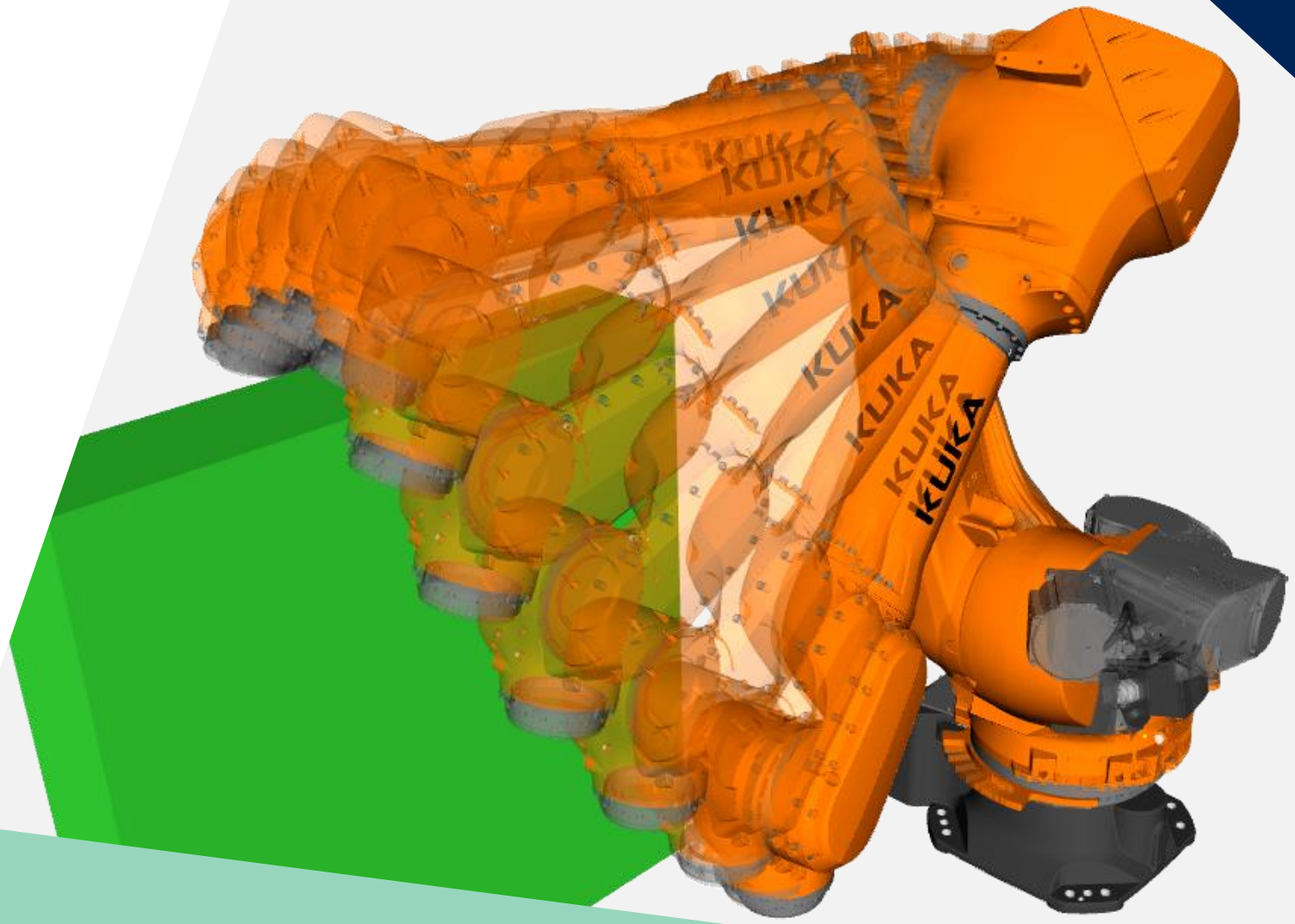
FZI Forschungszentrum Informatik

Tracing ros2_control

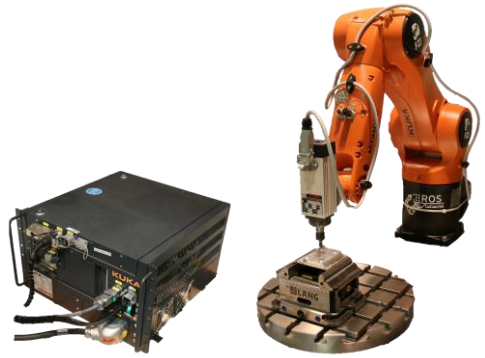
Erfahrungen aus einem neuen Kuka-Treiber

Robert Wilbrandt, Georg Heppner,
Tristan Schnell



Warum ein neuer Treiber?

KR10 R900 sixx



KR50 R2100

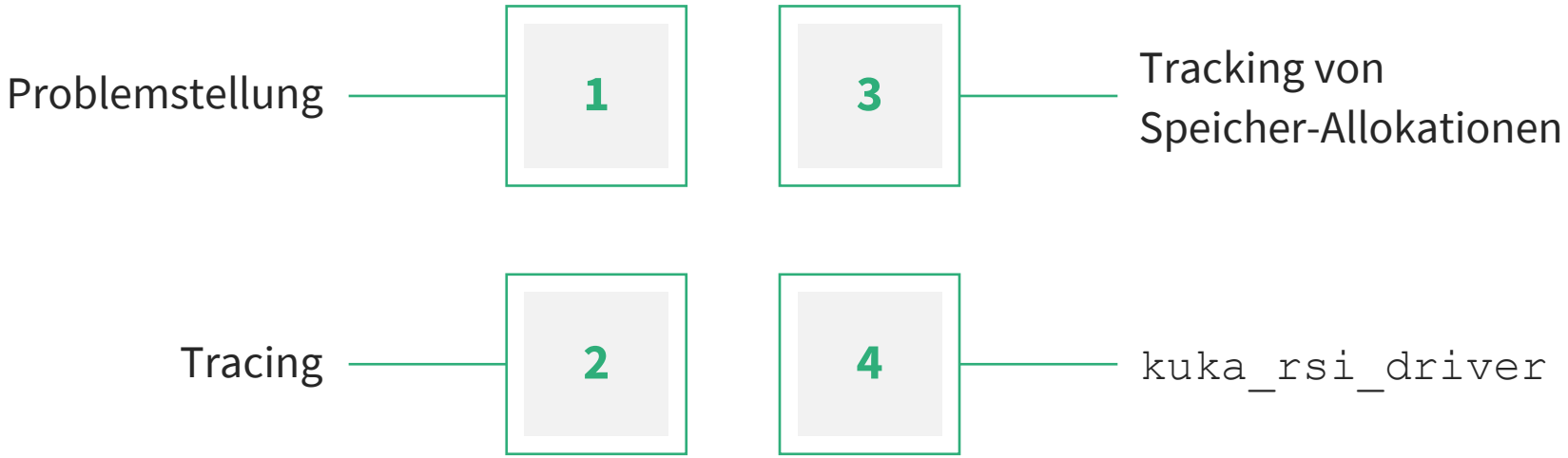
Robotisches Fräsen

- Robust gegenüber Jitter und Verzögerungen
- Stabil auch ohne RT-Patches

Montage

- Integration mit anderen Robotern
- Integration weiterer Sensoren & Aktoren

Agenda


Problemstellung
1
3
Tracking von
Speicher-Allokationen
Tracing
2
4
kuka_rsi_driver

