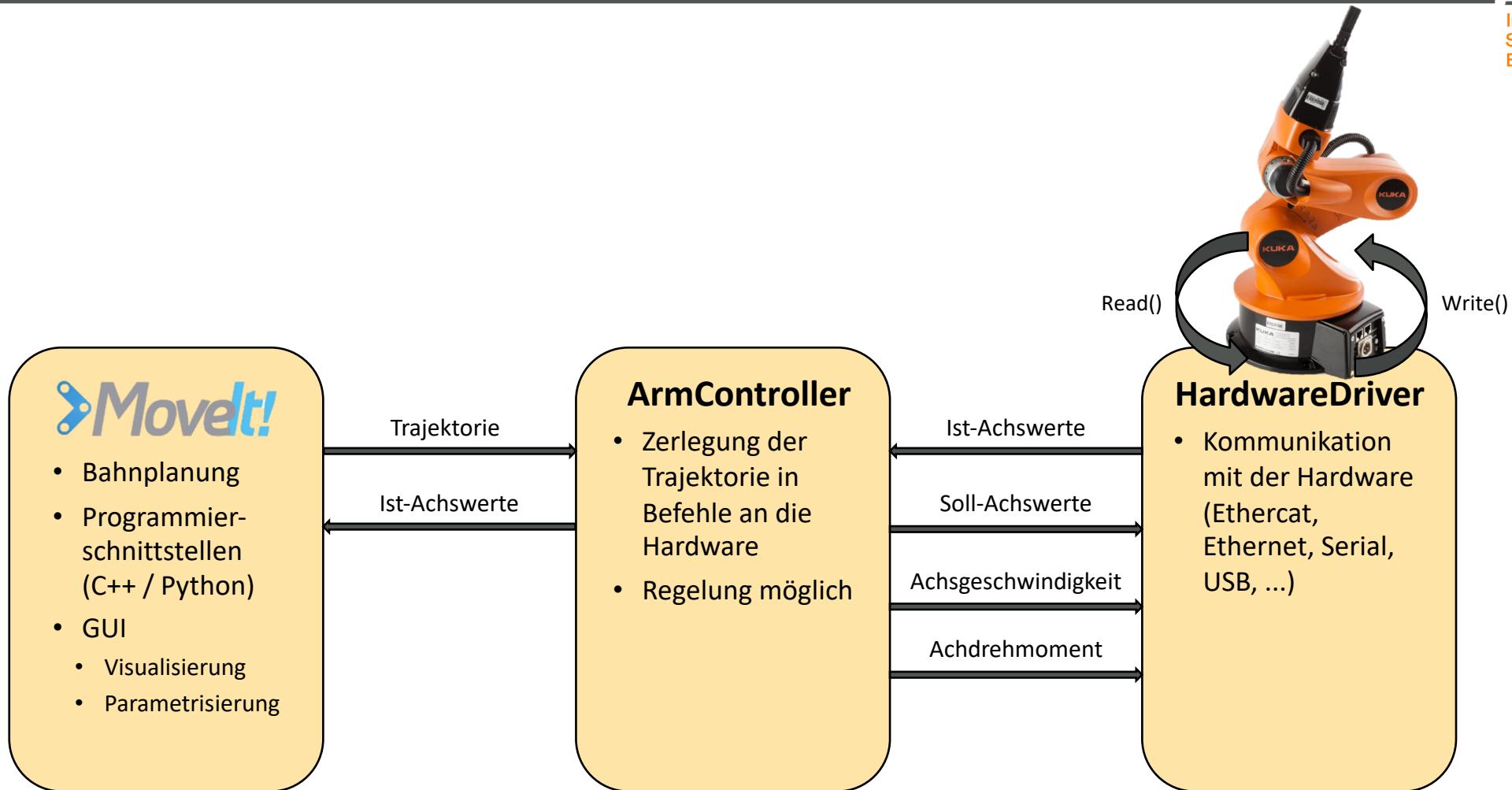




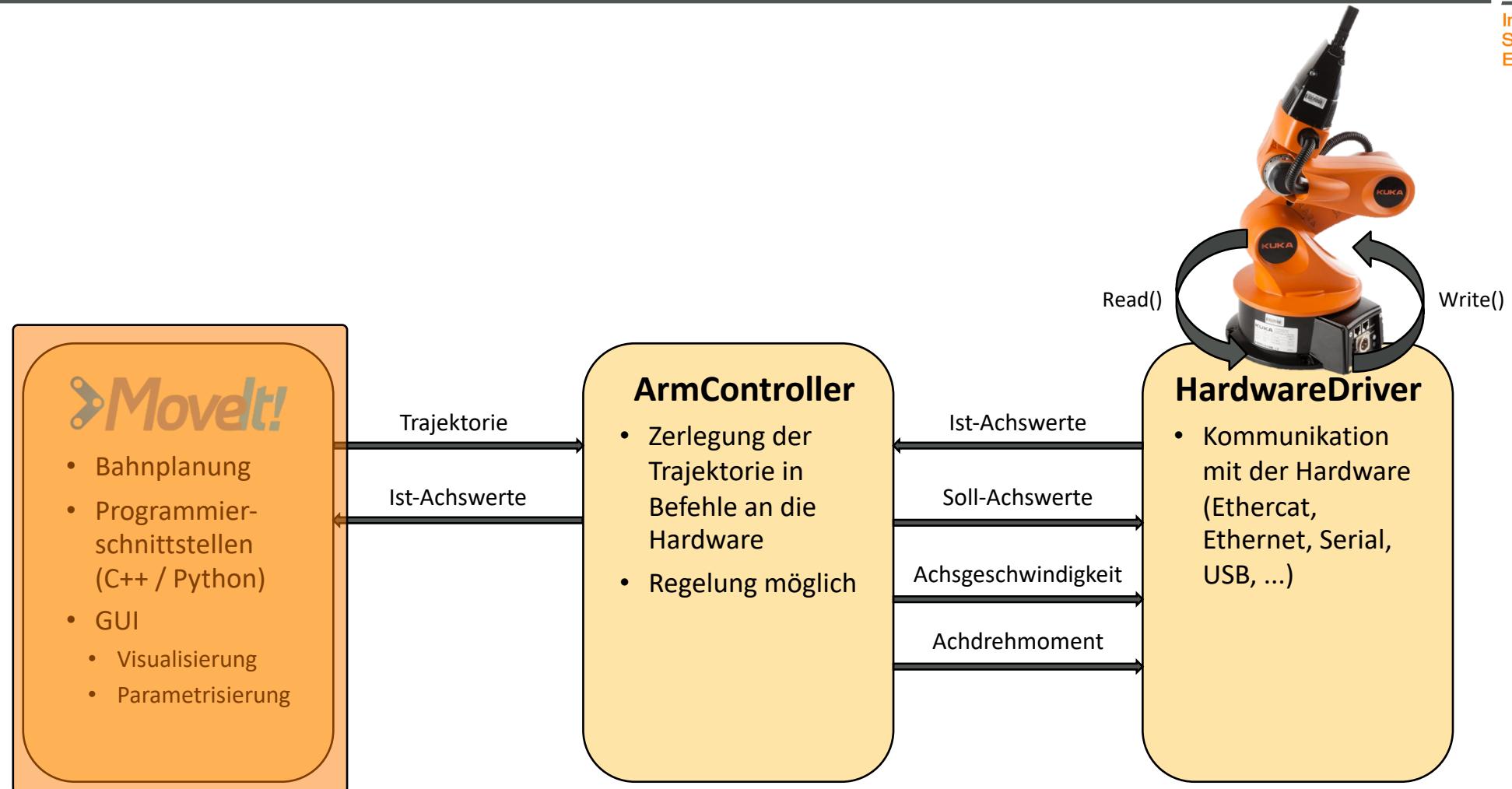
ROS-Arm-Stack



Überblick: Roboterarme in ROS

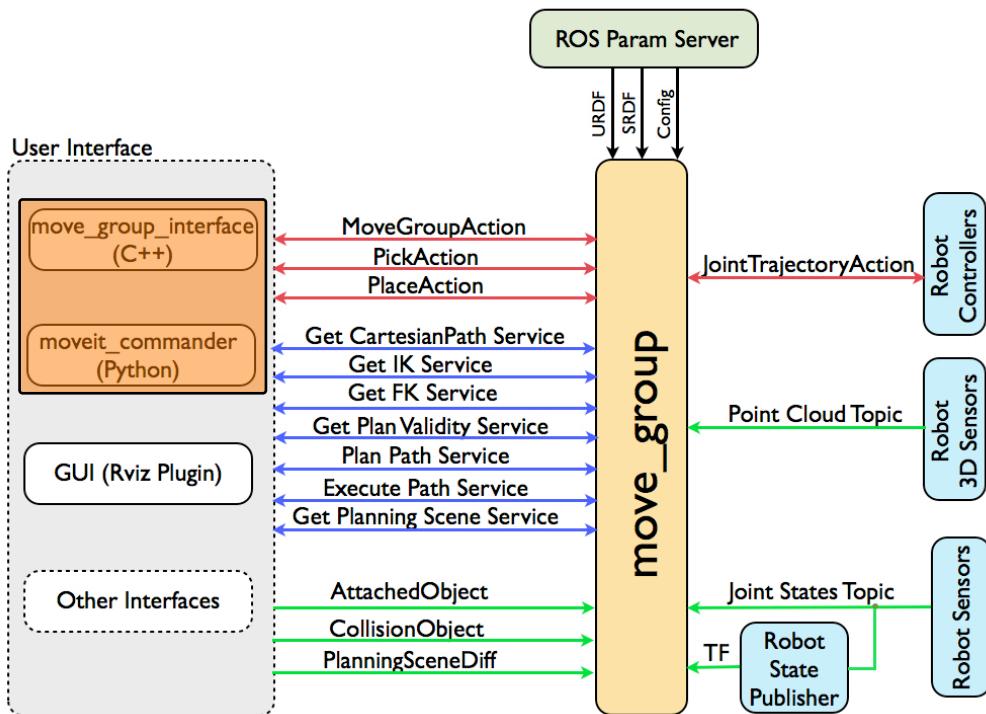


Überblick: Roboterarme in ROS



- Movelt! ermöglicht die Angabe der folgenden kinematischen Bedingungen (Constraints) :
 - **Position constraints**: Beschränkt die Position einer Joints im Raum
 - **Orientation constraints**: Beschränkt die Orientierung eines Gelenks
