            "/forward_position_controller/commands",
            "/position_commands",
        ),
    ],
    output="both",
)

robot_state_pub_node = Node(
    package="robot_state_publisher",
    executable="robot_state_publisher",
    output="both",
    parameters=[robot_description],
)

joint_state_broadcaster_spawner = Node(
    package="controller_manager",
    executable="spawner",
    arguments=["joint_state_broadcaster", "--controller-manager", "/controller_manager"],
)

robot_controller_spawner = Node(
    package="controller_manager",
    executable="spawner",
    arguments=["forward_position_controller", "-c", "/controller_manager"],
)

nodes = [
    control_node,
    robot_state_pub_node,
    joint_state_broadcaster_spawner,
    robot_controller_spawner,
]

return LaunchDescription(nodes)
```