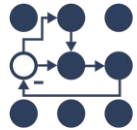
STOGL ROBOTICS

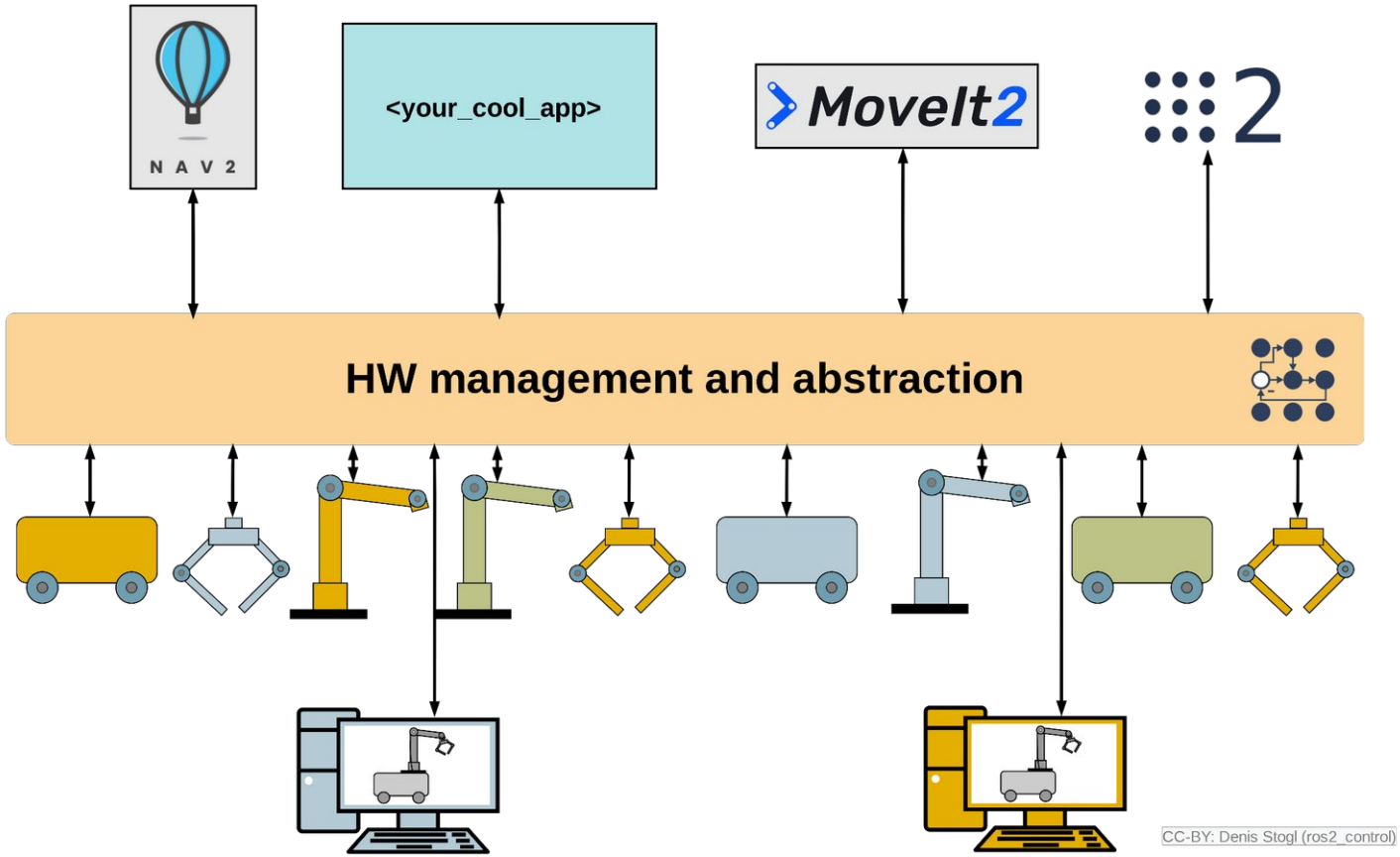
ros2_control - the Reusable Kernel for Robots



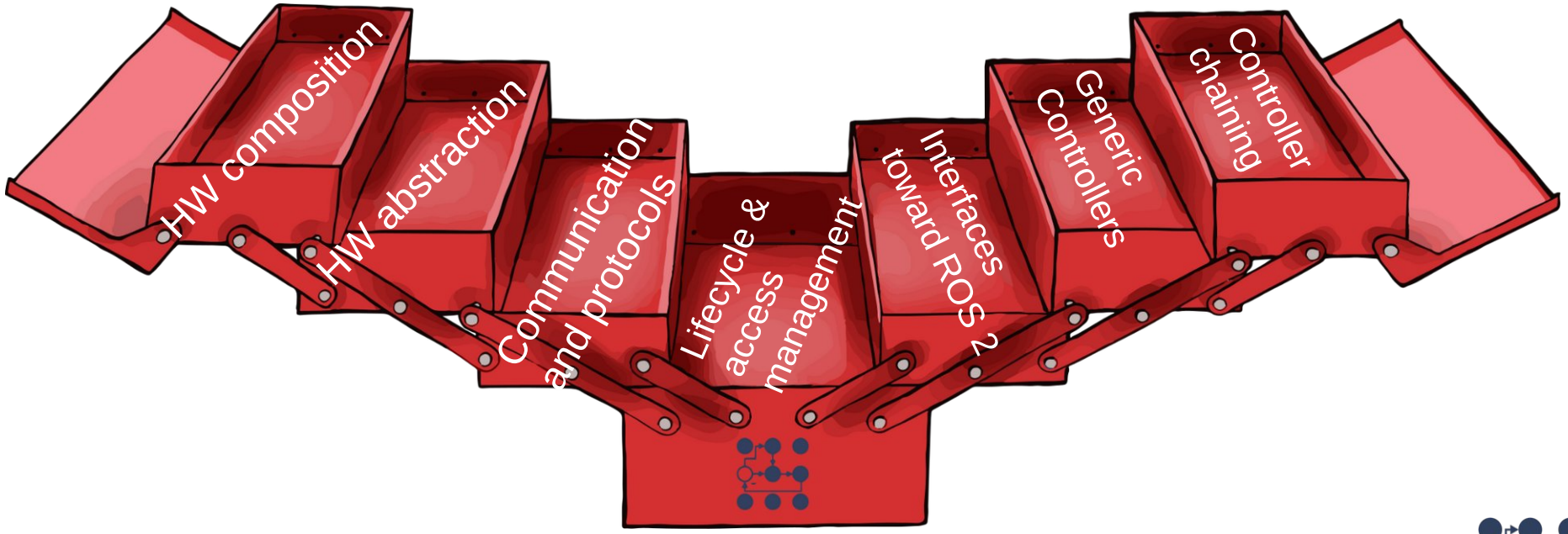
ros2_control@stoglrobotics.de

- Professional support for ros2_control
  - and ROS / ROS2
- CxO: Dr.-Ing. Denis Štogl
  - ros2_control maintainer
- 2 full-time engineers
- 2 long-term Freelancers