 - **Visibility constraints**: Bedingungen, dass ein Punkt innerhalb des Kegels des Sensors liegen muss
 - **Joint constraints**: Beschränkt die Achswinkel eines Gelenks
 - **User-specified constraints**
- Movelt! nutzt Flexible Collision Library (FCL) die folgende Technologien unterstützen:
 - Meshes
 - Primitive Formen (Quader, Zylinder, Kegel)
 - Octomap (3D Sensor Information)

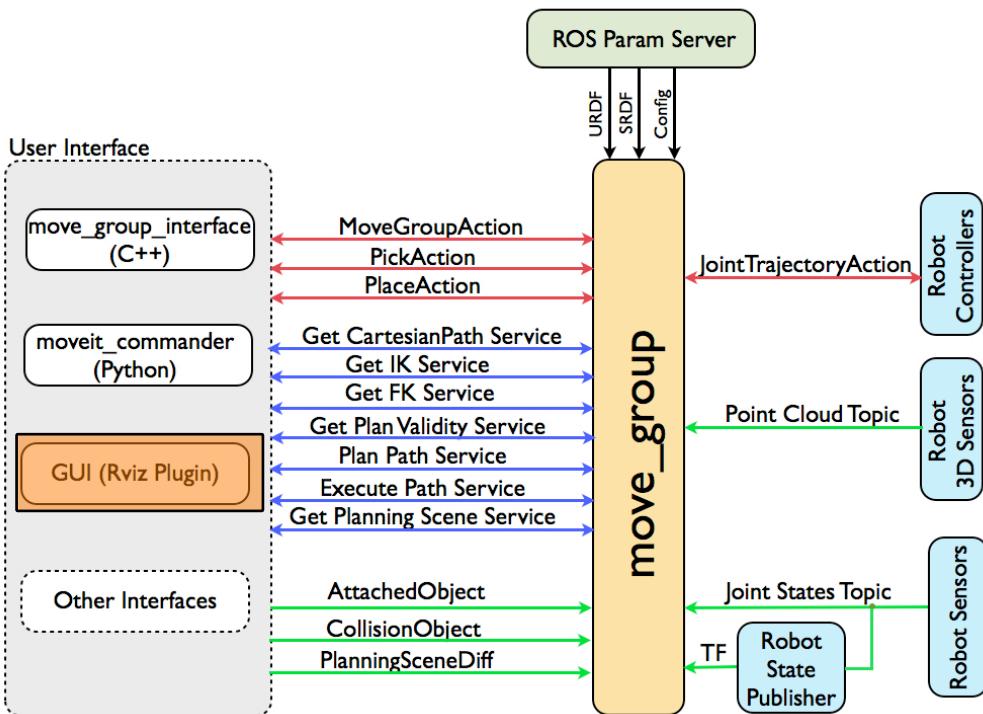
MoveIt! – move_group_interface / moveit_commander



