STOGL
ROBOTICS

Wie einfach ist die System (Hardware) Integration mit ROS 2?

DR. DENIS

@

ROSCON DE 2023

Warum?

Und… Wie komplex ist es?

- In ros2_control gibt es integrierten Hardware Mocking!

Worum geht es eigentlich hier?

- In ros2_control gibt es integrierten Hardware Mocking!

- Du kannst das anstatt Simulation nutzen!

Worum geht es eigentlich hier?

- In ros2_control gibt es integrierten Hardware Mocking!

- Du kannst das anstatt Simulation nutzen!

- Es ist auch nützlich um einzelne Module im System zu Entwickeln!

Worum geht es eigentlich hier?

- In ros2_control gibt es integrierten Hardware Mocking!
- Du kannst das anstatt Simulation nutzen!
- Es ist auch nützlich um einzelne Module im System zu Entwickeln!
- Man kann komplette Szenarien testen und Debuggen!

Worum geht es eigentlich hier?

- In ros2_control gibt es integrierten Hardware Mocking!
- Du kannst das anstatt Simulation nutzen!
- Es ist auch nützlich um einzelne Module im System zu Entwickeln!
- Man kann komplette Szenarien testen und Debuggen!
- Als Hilfe kannst du Vorlagen aus RTW nutzen!

Es ist einfach!

- In ros2_control gibt es integrierten Hardware Mocking!
- Du kannst das anstatt Simulation nutzen!
- Es ist auch nützlich um einzelne Module im System zu Entwickeln!
- Man kann komplette Szenarien testen und Debuggen!
- Als Hilfe kannst du Vorlagen aus RTW nutzen!
- **Danke für die Aufmerksamkeit! Gibt es Fragen?**

Die Herausforderungen

„Ich muss neues Hardware muss ins System einbinden“

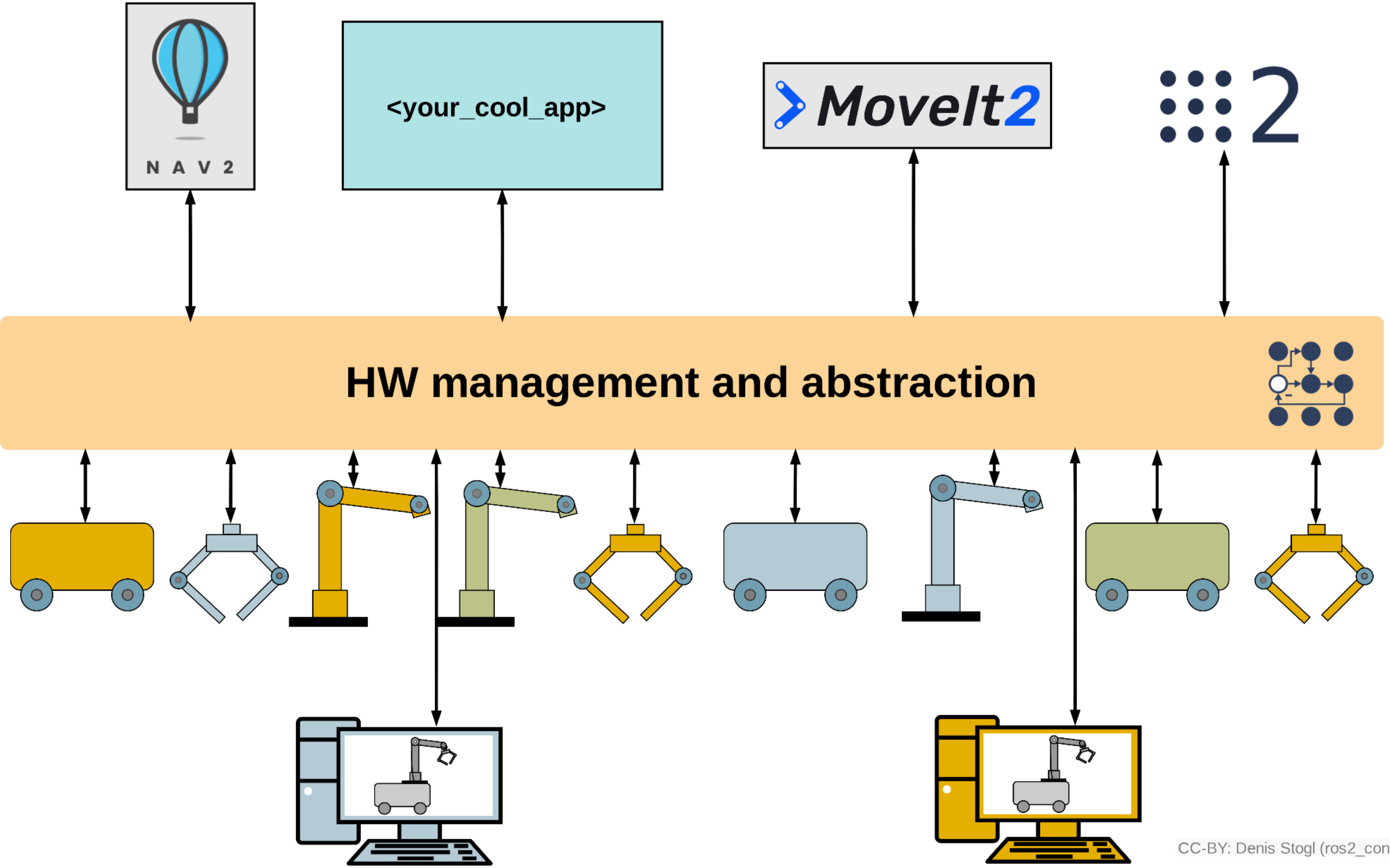
„Hmmm, wir brauchen ein neues Modul, aber das könnte den kompletten Stack zerschießen…“