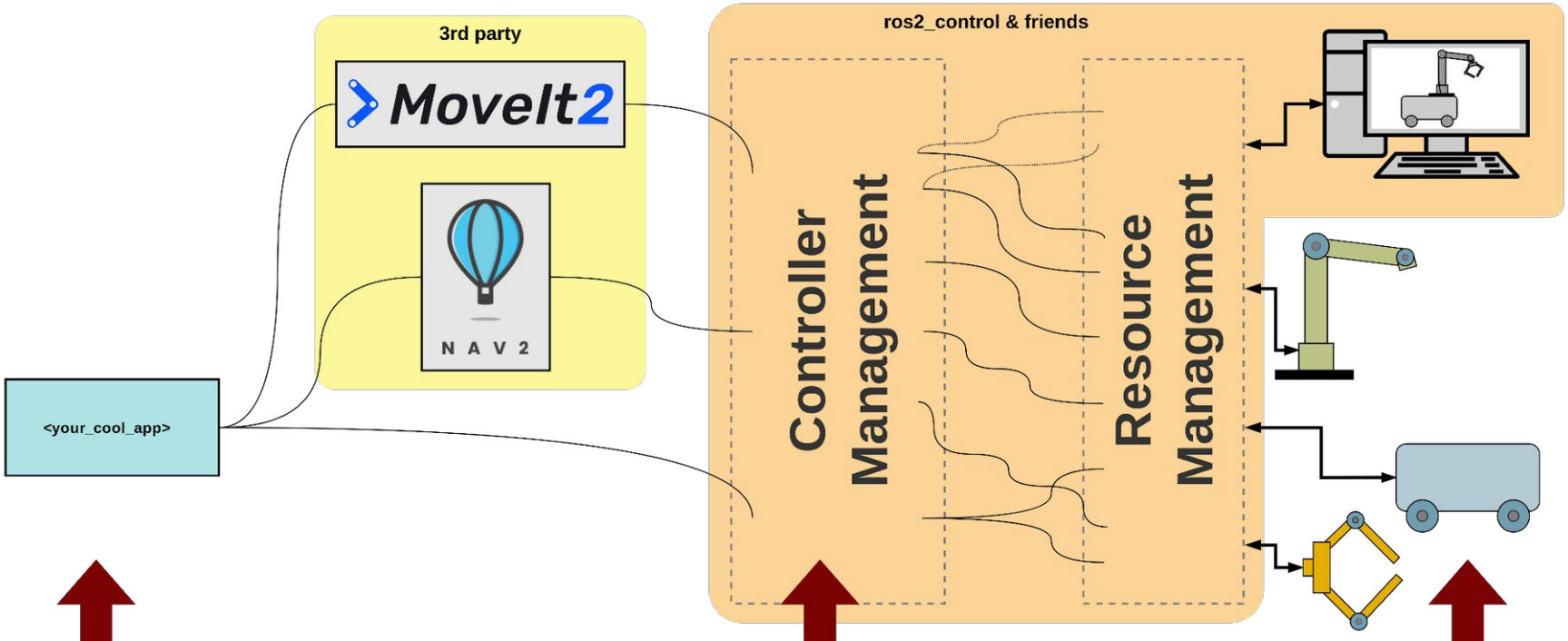
ros2_control – Kernel for ROS 2



ros2_control – Kernel for ROS 2



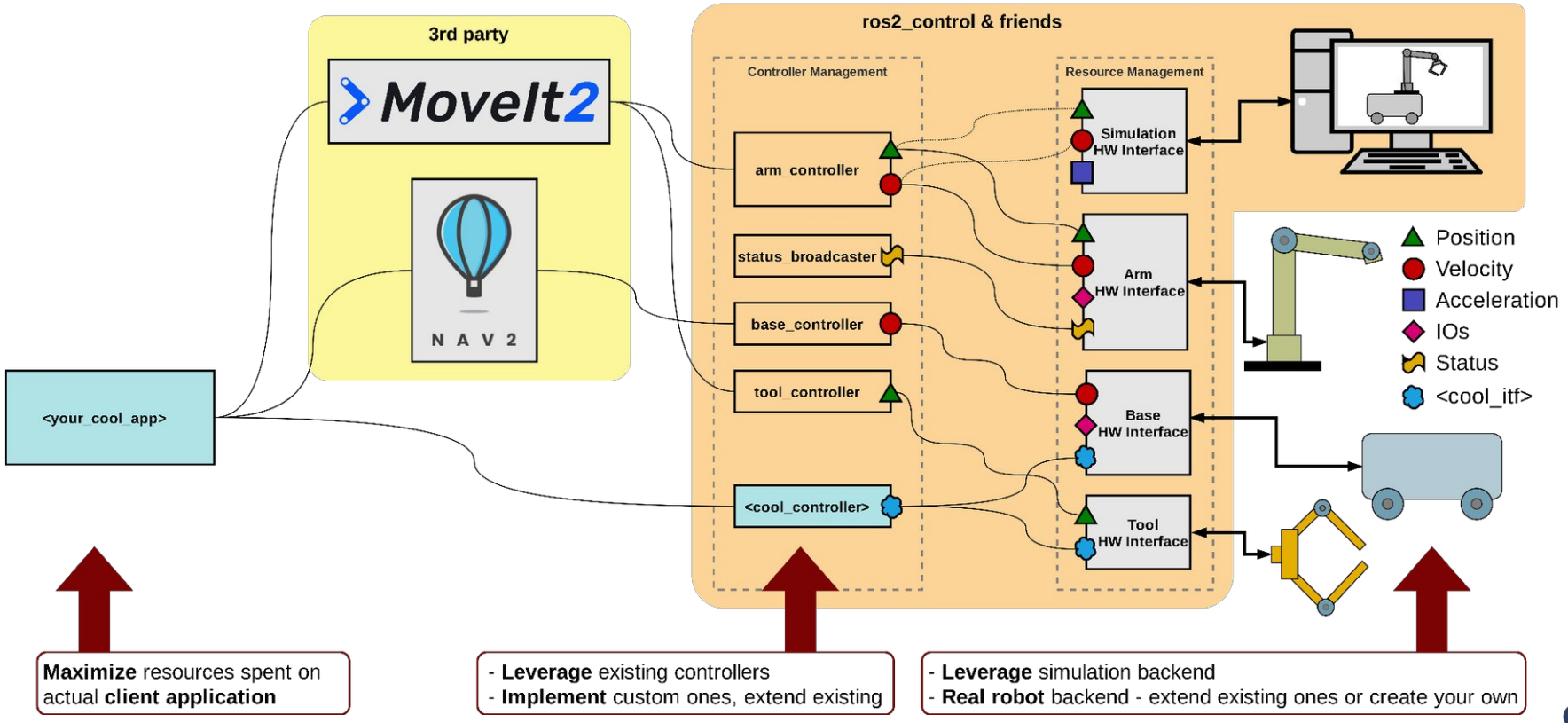
ros2_control – Kernel for ROS 2



Maximize resources spent on actual **client application**

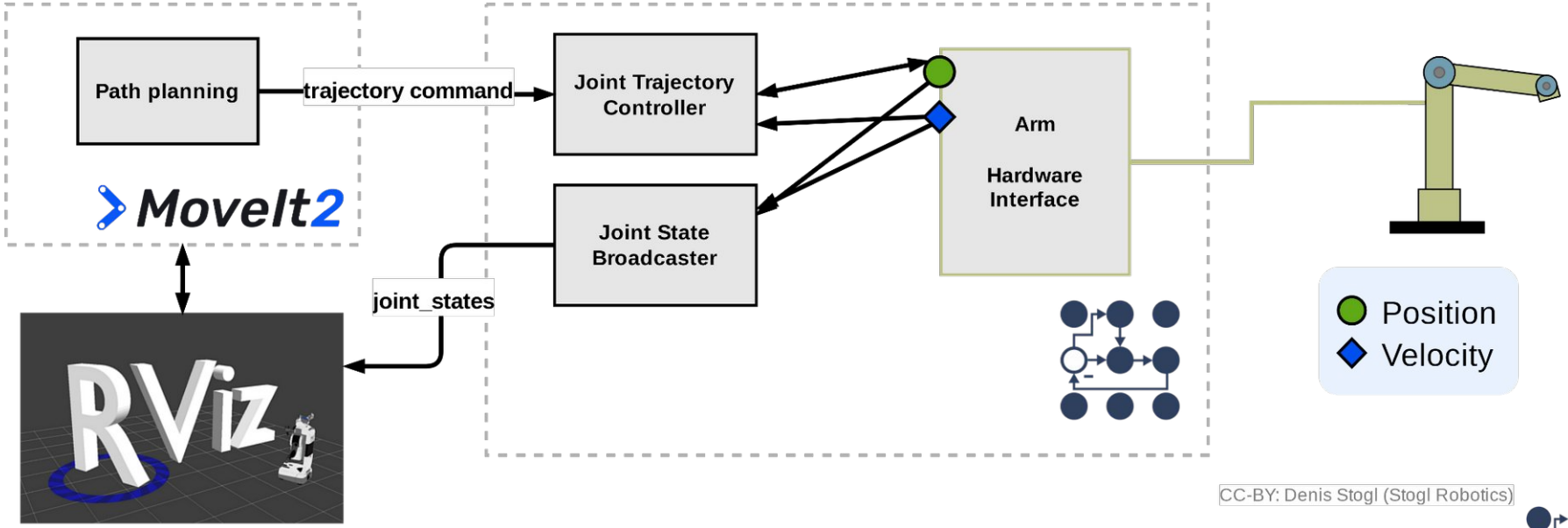
- **Leverage** existing controllers
- **Implement** custom ones, extend existing

- **Leverage** simulation backend