move_group_interface / moveit_commander

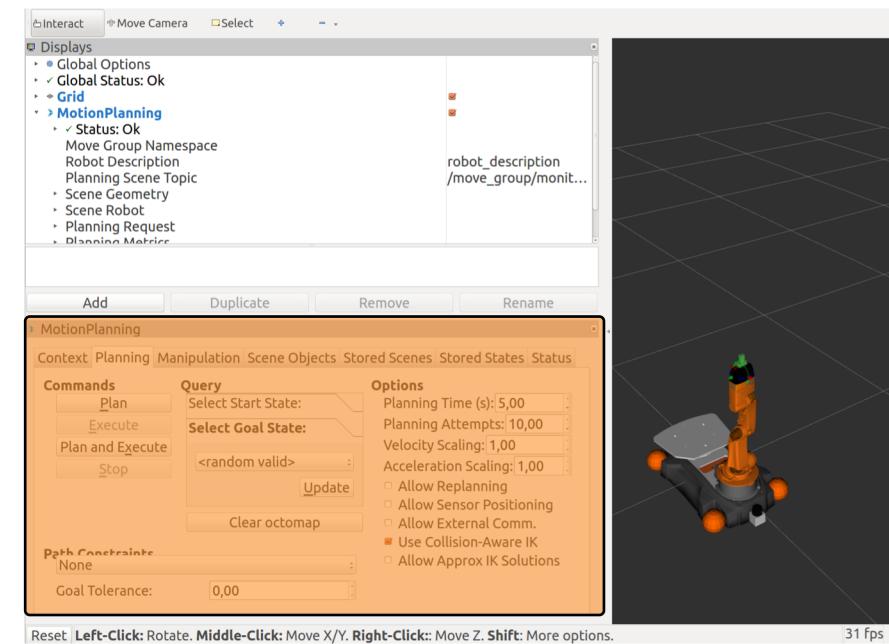
- Setzten von Zielpositionen im Achs- oder Kartesischen-Raum
- Erstellen von Bahnplanungen
- Hinzufügen von Objekten in die Umwelt (z. B. Hindernisse)
- Hinzufügen von Objekten an den Roboter (z. B. mit Greifer aufgenommenes Objekt)
- Ausführung der geplanten Bahn

MovIt! – GUI

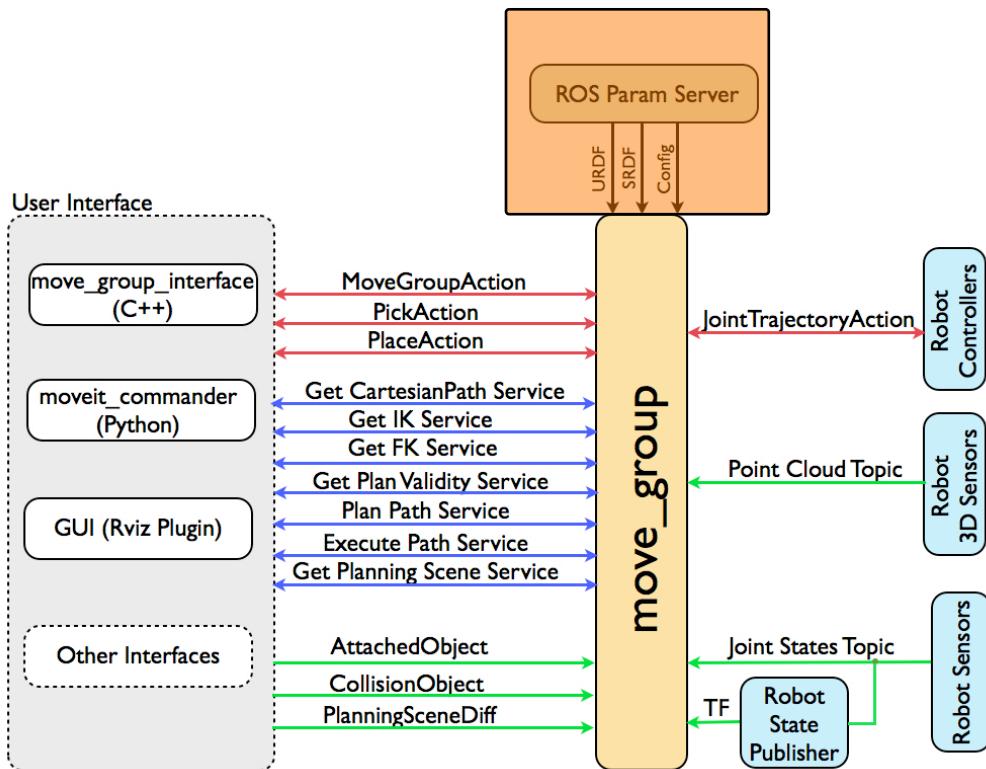


RVIZ

- MotionPlanning Plugin zum Planen und anzeigen von Bewegungen

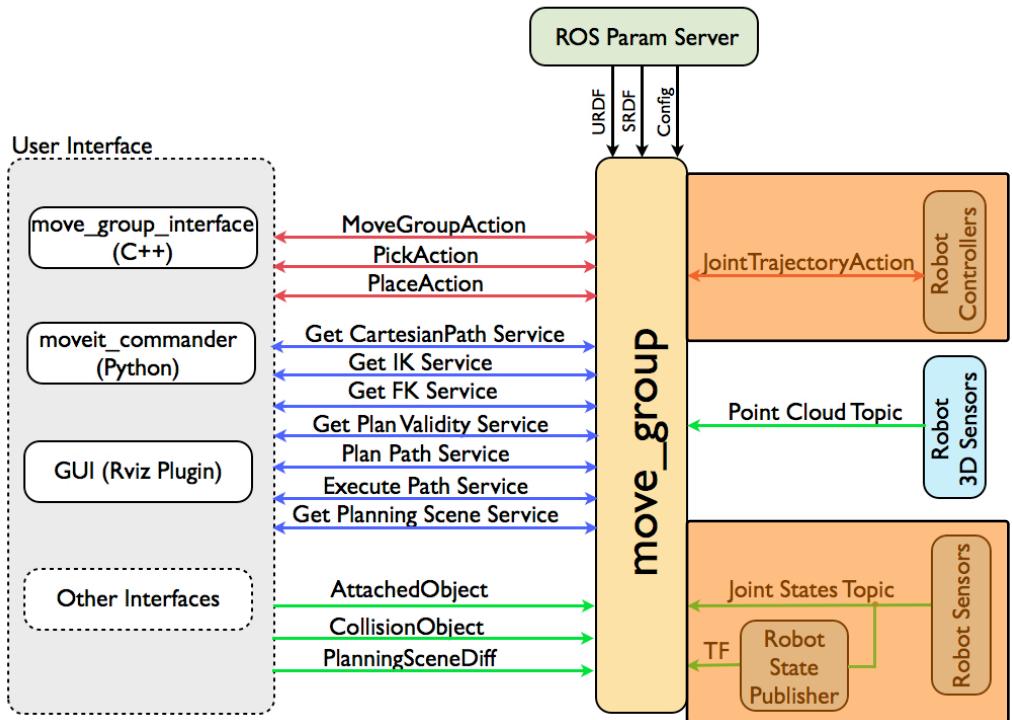


MoveIt! – ROS Param Server



ROS Param Server

- **URDF:** Zugriff auf die *robot_description*
- **SRDF :** Zugriff auf die *robot_description_semantic*
- **Configuration:** Zugriff auf zusätzliche Parameter wie z. B. Achsgrenzen, Kinematik, Bahnplanungsparameter, ...