Kurzeinführung RSI

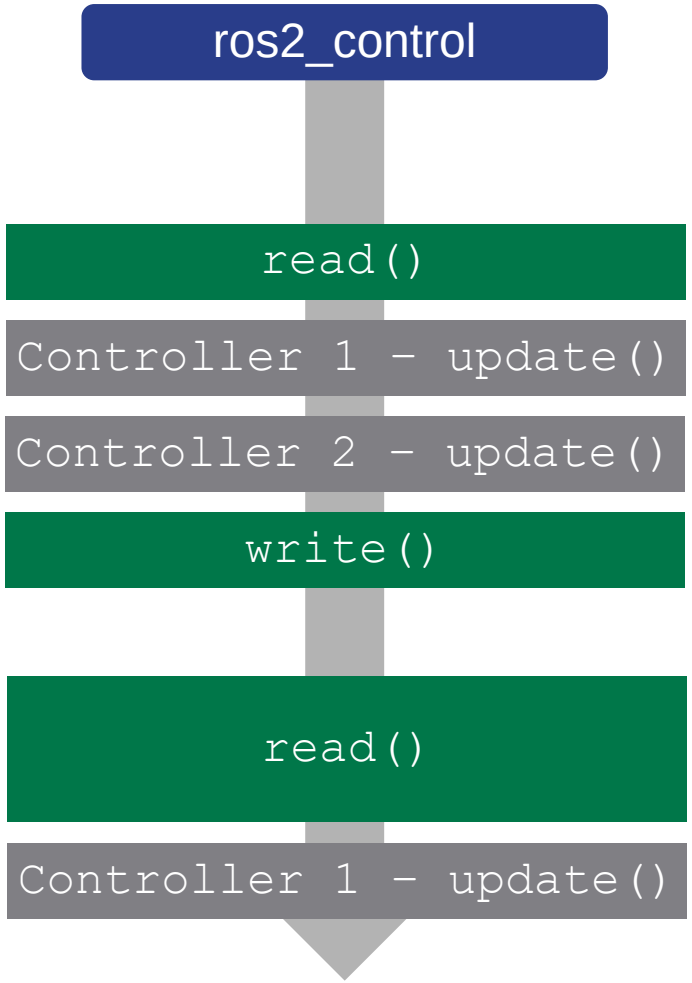
UDP Server

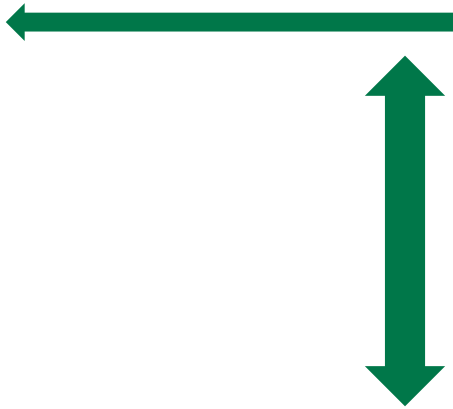
```
<Rob TYPE="KUKA">
  <Rist X="149.0" Y="19.5" … />
  <Rsol X="150.0" Y="20.0" … />
  <Delay D="42" />
  […]
</Rob>
```