- **Real robot** backend - extend existing ones or create your own


3rd party
MoveIt2
N A V 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

Use-Case: Force Compensation along Trajectories

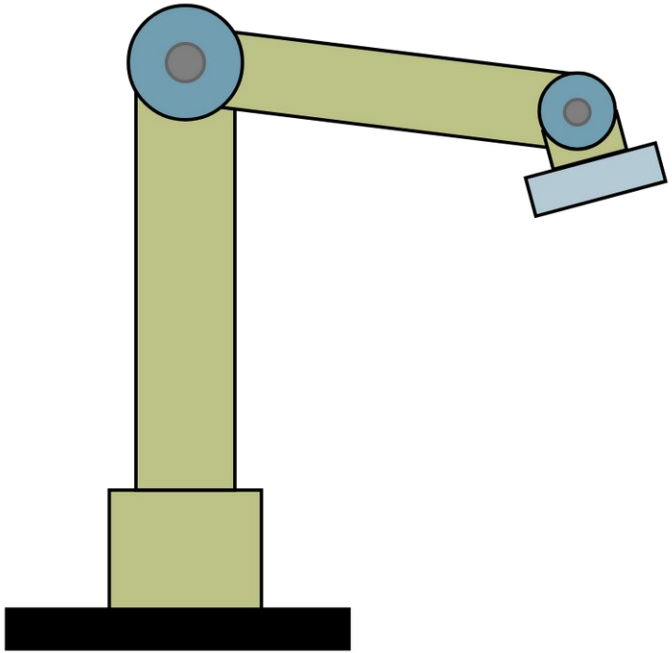
Starting point: a manipulator working with JTC 🤔



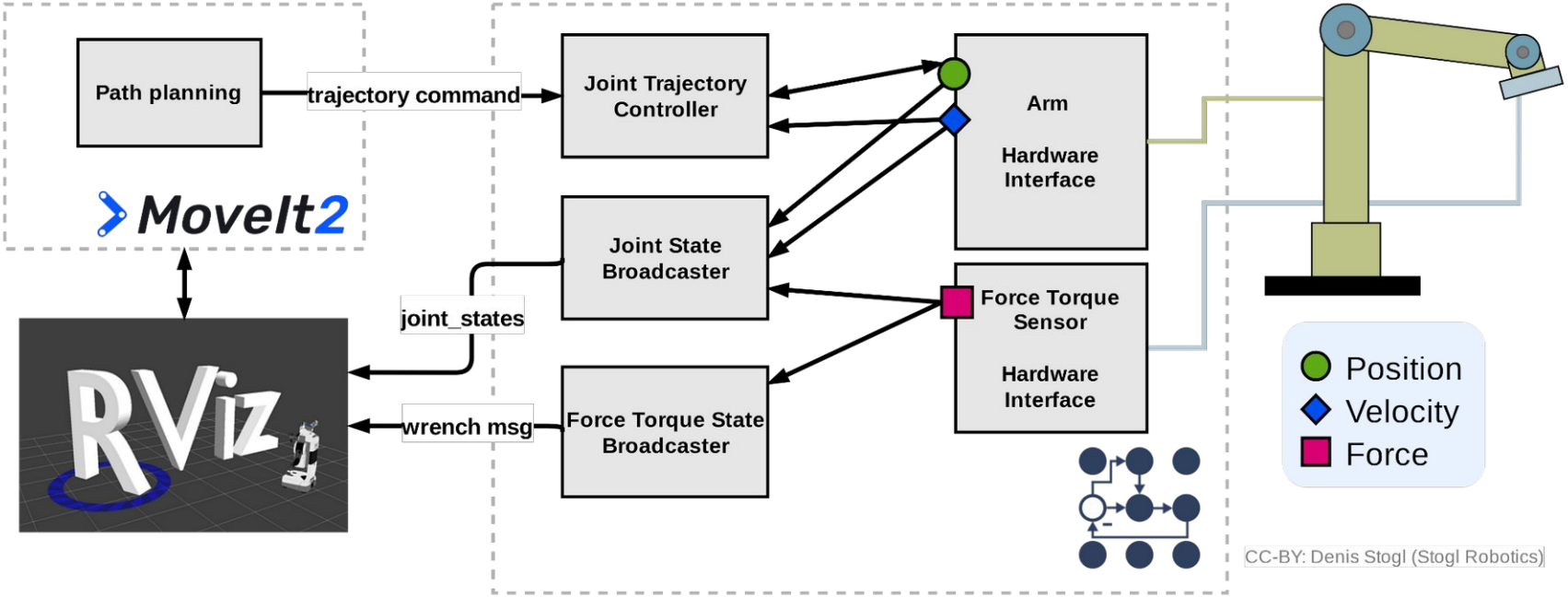
1. “I need also a sensor for Force Control”

Ask yourself a Question:

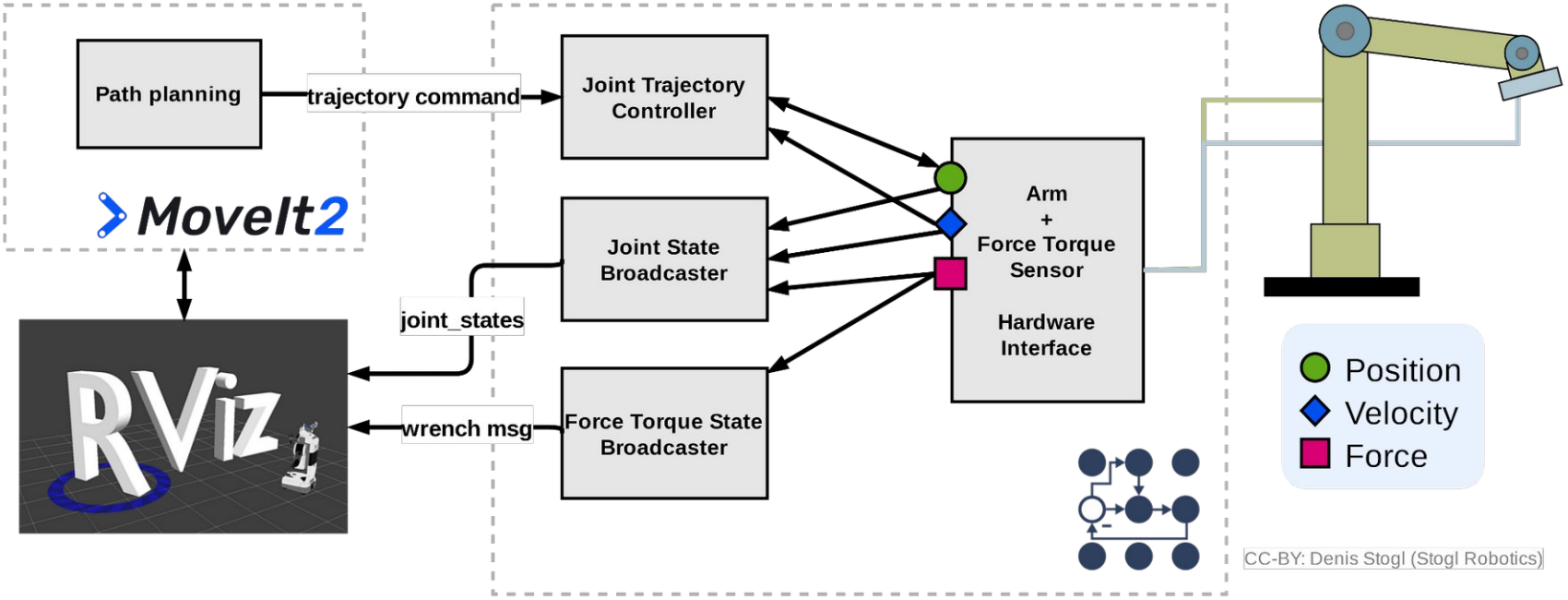
“How many communications paths are there?”



Modelling complex hardware – individual components

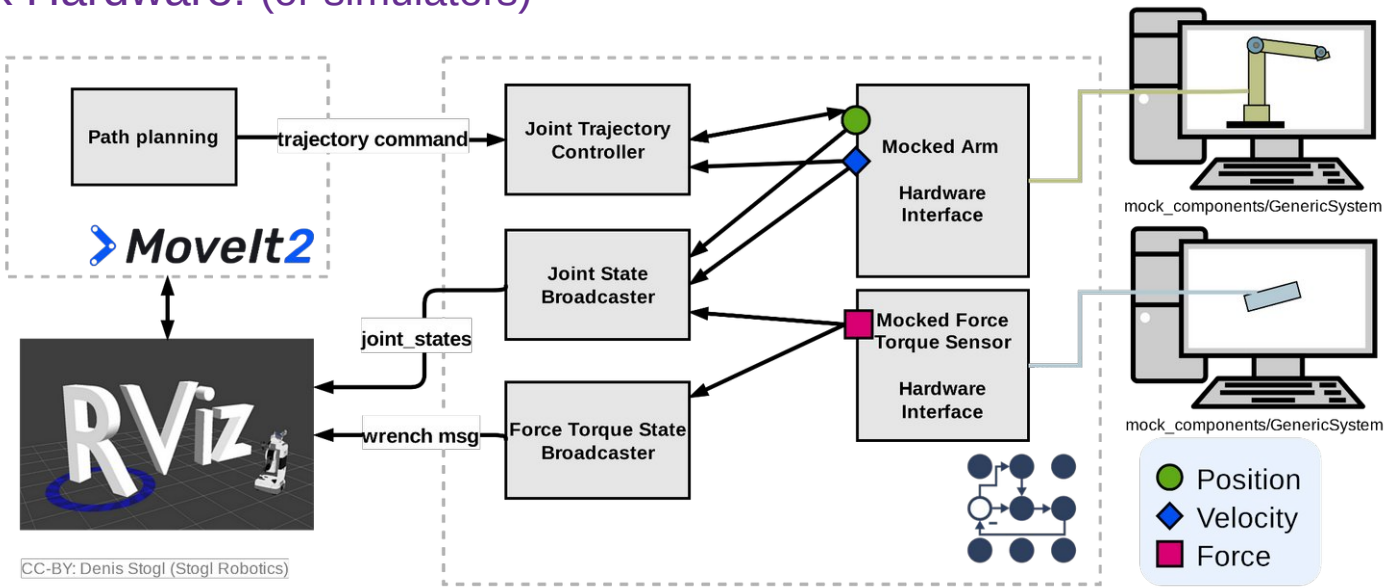


Modelling complex hardware – “bus through arm”



2. “I don’t have the hardware when I work remotely“ 😢

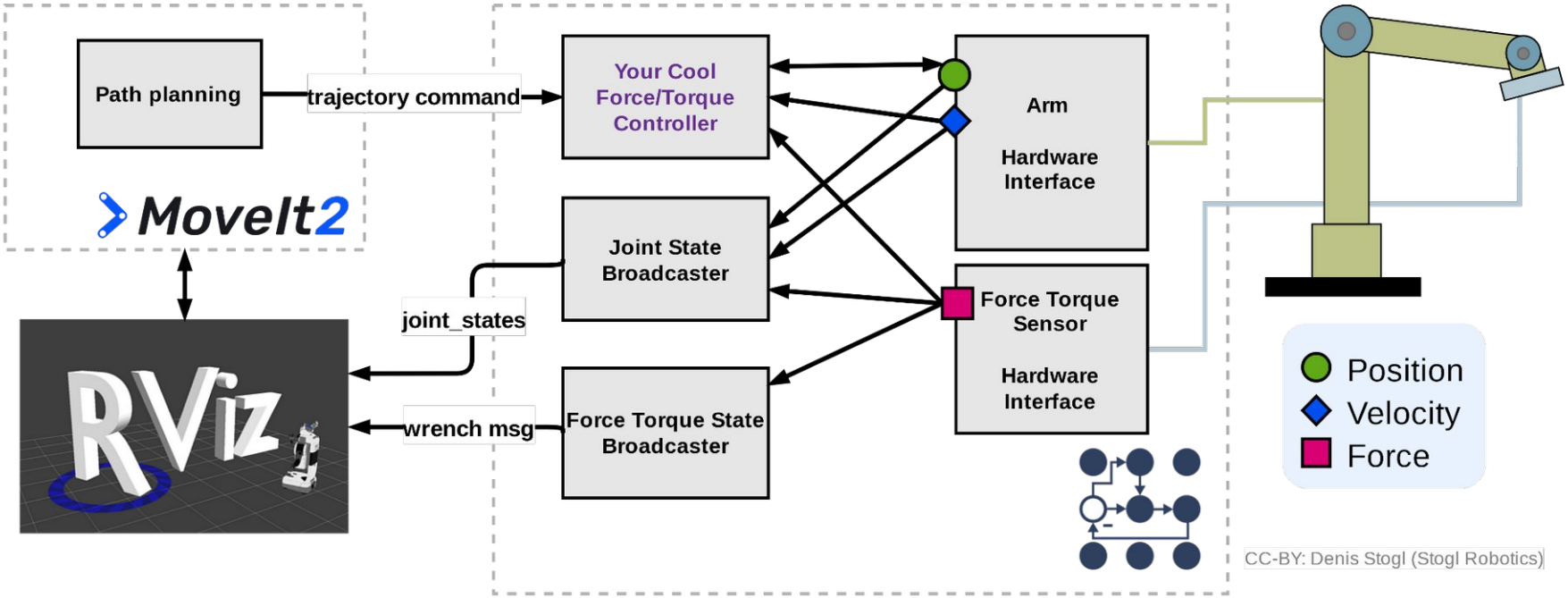
Use Mock Hardware! (or simulators)