„Oh, unsere Bahnplanung braucht definitiv Optimierung, es muss schneller und zuverlässiger werden!“

„Mein Behavior Trees ist kaputt, ich brauche dringend remote Hilfe!“

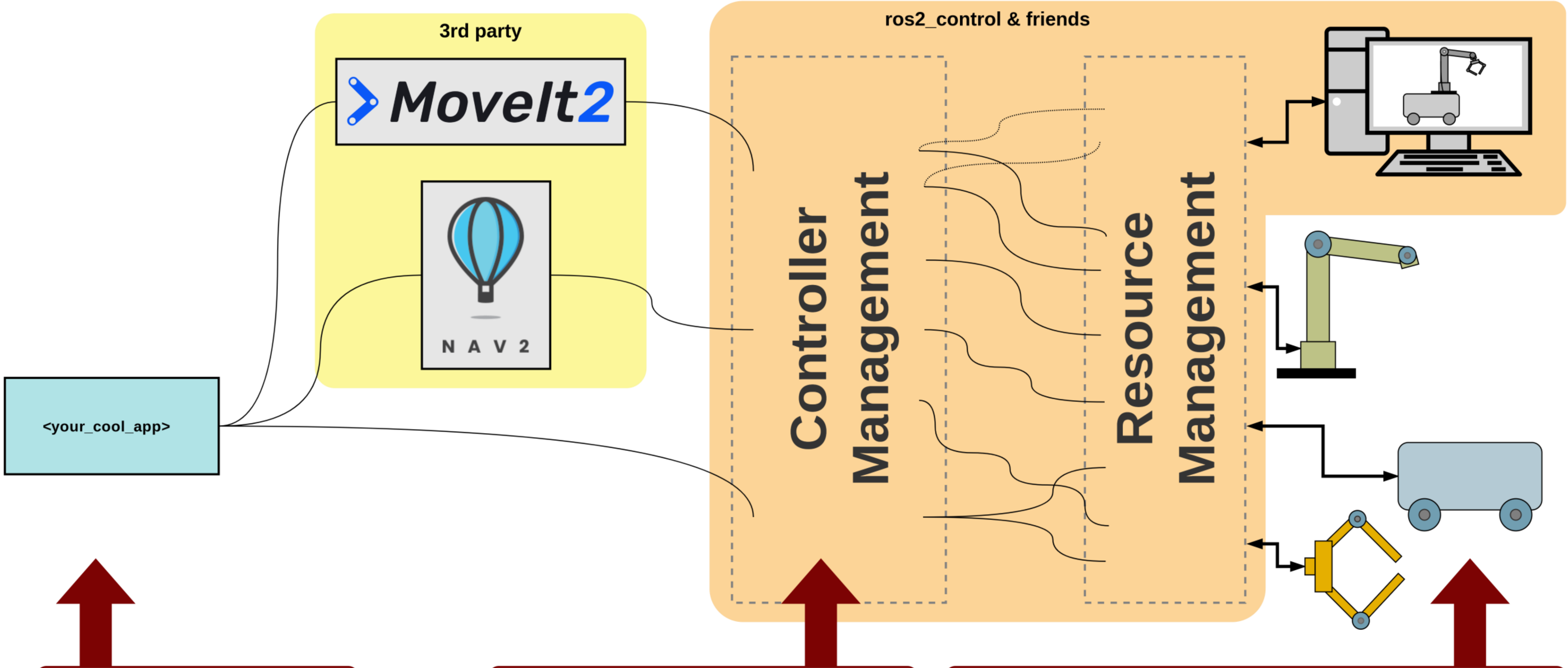
„Aaaaa, das macht kein Spaß mehr, wegen einen blöden Parameter den Roboter neu starten!“