250Hz

```
<SEN Type="SensorName">
  <AK A1="0.0" A2="0.1" … />
</Sen>
```

Basisstruktur

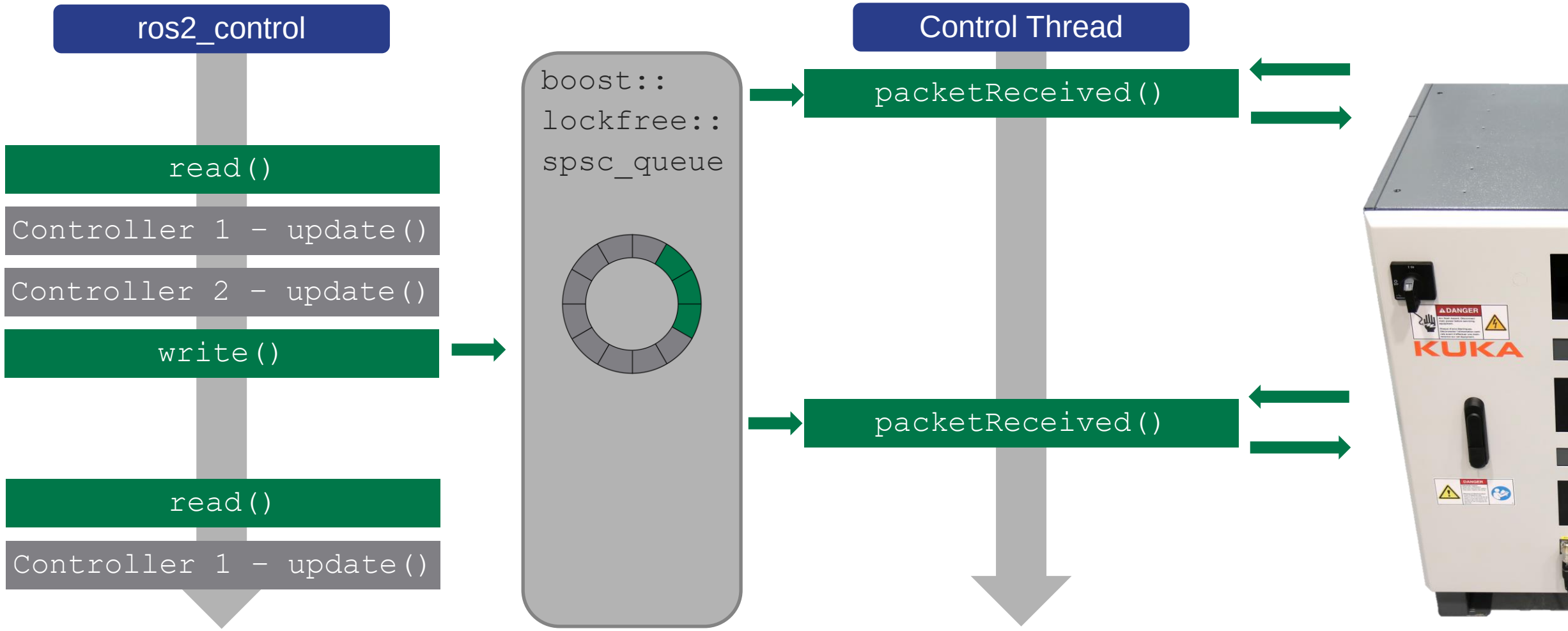




Timing abhängig von anderen Komponenten

Regelzyklus verzögert bis Paket kommt

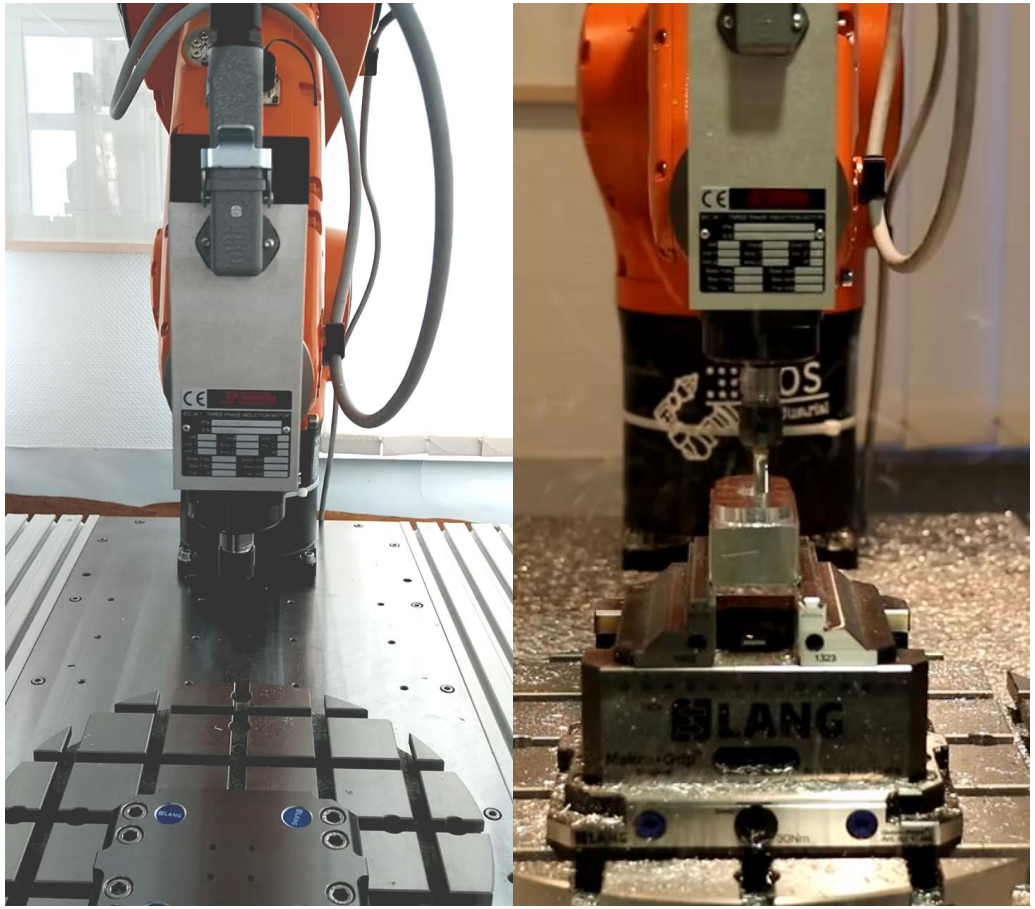
Asynchrone Kommunikationsstruktur



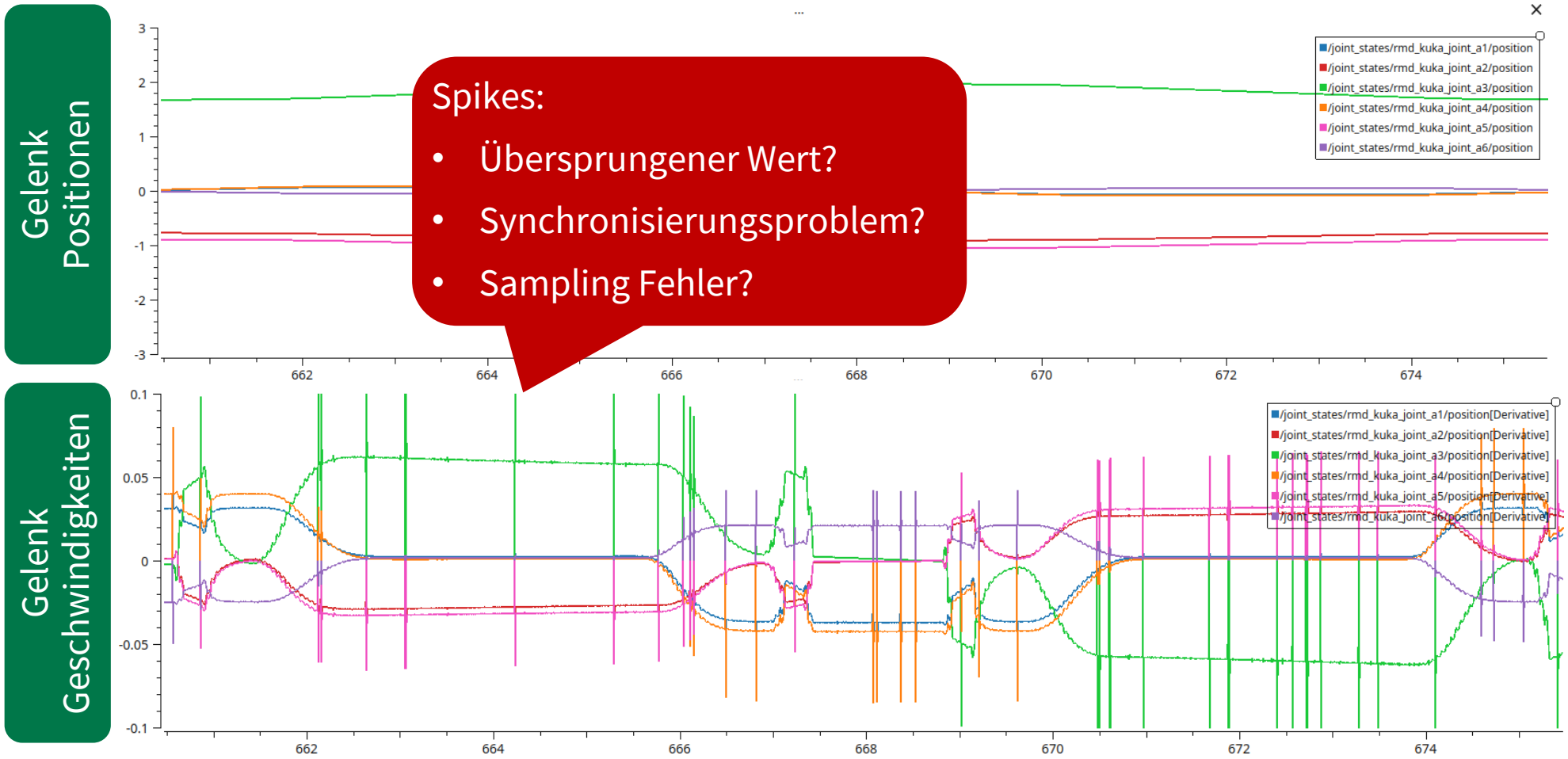
Fertig implementiert – Was nun?