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

  <hardware>
    <plugin>ros2_control_demo_hardware/RRBotSystemPositionOnlyHardware</plugin>
    <param name="hw_start_duration_sec">0.0</param>
    <param name="hw_stop_duration_sec">3.0</param>
    <param name="hw_slowdown_factor">2.0</param>
  <hardware>
```

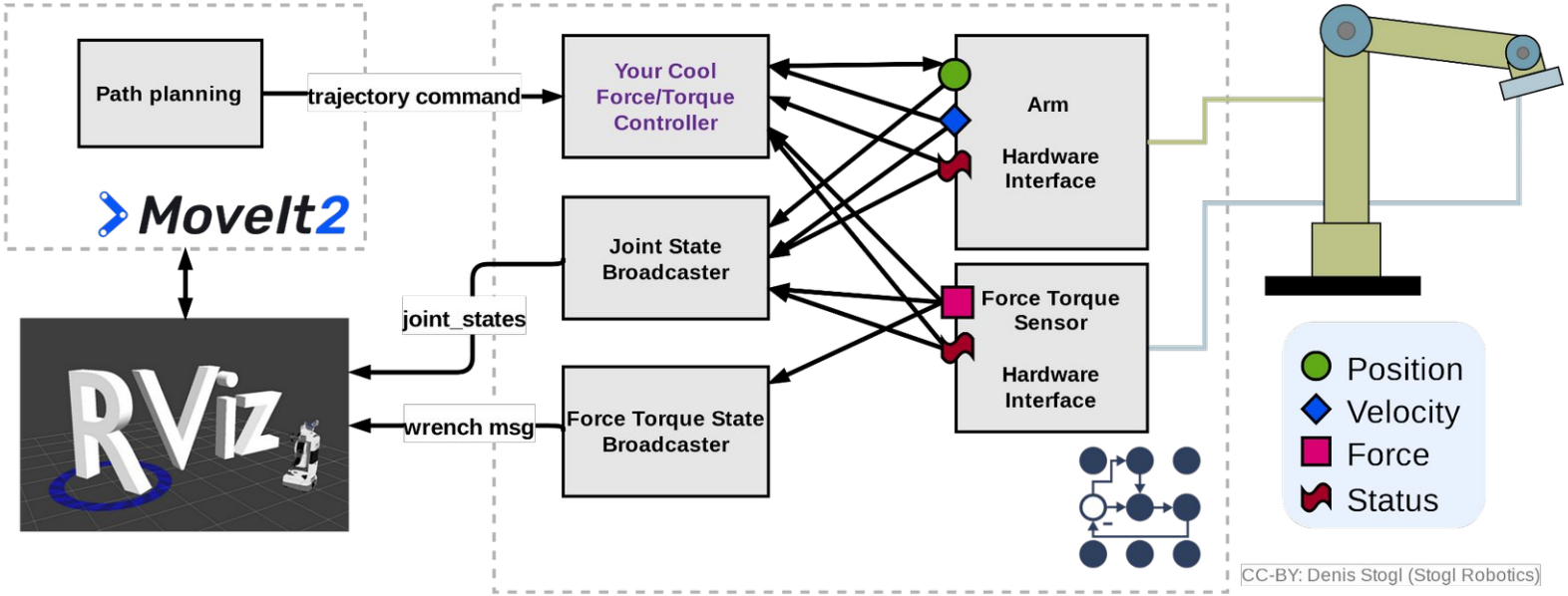
```
<ros2_control name="rrbot_mock" type="system">