1. Hardware… Abstraktion!



CC-BY: Denis Stogl (ros2_control)

1. Hardware → Mock HW



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

1. Hardware… Ergebnis!

1. Hardware… Ablauf

1. ROS-Paket Struktur (RTW - Dokumentation)
2. Roboterbeschreibung erstellen (RTW)
   - ➢Testen ☑
3. URDF Anpassung + <ros2_control>-Tag
   - ➢Testen ☑
4. Roboter-Bringup Dateien erstellen (RTW)
   - ➢Anpassen von Reglerkonfiguration ▯
   - ➢Testen ☑
5. Anstoßen + Party machen 🎉
6. Roboter-spezifische Regler erstellen (RTW)
   - ➢Entwickeln ▯
   - ➢Testen mit Mock HW ☑
7. Hardware Interface für den Roboter erstellen (RTW)

```
<manufacturer>_<robot_name>[_<type>]/                     # Repository
├── <manufacturer>[_<robot_name>[_<type>]|_manipulators]/       # Meta-package of the robot
│   ├── CMakeLists.txt
│   └── package.xml
├── <robot_name>_bringup/                              # Launch and config files for starting the robot using ros2_control
│   ├── [CMakeLists.txt]                               # if ament_cmake is used (recommended)
│   ├── package.xml
│   ├── [setup.py]                                     # if ament_python is used
│   ├── [setup.cfg]                                    # if ament_python is used
│   ├── config/
│   │   ├── <robot_name>_controllers.yaml              # Controllers' configuration for ros2_control
│   │   ├── <robot_name>_forward_position_publisher.yaml  # Setup test publisher for forward position controller
│   │   └── <robot_name>_joint_trajectory_publisher.yaml  # Setup test publisher for joint trajectory controller
│   └── launch/
│       ├── <robot_name>.launch.py                     # Robot's default launch file
│       ├── test_forward_position_controller.launch.py # Start test publisher for forward position controller
│       └── test_joint_trajectory_controller.launch.py # Start test publisher for joint trajectory controller
├── <manufacturer|robot_name>_description/             # Robot's description files
│   ├── [CMakeLists.txt]                               # if ament_cmake is used (recommended)
│   ├── package.xml
│   ├── [setup.py]                                     # if ament_python is used
│   ├── [setup.cfg]                                    # if ament_python is used
│   ├── config/                                        # general YAML files for a robot
│   │   └── <robot_name>_<someting_specific>.yaml
│   ├── launch/                                        # launch files related to testing robots' description
│   │   └── test_<robot_name>_description.launch.py
│   ├── meshes/                                        # meshes used in <robot_name>_macro.urdf.xacro
│   │   ├── collision
│   │   │   └── <robot_name|robot_model>               # meshes are sorted by robot name or model
│   │   │       ├── <link_xy>.stl
│   │   │       └── ...
│   │   └── visual
│   │       └── <robot_name|robot_model>
│   │           ├── <link_xy>.dae
│   │           └── ...
│   ├── rviz/                                          # rviz display configurations
│   │   └── <robot_name>_default.rviz
│   └── urdf/                                          # URDF file for the robot
│       ├── common.xacro                               # Common XACRO definitions
│       ├── <robot_name>.urdf.xacro                    # Main URDF for a robot - loads macro and other files
│       └── <robot_name|robot_model>
│           ├── <robot_name>_macro.xacro               # Macro file of the robot - can add prefix, define origin, etc.
│           └── <robot_name>_macro.ros2_control.xacro  # URDF-part used to configure ros2_control
└── <robot_name>_hardware_interface/                # Implementation of the ros2_control interface
│   ├── CMakeLists.txt
│   ├── package.xml
│   ├── include/
│   │   └── <robot_name>_hardware_interface/
│   │       └── <robot_name>_<something_specific>.hpp
│   └── src/
│       └── <robot_name>_<something_specific>.cpp
└── <manufacturer>_controllers/                 # Implementation of hardware specific controllers
    ├── CMakeLists.txt
    ├── package.xml
    ├── include/
    │   └── <manufacturer>_controllers/
    │       └── <robot_specific_controller>.hpp
    └── src/
        └── <robot_specific_controller>.cpp
```