- Treiber implementiert
  - `hardware_interface` steht
  - Robot Description gebaut
  - MoveIt aufgesetzt
  - …

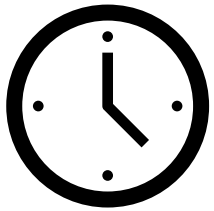
Problem: Roboter fängt an zu “zittern”
ca. 150s nach Beginn der Trajektorie



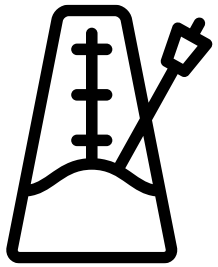
Kein Vortrag ohne plotjuggler



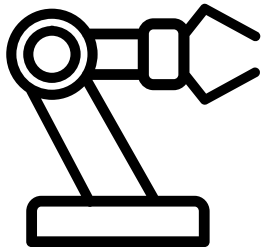
Wie debuggt man so etwas?



Lange Laufzeit: Bis zu 5 Minuten bis Probleme auftreten



Hohe Frequenz: 250Hz Regelfrequenz



Tests an Hardware: Potentielle Beschädigung der Getriebe

LTTng - ros2_tracing

- **ros2_tracing:** Instrumentation für ROS 2 Pakete
  - Tracing Instrumentation für Kernpakete
  - Integration in launch Action, CLI Kommando, …
- Default backend: **L**inux **T**race **T**oolkit: **n**ext **g**eneration
  - Per-CPU Buffering, effizientes binäres Trace Format, ...
- Workflow:
  1. Definiere Tracepoints
  2. Rufe Tracepoints in eigenem Code auf
  3. Session Daemon zeichnet Trace auf
  4. Analyse kann später durchgeführt werden

Tracing: Tracepoints definieren und nutzen

Kuka_rsi_driver/tracepoints.h

1

```
[…]
LTTNG_UST_TRACEPOINT_EVENT(
  kuka_rsi_driver,
  read,
  LTTNG_UST_TP_ARGS(
    unsigned long, time_nsec,
    unsigned long, period_nsec
  ),
  LTTNG_UST_TP_FIELDS(
    lttng_ust_field_integer(
      unsigned long, time_nsec, time_nsec
    ) lttng_ust_field_integer(
      unsigned long, period_nsec, period_nsec
    )
  )
)
[…]
```

Hardware_interface.cpp

2

```
[…]

Hardware_interface::return_type
KukaRsiHardwareInterface::read(
  const rclcpp::Time& time,
  const rclcpp::Duration& period
) {

  lttng_ust_tracepoint(
    kuka_rsi_driver,
    read,
    time.nanoseconds(),
    period.nanoseconds()
  );

[…]
```

Tracing: Trace aufzeichnen

3

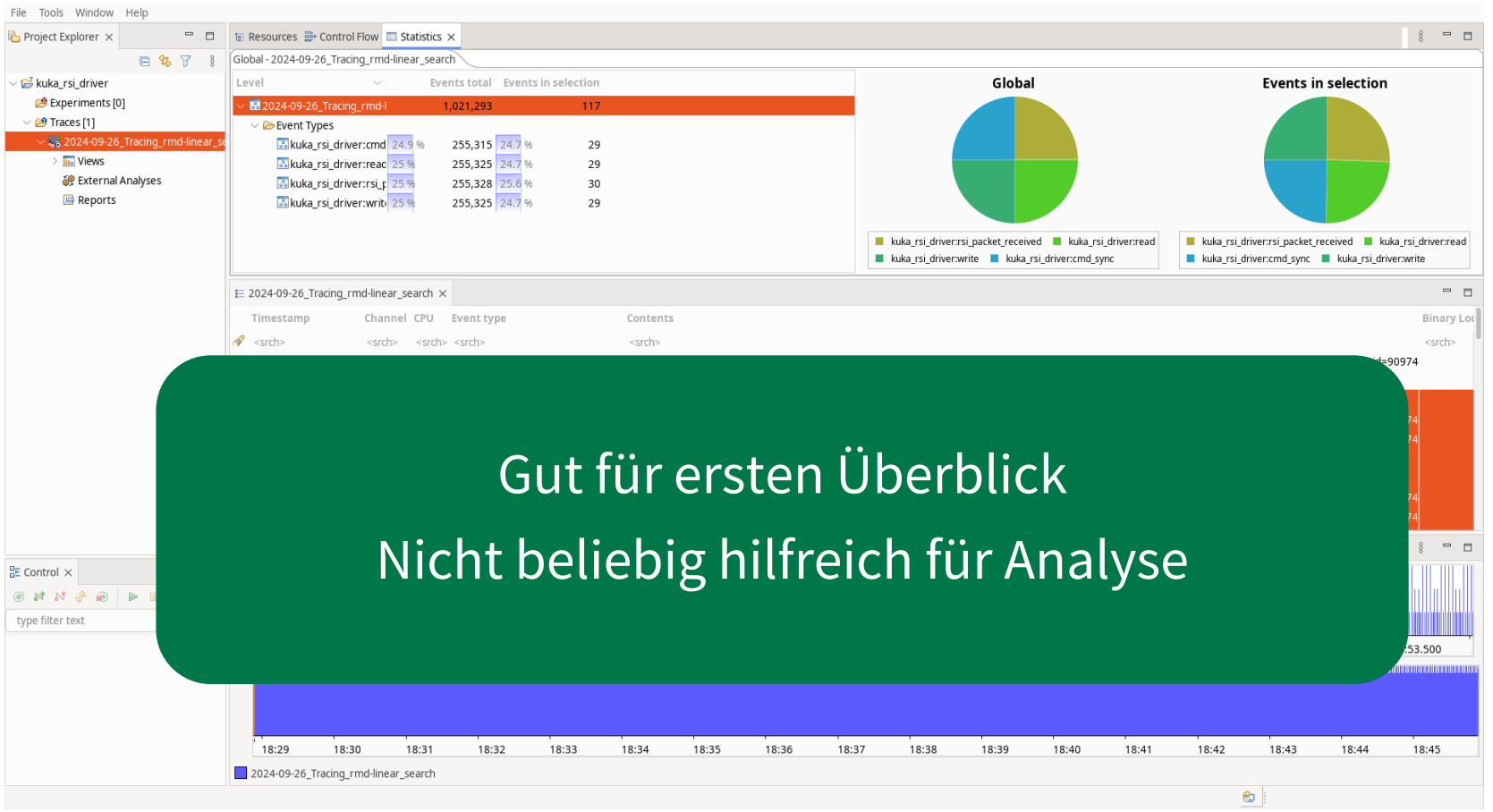
start_robot.launch.py

```
from tracetools_launch.action import Trace

[…]

launch_description.add_action(
  Trace(
    session_name="kuka_rsi_driver",
    events_ust=[
      "kuka_rsi_driver:read",
      "kuka_rsi_driver:write",
      "kuka_rsi_driver:rsi_packet_received",
      […]
    ]
  )
)

[…]
```