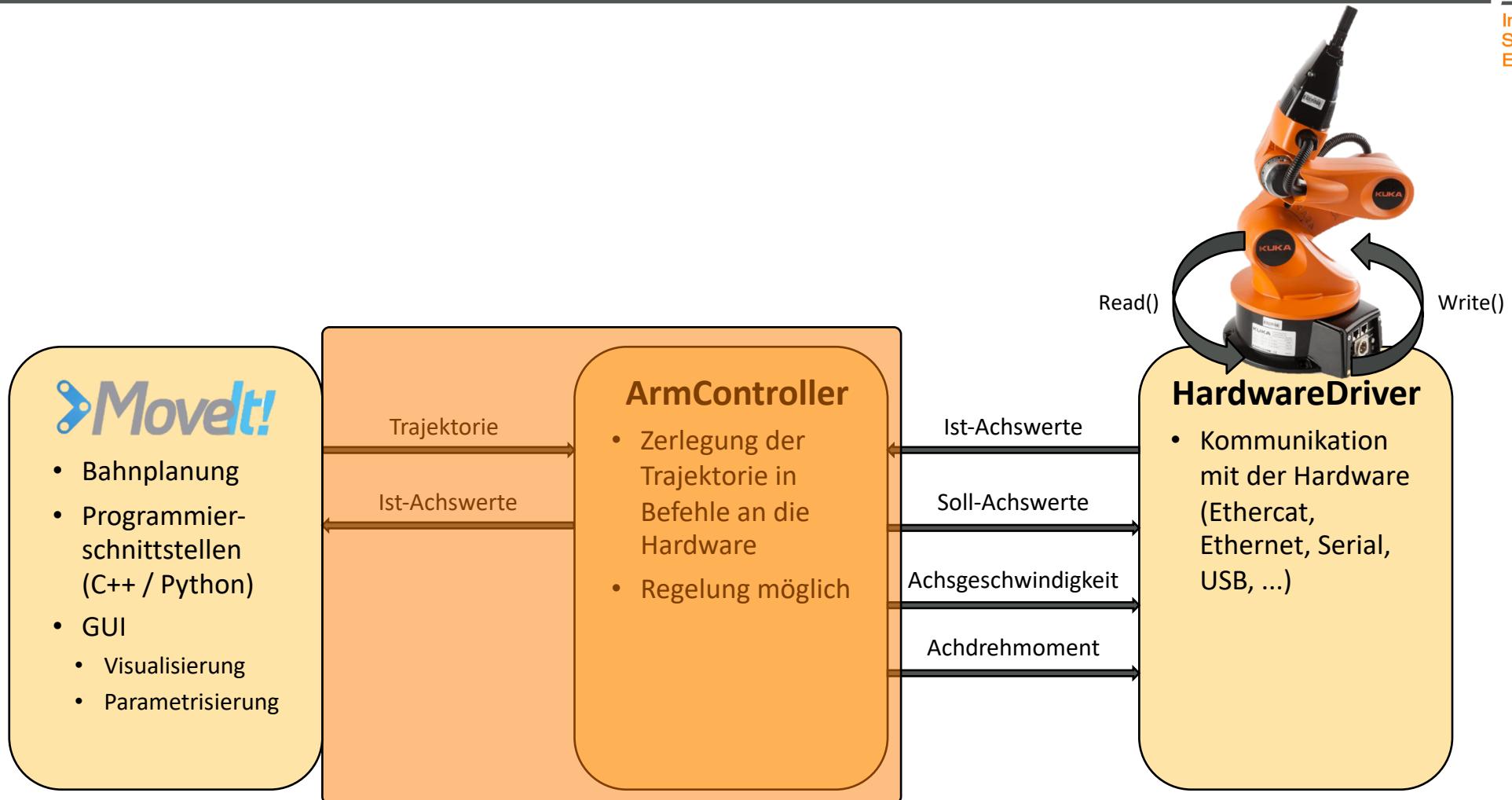
RobotController

- Direkter Zugriff auf die Controller des Roboters
- Nutzt JointTrajectoryAction zur Übertragung von Trajektorien zum Roboter

RobotSensors

- Schnittstelle zum Abgreifen der aktuellen Achsstellungen
- Darstellung der Transformationen des Roboters im TF-Framework

Überblick: Roboterarme in ROS



- Controller zum Ausführen von Trajektorien mehrerer Achsen im Achsraum
- Trajektorien Repräsentation
 - **Linear:** Nur Position
 - **Cubic:** Position und Geschwindigkeit
 - **Quintic:** Position, Geschwindigkeit und Beschleunigung
- Der Controller unterstützt mehrere Hardware-Schnittstellen
 - **Position-Controlled** (rad): Soll-Position zur Zeit x
 - **Velocity-Controlled** (rad/sec): Soll-Geschwindigkeit zur Zeit x
 - **Effort-Controlled** (Nm): Soll-Drehmoment zur Zeit x

FollowJointTrajectory – Action

```
JointTrajectory trajectory
JointTolerance[] path_tolerance
JointTolerance[] goal_tolerance
duration goal_time_tolerance
---
int32 error_code
int32 SUCCESSFUL = 0
int32 INVALID_GOAL = -1
int32 INVALID_JOINTS = -2
int32 OLD_HEADER_TIMESTAMP = -3
int32 PATH_TOLERANCE_VIOLATED = -4
int32 GOAL_TOLERANCE_VIOLATED = -5
string error_string
---
Header header
string[] joint_names
JointTrajectoryPoint desired
JointTrajectoryPoint actual
JointTrajectoryPoint error
```