  <hardware>
      <plugin>mock_components/GenericSystem</plugin>
      <param name="fake_sensor_commands">True</param>
      <param name="state_following_offset">0.0</param>
  </hardware>
```

3. 🎉 “I have my controller!”



4. “Oh, but my robot ends in torque limits, can I detect this?”

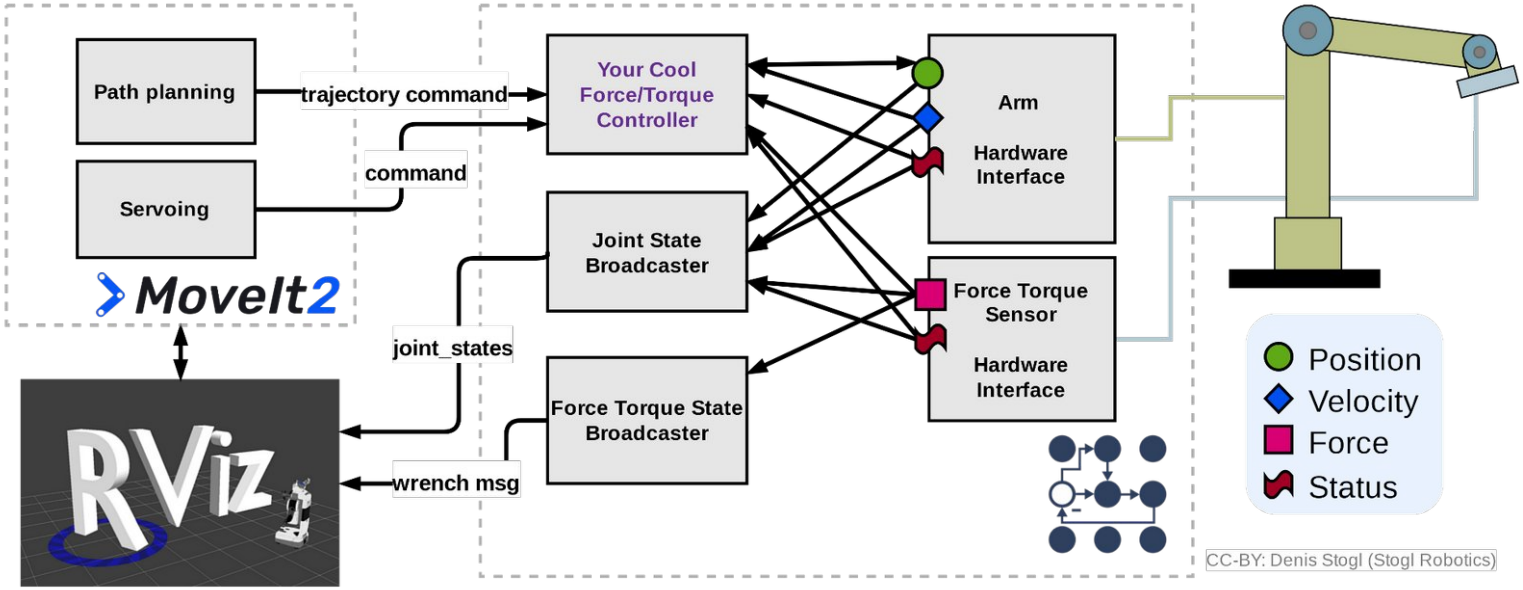
Just add interfaces for it!



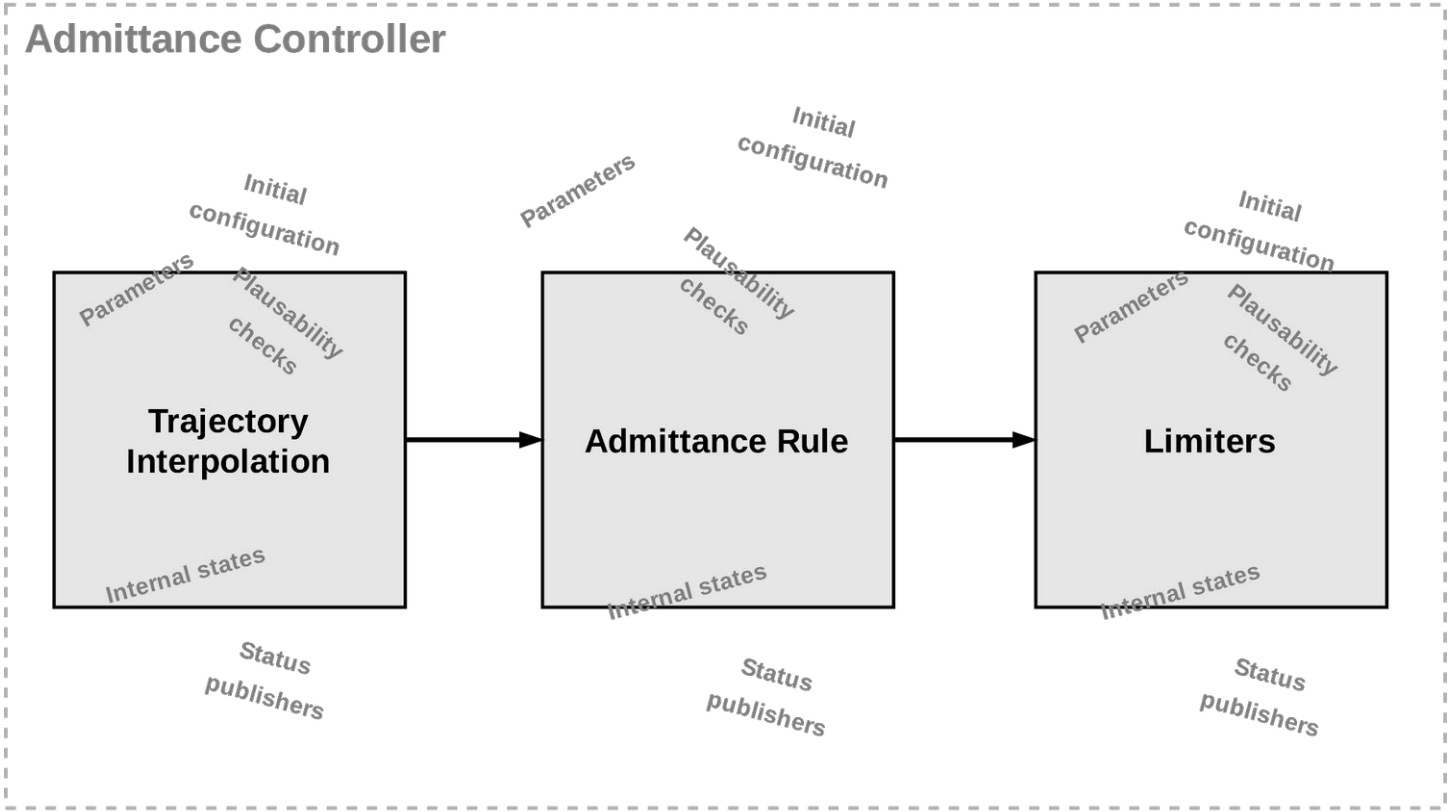
5. “My colleagues want to use my algorithm for teleoperation…” 😐

STOGL ROBOTICS

No problem → new subscriber in the controller!

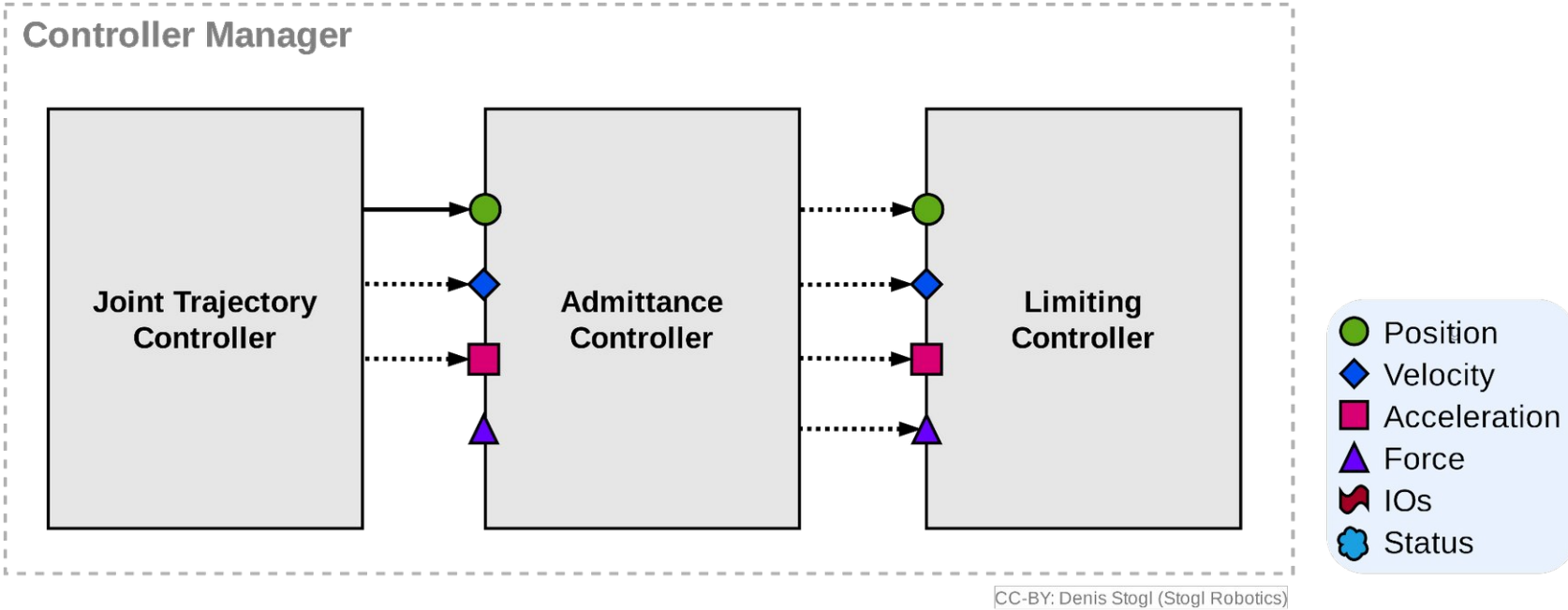


6. “My controller is getting too complex” 🤯



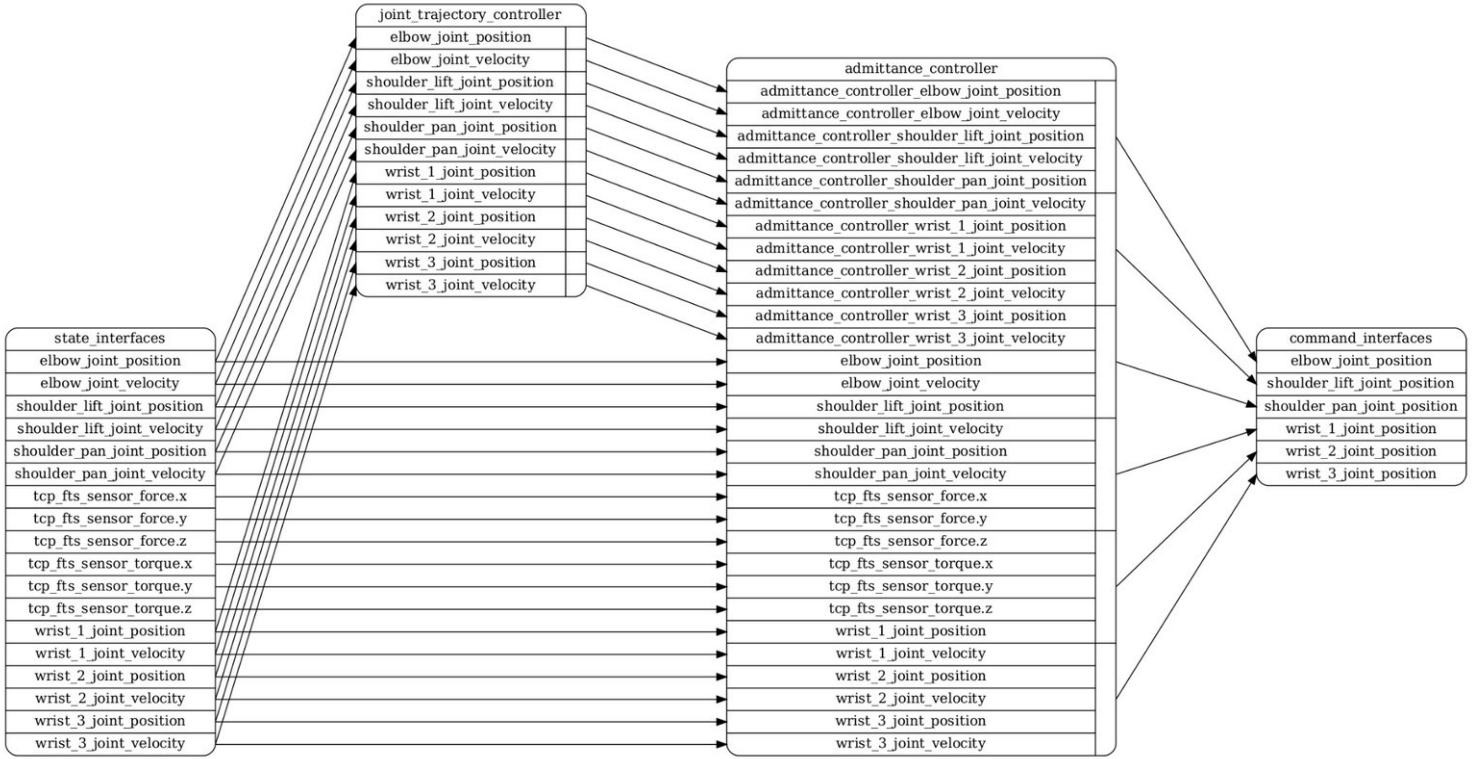
6. “My controller is getting too complex” 🤯

No problem → use controller’s chaining!

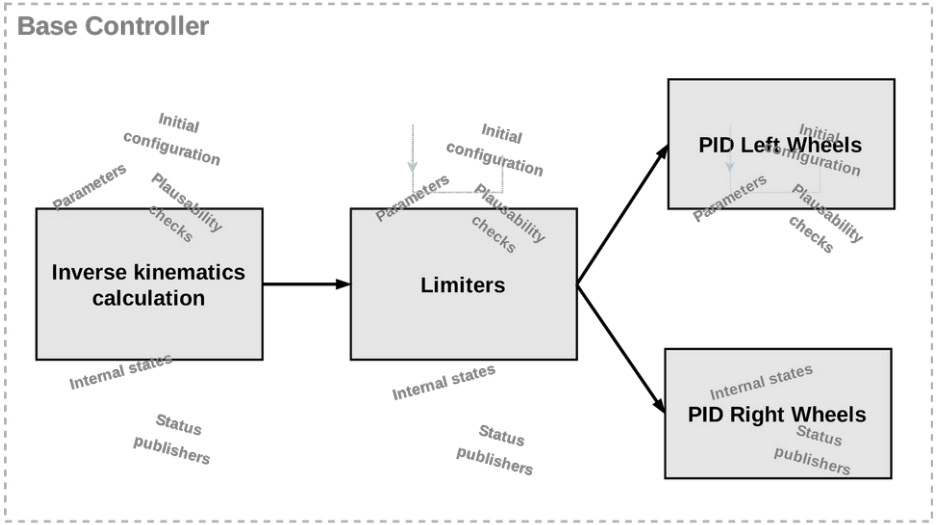


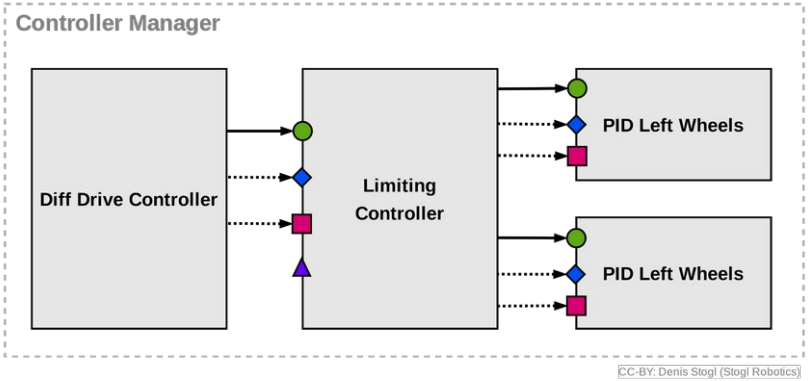
Chaining Controllers – great introspection

$ ros2 control view_controller_chains



Chaining Controllers – arbitrary architecture

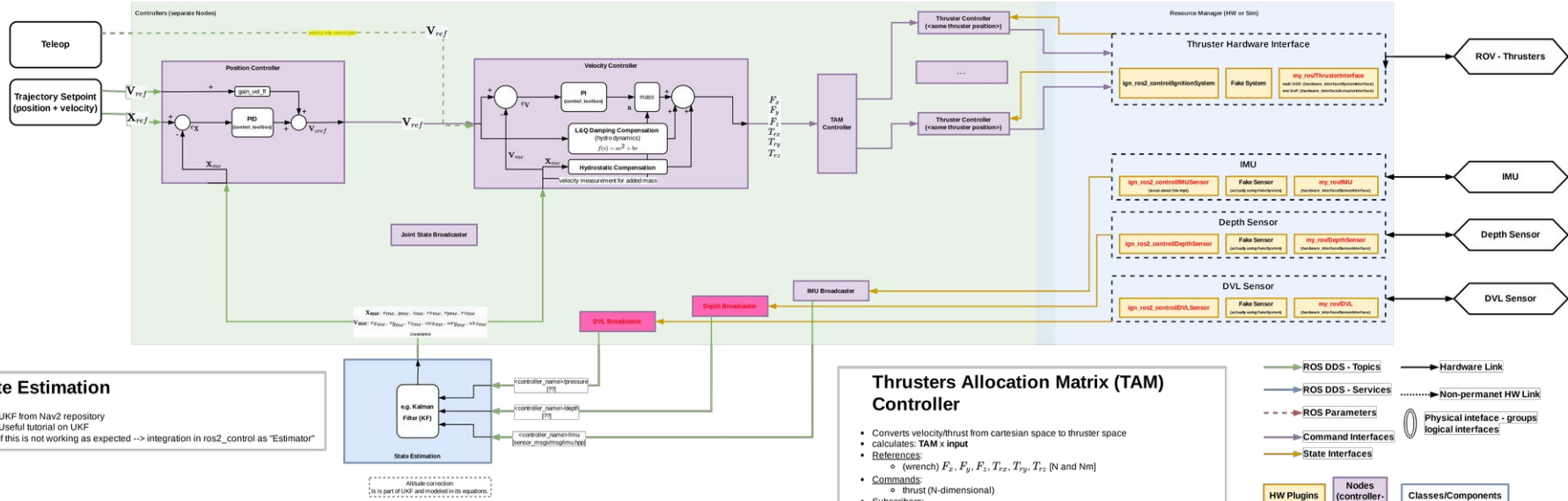




Chaining Controllers – arbitrary architecture

$$Axes = \begin{bmatrix} x \\ y \\ z \\ rx \\ ry \\ rz \end{bmatrix} = \begin{bmatrix} x \\ y \\ depth \\ roll \\ pitch \\ heading \end{bmatrix}$$

For better overview, not all topics and parameters are shown for all controllers.
Please check controllers' description below for more details.



State Estimation

- UKF from Nav2 repository
- Useful tutorial on UKF
- if this is not working as expected --> integration in ros2_control as "Estimator"

Thrusters Allocation Matrix (TAM) Controller

- Converts velocity/thrust from cartesian space to thruster space
- calculates: **TAM** x **input**
- References:
  - (wrench) $F_x, F_y, F_z, T_{rx}, T_{ry}, T_{rz}$ [N and Nm]