1. Hardware… um sonst?

1. Hardware… um sonst?

2. Neues Modul… Ergebnis

3. Bahnplanung... Best-Practices

1. Erstelle ein Separates *ros2_control_support* Paket fur deinen Roboter
   - ➢Entferne <ros2_control>-Tag aus der Beschreibungspaket
   - ➢ *ros2_control_support* hängt von Beschreibungspaketen ab (nicht umgekehrt)
   - ✓Unterstützung mehreren Roboterschnittstellen
   - ✓Geringeres Fußabdruck an den Systemen, die nur den Roboter visualisieren
   - – kuka_exprimental (rolling-overview); motoman (separate-ros2-control-launch)
2. Erstelle immer ein separates MoveIt2 / Nav2 “common” oder “support” Paket
   - ➢ggf. Füge spezielle *SRDFs* in Roboterbeschreibung
   - ✓Warum kompliziert, wenn es geht einfach!

3. Bahnplanung… Best-Practices

4. Team Arbeit… Vorbereitung

1. Erstelle ein Separates *ros2_control_support* Paket fur deinen Roboter
   - ➢ Entferne <ros2_control>-Tag aus der Beschreibungspaket
   - ➢ *ros2_control_support* hängt von Beschreibungspaketen ab (nicht umgekehrt)
   - ✓ Unterstützung mehreren Roboterschnittstellen
   - ✓ Geringeres Fußabdruck an den Systemen, die nur den Roboter visualisieren
   - – kuka_exprimental (rolling-overview); motoman (separate-ros2-control-launch)
2. Erstelle immer ein separates MoveIt2 / Nav2 “common” oder “support” Paket
   - ➢ ggf. Füge spezielle *SRDFs* in Roboterbeschreibung
   - ✓ Warum kompliziert, wenn es geht einfach!

4. Team Arbeit… Ablauf

1. 🤖♡🔥⏲
2. Prüfe Stand deiner Repositorien
   - ➢ und gleiche es mit Remote server aus ❗
   - ➢ Ist dein „.repos“ – Datei auch aktuell ❓
   - ✓ vcs tool ist dein sehr guter Freund
3. Rufe deinen Kollegen an 📞
   - ➢ Slack geht auch
   - ✓ Gebe einfach name deines Branches weiter
4. Jetzt kann man das Ganze einfach Remote starten
   - ➢ Containers, VMs, usw. natürlich vorausgesetzt ❗

PRO Tipp: ros_team_workspace ist dafür gebaut, dass wir schnell neue WS aussetzen und replizieren können!

4. Team Arbeit...

5. Szenario Parametrierung

Der Schmerz ist mega Groß!

Und? Wie ist es?

DR. DENIS

@

ROSCON DE 2023