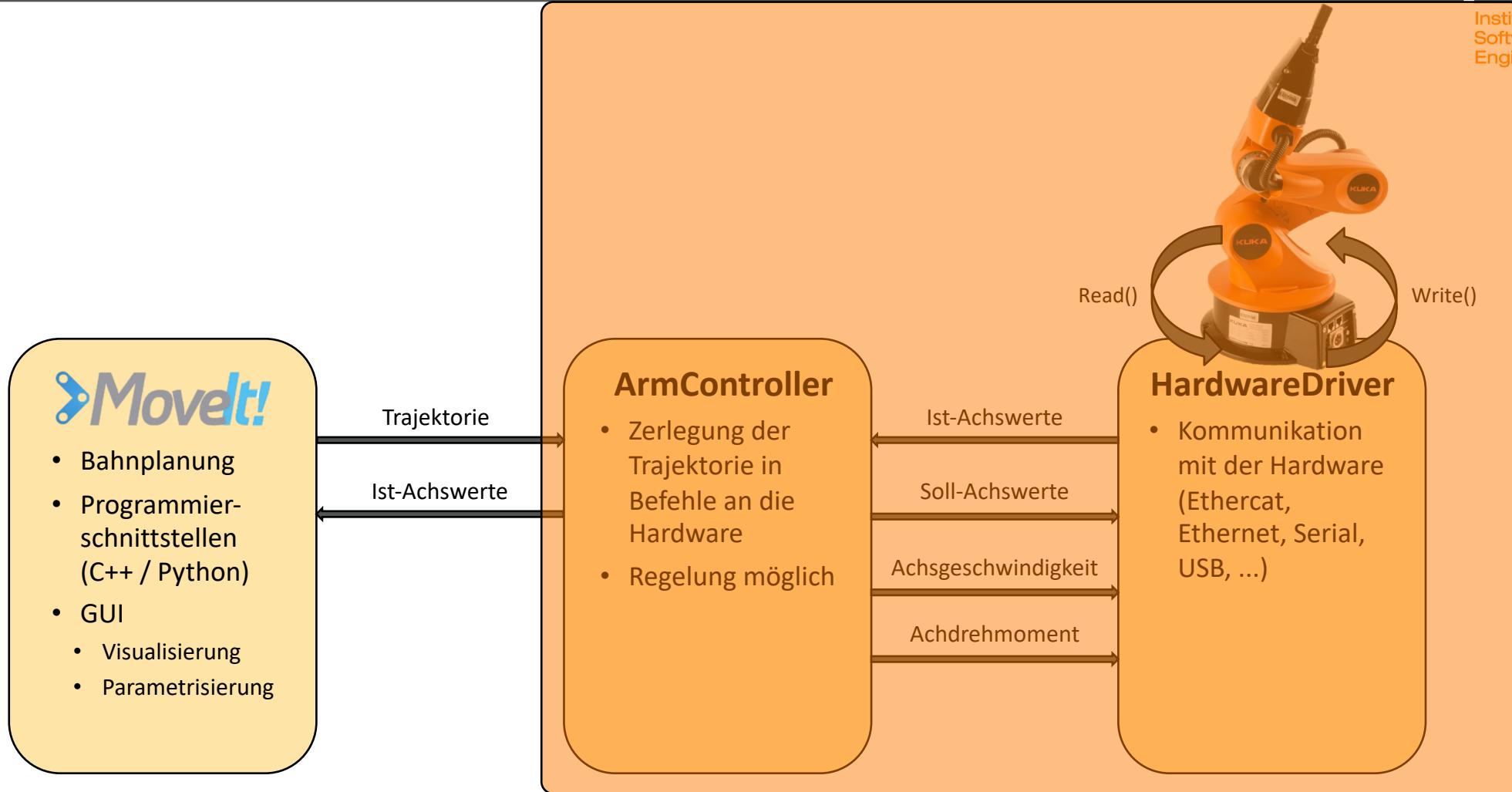
JointTrajectory.msg

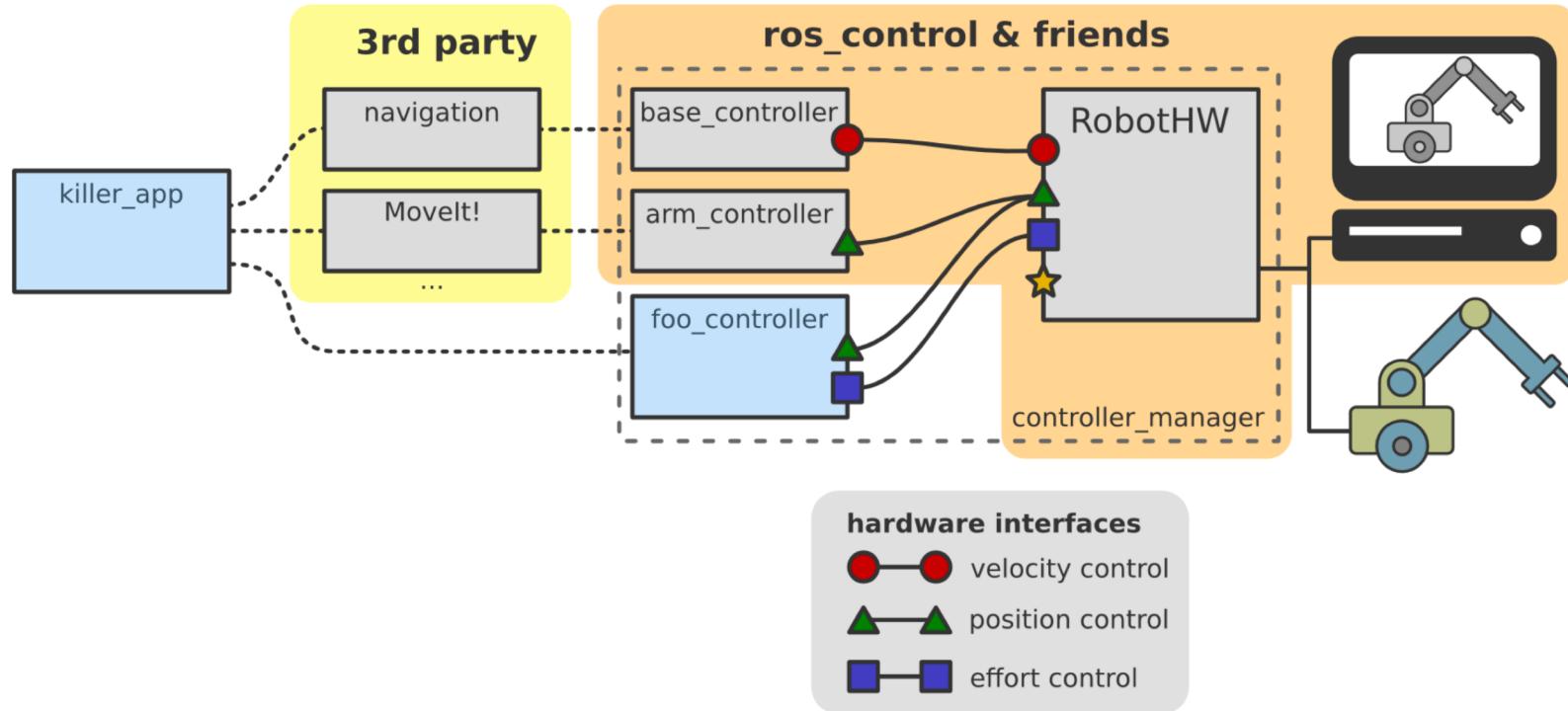
```
Header header
string[] joint_names
JointTrajectoryPoint[] points
```

JointTrajectoryPoint.msg

```
float64[] positions
float64[] velocities
float64[] accelerations
float64[] effort
duration time_from_start
```