- Commands:
  - thrust (N-dimensional)
- Subscribers:
  - <controller_name>/wrench [geometry_msgs/msg/WrenchStamped]
- Parameters:
  - TAM columns:
    - TAM_x [N-sized double array]
    - TAM_y [N-sized double array]
    - TAM_z [N-sized double array]
    - TAM_{rx} [N-sized double array]
    - TAM_{ry} [N-sized double array]
    - TAM_{rz} [N-sized double array]
- ALTERNTIVE:
  - get thruster positions from URDF or /tf

ROS DDS - Topics
ROS DDS - Services
ROS Parameters
Command Interfaces
State Interfaces
Hardware Link
Non-permanet HW Link
Physical inteface - groups logical interfaces
HW Plugins
Nodes (controller-plugins=
Nodes
Hardware
Classes/Components
has to be implemented
needs discussion / optional

Position Controller

- PD Controller
- Inputs:
  - Target position $\mathbf{X_{ref}}$: $x_{ref}, y_{ref}, z_{ref}, rx_{ref}, ry_{ref}, rz_{ref}$ [m and rad]
  - Target velocity $\mathbf{V_{ref}}$: $v_{x_{ref}}, v_{y_{ref}}, v_{z_{ref}}, \omega_{rx_{ref}}, \omega_{ry_{ref}}, \omega_{rz_{ref}}$ [$\frac{m}{s}$ and $\frac{rad}{s}$]
  - Measured position $\mathbf{X_{me}}$: $x_{me}, y_{me}, z_{me}, rx_{me}, ry_{me}, rz_{me}$ [m and rad]
- Outputs:
  - Calculated Target velocity $\mathbf{V_{cref}}$: $v_{x_{cref}}, v_{y_{cref}}, v_{z_{cref}}, \omega_{rx_{cref}}, \omega_{ry_{cref}}, \omega_{rz_{cref}}$ [$\frac{m}{s}$ and $\frac{rad}{s}$]
- Parameters:
  - pids:
    - x: {p, d}
    - y: {p, d}
    - z: {p, d}
    - rx: {p, d}
    - ry: {p, d}
    - rz: {p, d}
  - gain_vel_ff: {x, y, z, rx, ry, rz} [double]

Velocity Controller

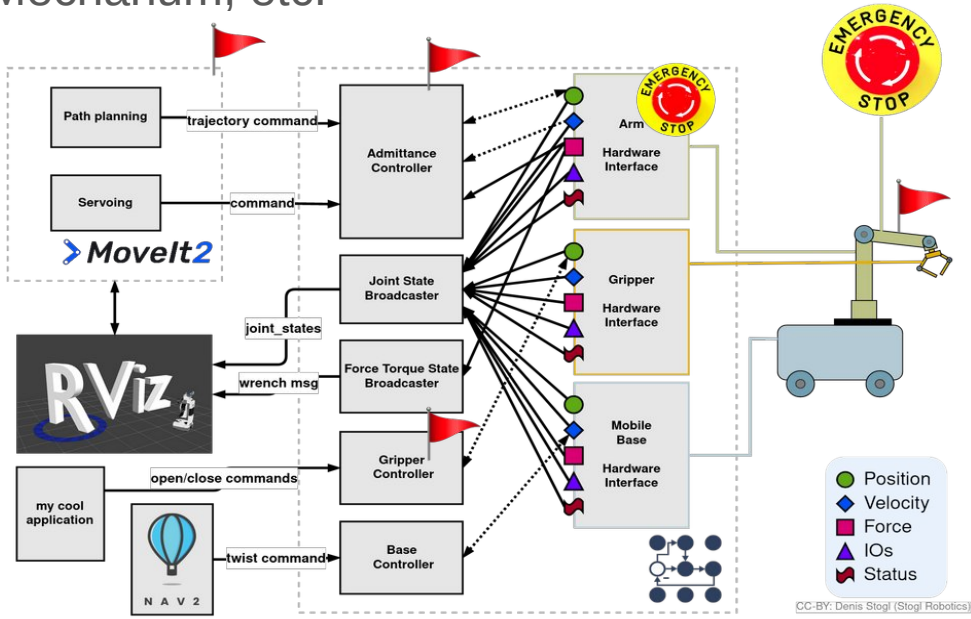
- PI Controller
- calculates acceleration and before output to TAM the values are scaled with the "mass" to get Forces and Torques