~/.ros/tracing/**kuka_rsi_driver**

Tracing: Analyse – Trace Compass

4



Gut für ersten Überblick

Nicht beliebig hilfreich für Analyse

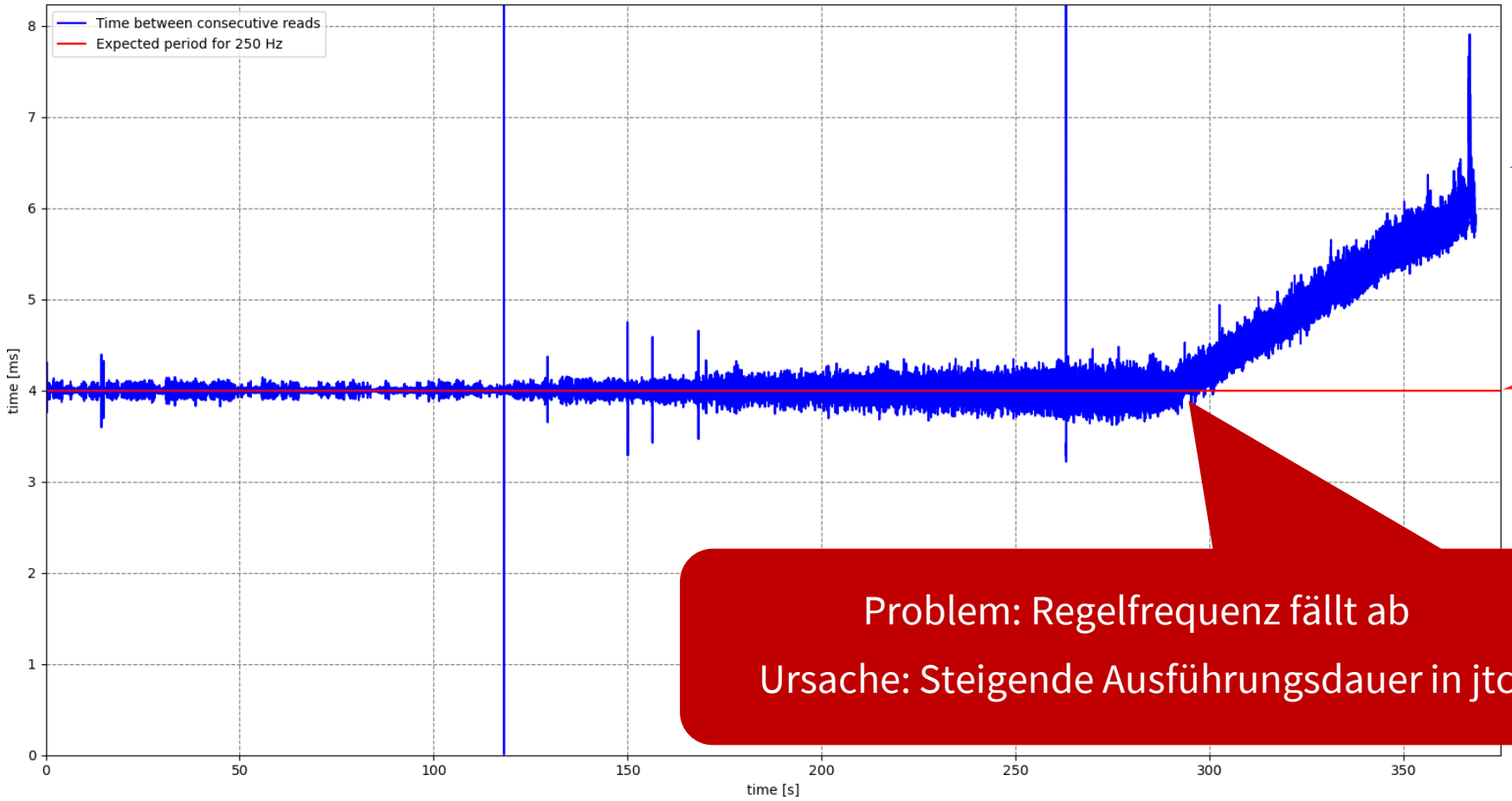
Problem 1 - Analyse

- Tiefergehende Analyse: **babeltrace** ➔ Erst aufzeichnen, später verschiedene Auswertungen durchführen

4

```
write_periods = []
for msg in bt2.TraceCollectionMessageIterator(path):
    if not isinstance(msg, bt2._EventMessageConst):
        continue
    t = msg.default_clock_snapshot.ns_from_origin

    if msg.event.name == "kuka_rsi_driver:write":
        write_periods.append((t, t - last_write_t))
        last_write_t = t
```