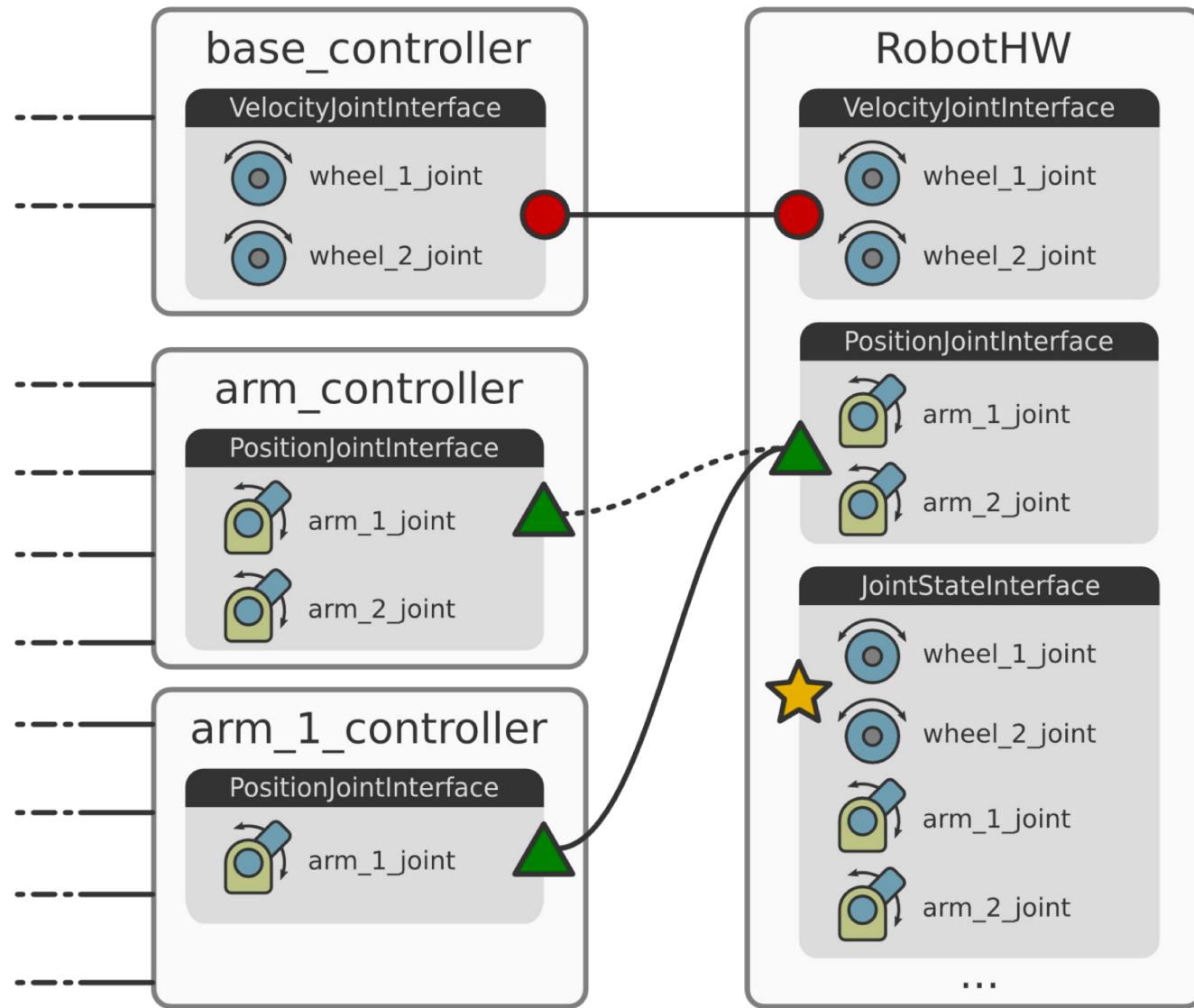
Überblick: Roboterarme in ROS





- + Controller abgesetzt vom Roboter
- + Ready-To-Use Controller
- + Einfach eigene Controller schreiben
- + Kompatibel mit vielen Tools
- + Echtzeitfähig (nur mit ROS2)

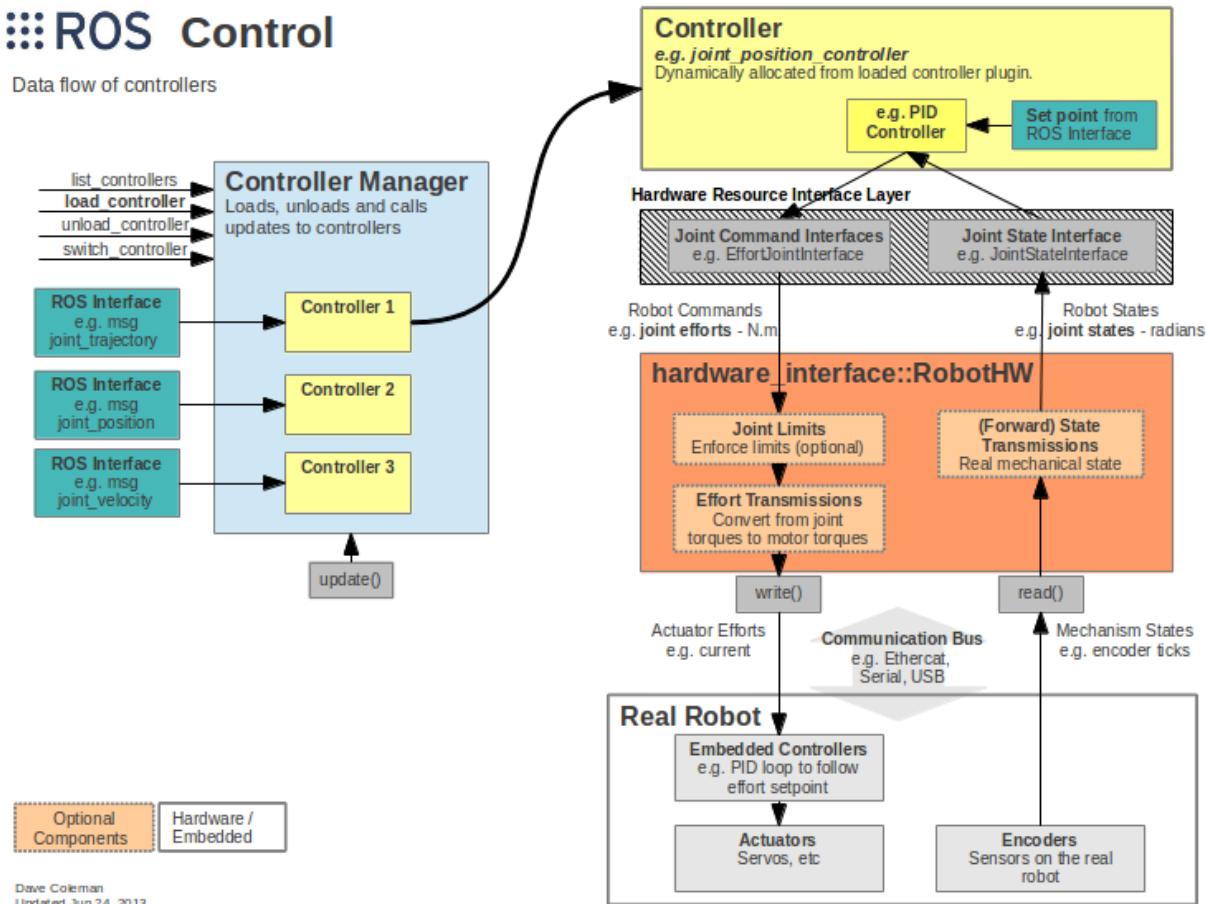
ros_control



ros_control – Architektur

ROS Control

Data flow of controllers



- Input:

- Achswerte der Motorenencoder
- Sollwert der Achse

- Ausgabe:

- Hardwarebefehle

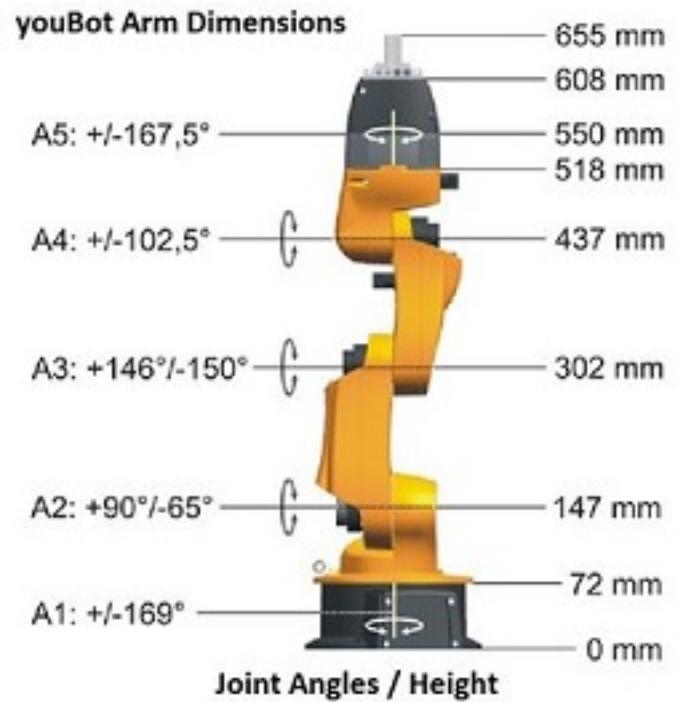
- Einheitliche Schnittstelle für den Einsatz von unterschiedlichen Controllern

- Einfaches Laden von Controllern durch den Controller Manager

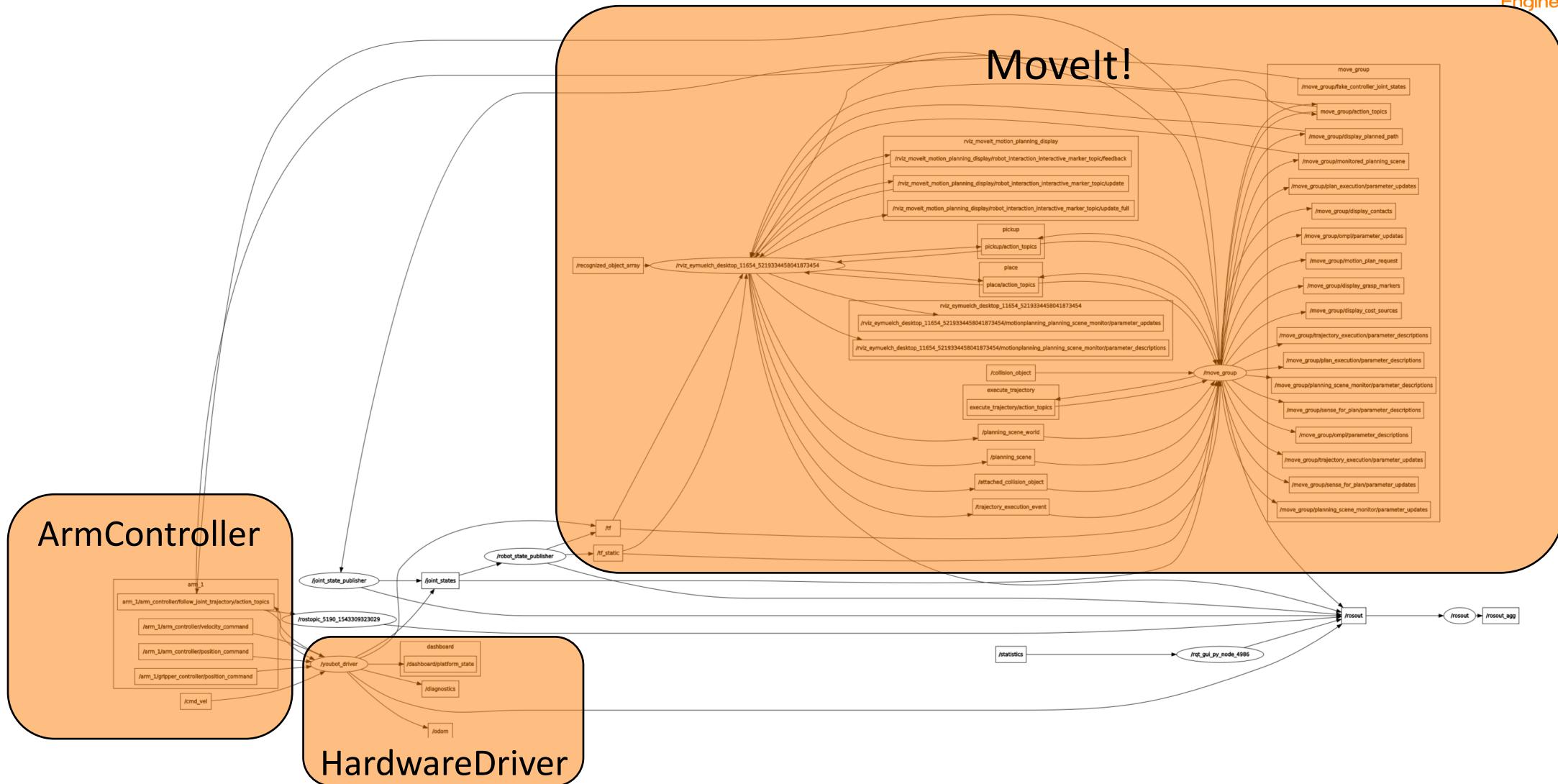


KUKA YOUBOT



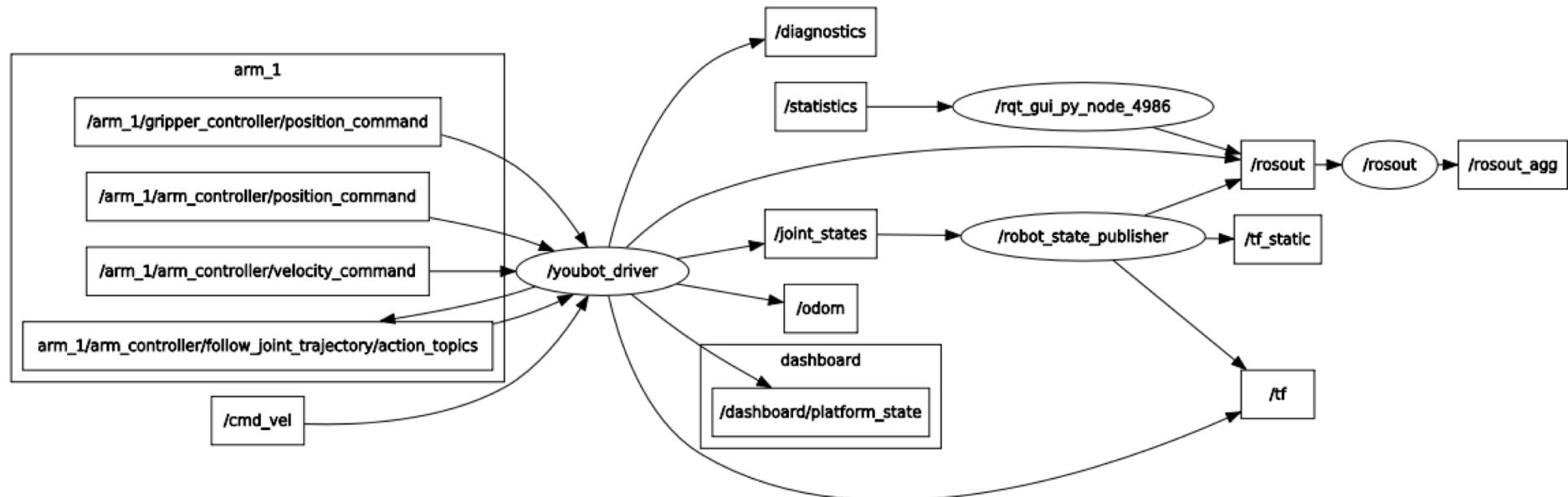


Youbot – Node Graph



Youbot – youbot_driver_ros_interface (echter Roboter)