- L&Q Damping compensation uses velocity reference for calculation
- Hydrostatic compensation uses measured position of pitch and roll for calculation
- Adjustment of mass based on measured velocity
- Inputs:
  - Target velocity $\mathbf{V_{ref}}$: $v_{x_{ref}}, v_{y_{ref}}, v_{z_{ref}}, \omega_{rx_{ref}}, \omega_{ry_{ref}}, \omega_{rz_{ref}}$ [$\frac{m}{s}$ and $\frac{rad}{s}$]
  - Measured velocity $\mathbf{V_{me}}$: $v_{x_{me}}, v_{y_{me}}, v_{z_{me}}, \omega_{rx_{me}}, \omega_{ry_{me}}, \omega_{rz_{me}}$ [$\frac{m}{s}$ and $\frac{rad}{s}$]
- Outputs:
  - $F_x, F_y, F_z, T_{rx}, T_{ry}, T_{rz}$ [N and Nm]
- Parameters:
  - pids:
    - x: {p, i}
    - y: {p, i}
    - z: {p, i}
    - rx: {p, i}

Thruster Controller

- 1 DoF controller
- Name: <thruster_id>_thrust_controller
- References:
  - <thruster_id>/thrust [double]
- States:

What's next?

- Chaining for state interfaces → Estimators
- Complex interface → vectors, (strings)
- Controllers: GPIO, Steering, Mechanum, etc.
- Interfaces to PLCs
- "SR – ros2_control box"



People behind ros2_control

Karsten Knese, Victor Lopez, Jordan Palacios, Tyler Weaver, Márk Szitanics, Anas Abou Allaban, Paul Gesel, Tony Najjar, Andy Zelenak, Olivier Stasse, Felix Exner, Sachin Kumar, Noel Jiménez García, Jaron Lundwall, Alejandro Hernández Cordero, Maciej Bednarczyk, Patrick Roncagliolo, Matt Reynolds, Colin MacKenzie, El Jawad Alaa, Auguste Bourgois, Vatan Aksoy, Tony Najjar, Erick G. Islas-Osuna, Christoph Fröhlich, Tezer, Tim Clephas, Lovro Ivanov, Jafar Abdi, Michael Wiznitzer, Patrick Roncagliolo, Bence Magyar, Denis Štogl and many more!

Control WG meeting
every second Wednesday
at 7pm CET (Jan, 11th)