Problem 1 - Analyse



`Write()` - Periode

250Hz

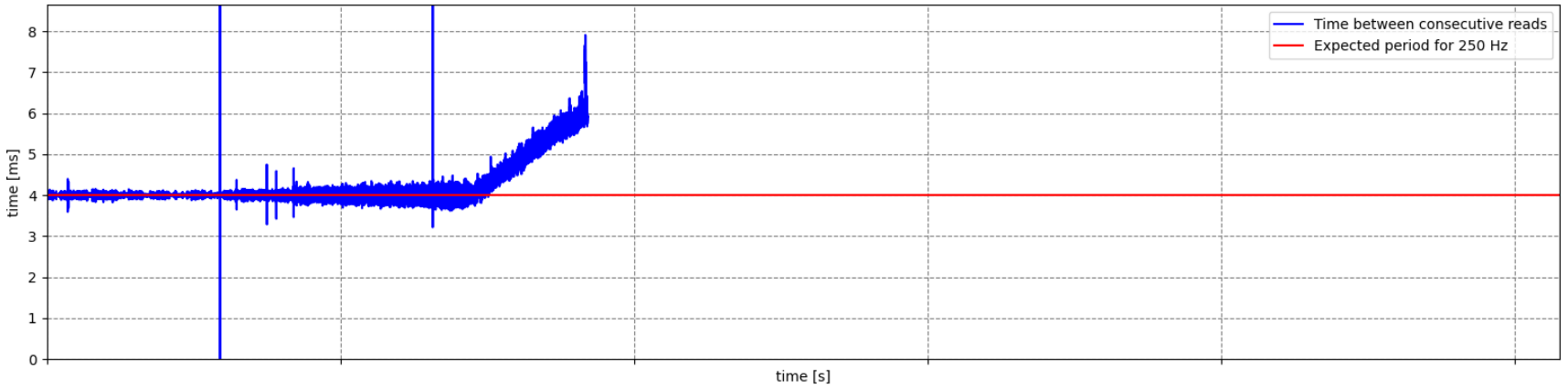
➔ 4ms

Problem: Regelfrequenz fällt ab

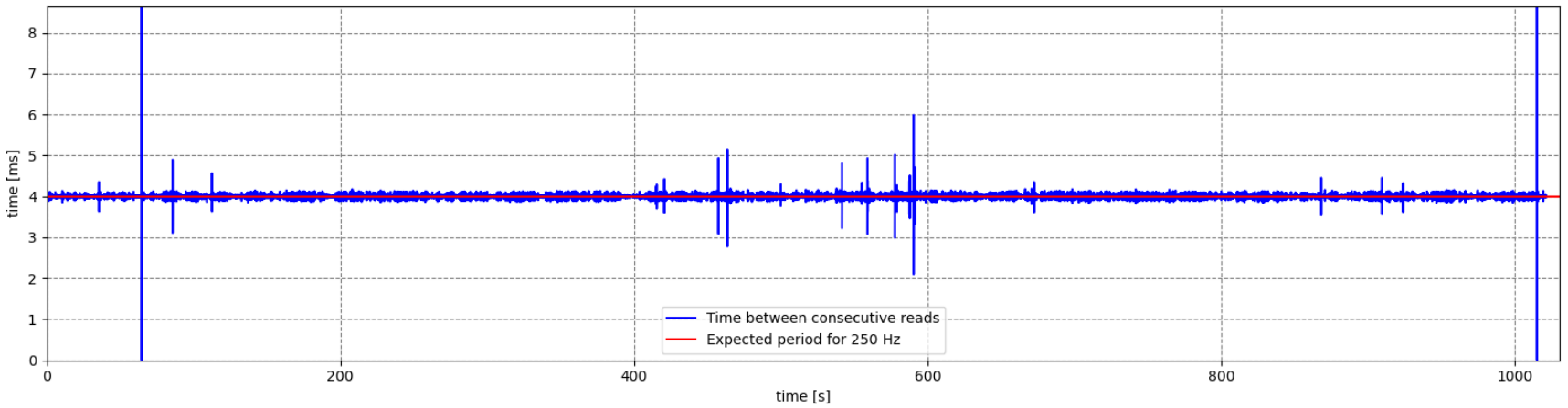
Ursache: Steigende Ausführungsdauer in jtc

Problem 1 - Lösung

Vorher



Nachher

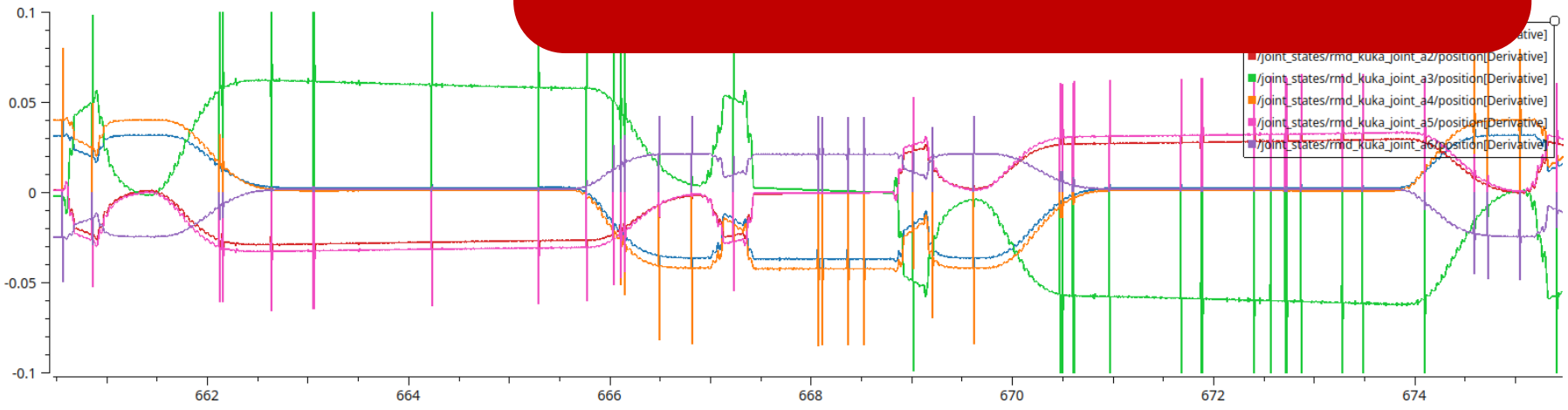


Alle Probleme beseitigt?

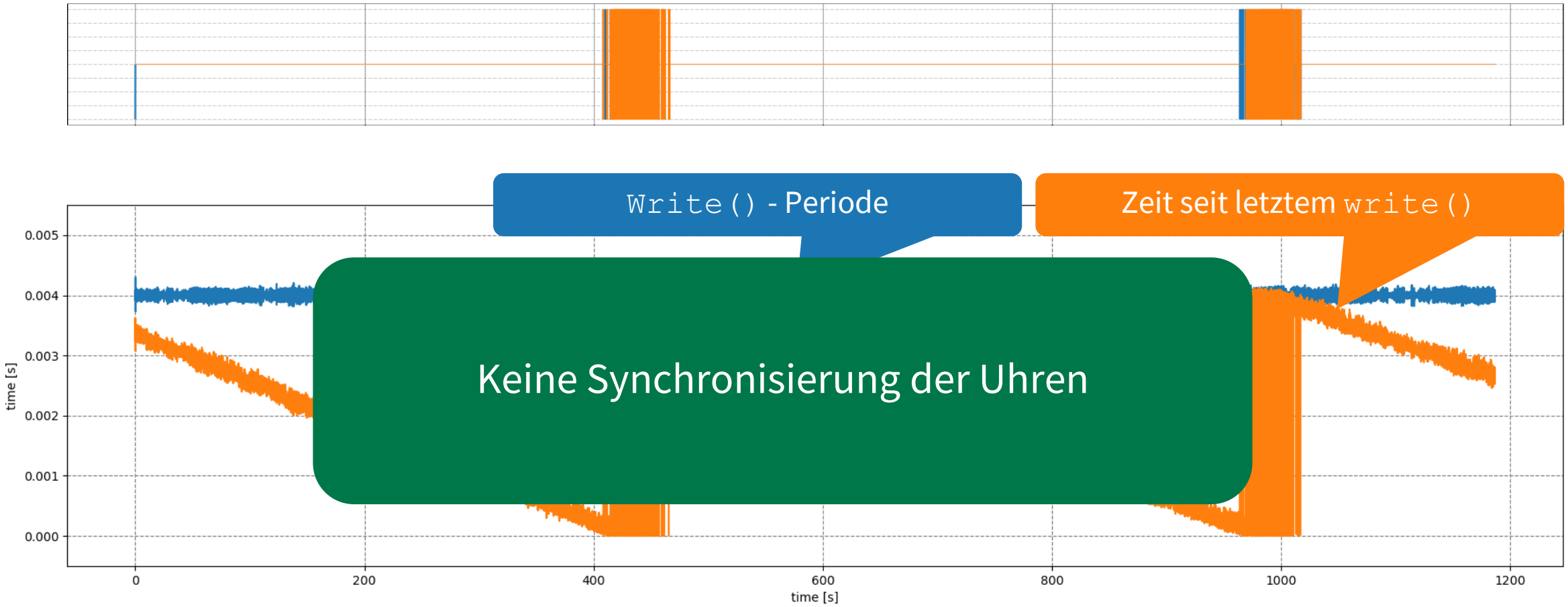
- Problem upstream repariert
- Neue Experimente