- Starten des youbot_driver_ros_interface:
 - `roslaunch youbot_driver_ros_interface youbot_driver.launch`
- Startet sowohl den Hardware Treiber als auch den Arm- und Base-Controller



Youbot – youbot_gazebo_robot (simulierter Roboter)

- Starten des youbot_gazebo_robot:
 - `roslaunch youbot_gazebo_robot youbot.launch`
- Startet sowohl die Gazebo-Simulation als auch den Arm- und Base-Controller



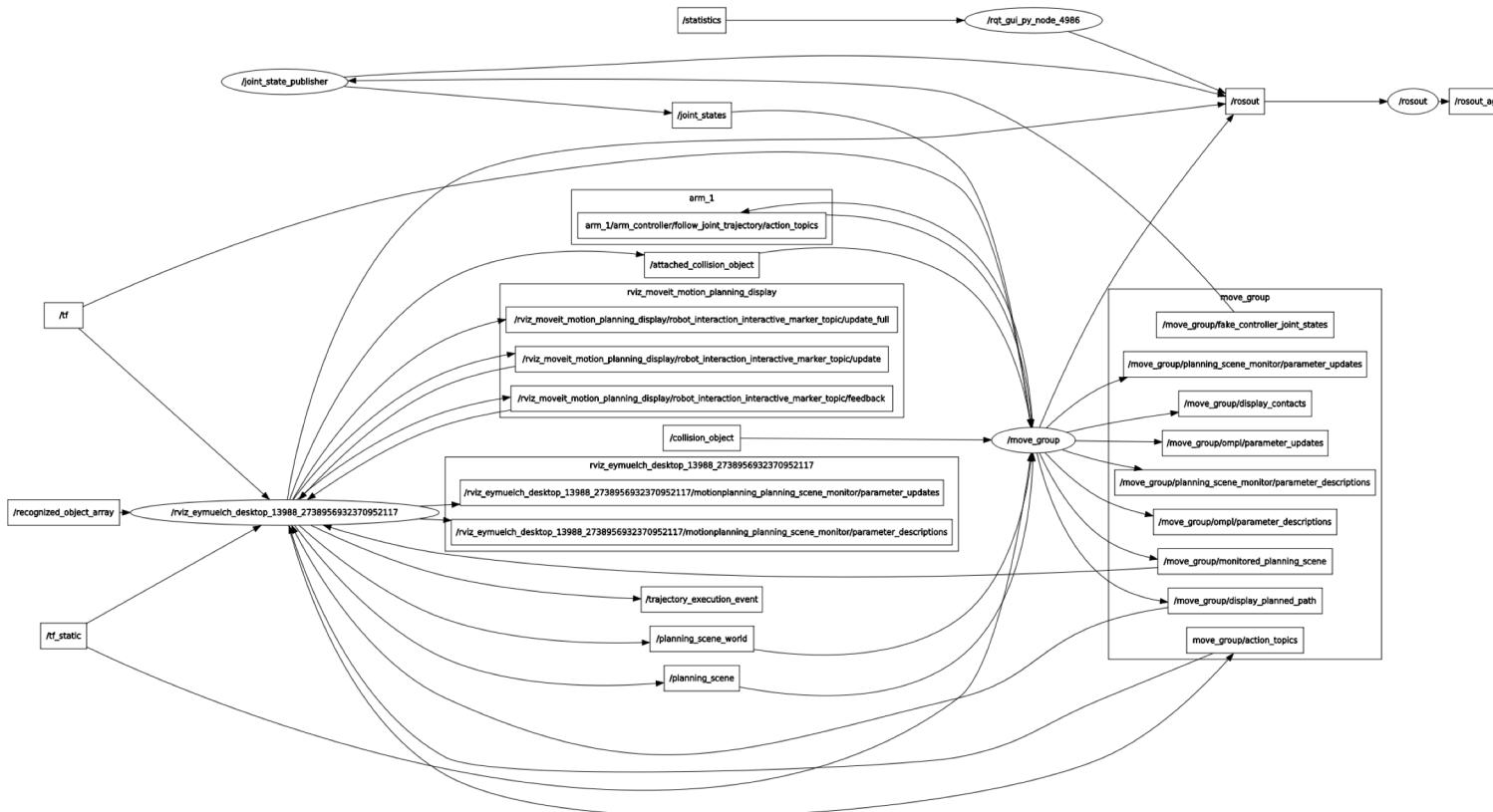
Gazebo-Simulation

Youbot – youbot_gazebo_robot (simulierter Roboter)

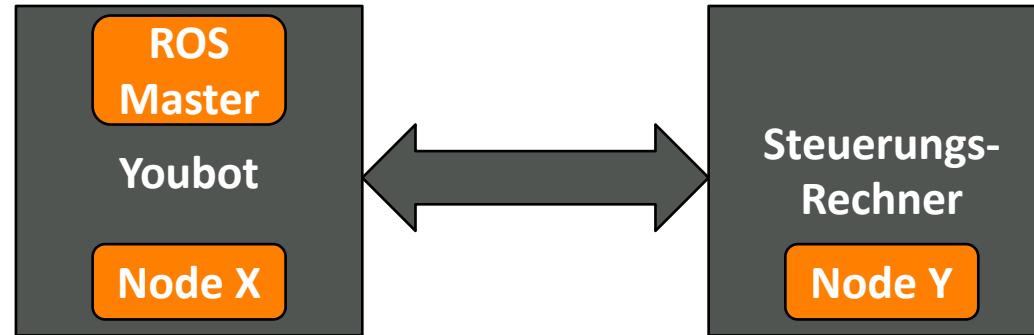


Youbot – youbot_moveit

- Starten von MoveIt! für den Youbot:
 - `roslaunch youbot_moveit demo.launch`
- Startet sowohl die move_group Nodes als auch die RVIZ Nodes (GUI)



- Beispiel: Steuerungs-Rechner zur Ansteuerung des Youbots



- Auf dem Youbot kann wie üblich der Ros Master mit dem Befehl `roscore` gestartet werden
- Auf dem Steuerungs-Rechner muss **in jedem Terminal** die Adresse des ROS Masters angegeben werden:

```
export ROS_MASTER_URI=http://<IP des ROS-Master>:11311
```