Problem noch nicht voll behoben

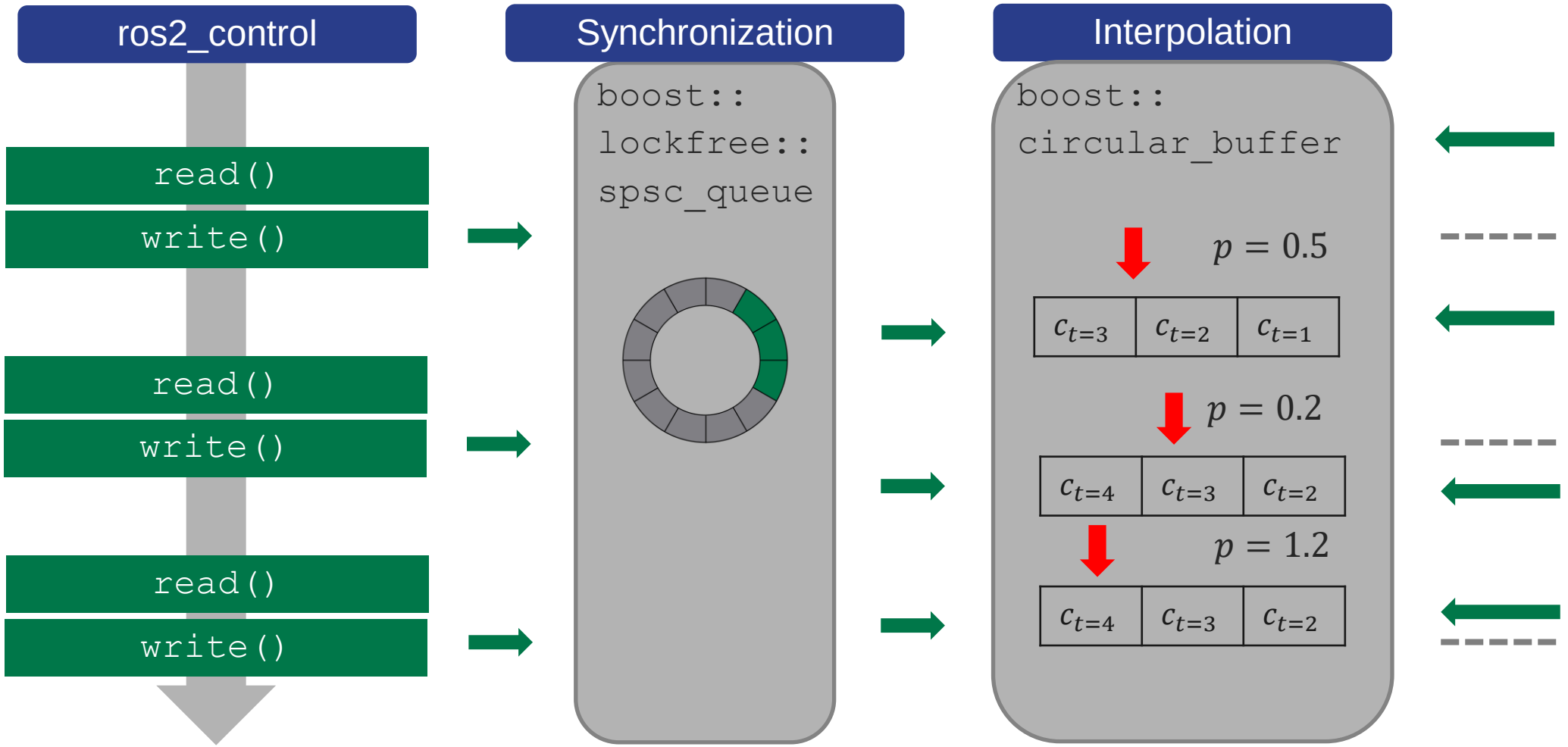
Alle ~500s für ~20s Vibrationen



Problem 2 - Analyse



Problem 2 - Lösung



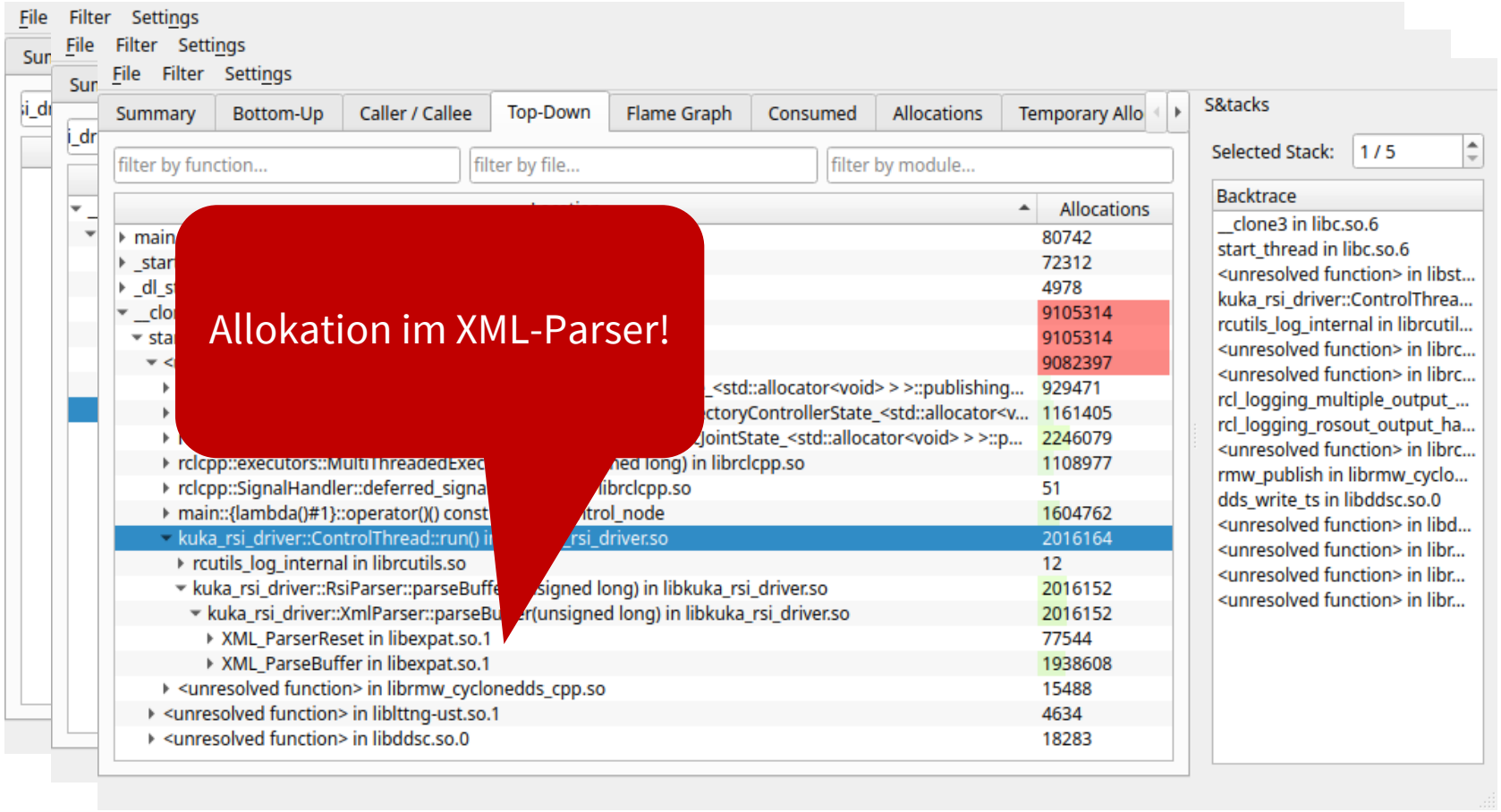
Heaptrack

- Realtime Komponenten müssen Jitter minimieren
  - Keine dynamischen Speicher Allokationen!
- Generelle Design Entscheidungen
  - Streaming-basierter XML-Parser **`libexpat`** statt DOM-Basierter **`tinyxml`**
  - Pre-Allokation aller Objekte mit dynamischer Größe
  - …
- Aber: Schwierig Ergebnis manuell zu beurteilen

start_robot.launch.py

```
[…]

launch_description.add_action(
  Node(
    package="controller_manager",
    executable="ros2_control_node",
    output="screen",
    parameters=[
      controllers_file,
    ],
    prefix=[“heaptrack”],
  )
)

[…]
```

Heaptrack



Offensichtlich nicht im normalen Kontrollzyklus
➔ Kein Problem

`read()`	
`write()`	183
`run()`	2016152

Allokation im Kontrollzyklus

kuka_rsi_driver

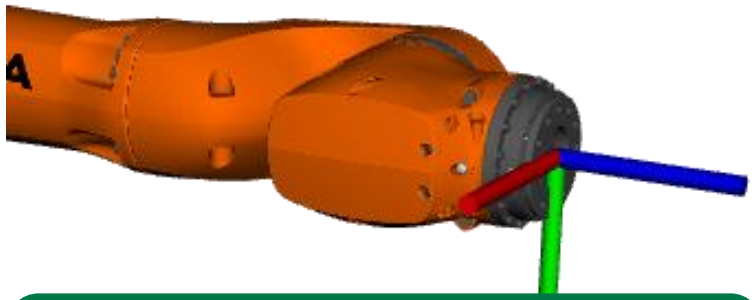
- Open Source (BSD 3-Clause)
- Einschränkung: Nur Treiber, kein URDF etc.

Speed-Scaling auf Teach Pendant (`$OV_PRO`)

`ros2_controllers #1191`

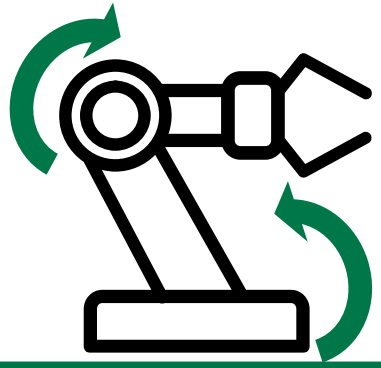
Asynchrone Kommunikation

RSI Simulator



Gemessene End-Effektor Pose

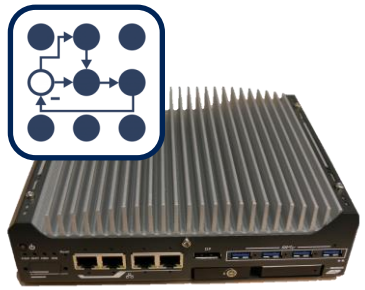
`pose_broadcaster/PoseBroadcaster`



Drehmomente in `joint_states`

Einbindung externer Komponenten

Anwendung

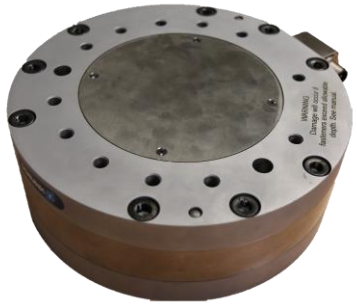


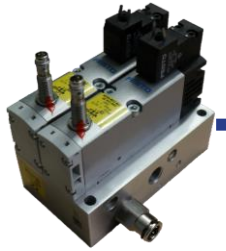
RSI

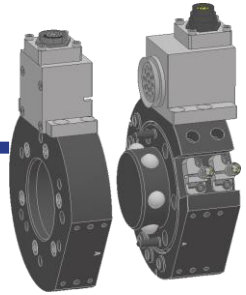
EtherCAT

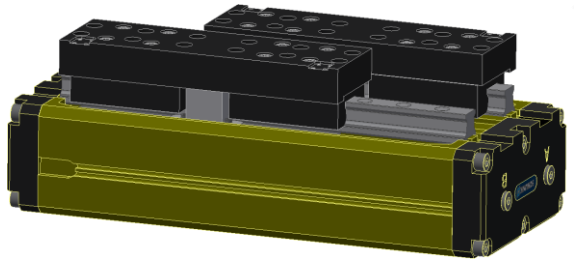
IO-Link

EtherCAT









Kontakt

FZI Forschungszentrum Informatik

Robert Wilbrandt

Haid-und-Neu-Str. 10-14
76131 Karlsruhe

+49 721 9654 - 230
wilbrandt@fzi.de

www.fzi.de

https://github.com/fzi-forschungszentrum-informatik/kuka_rsi_driver