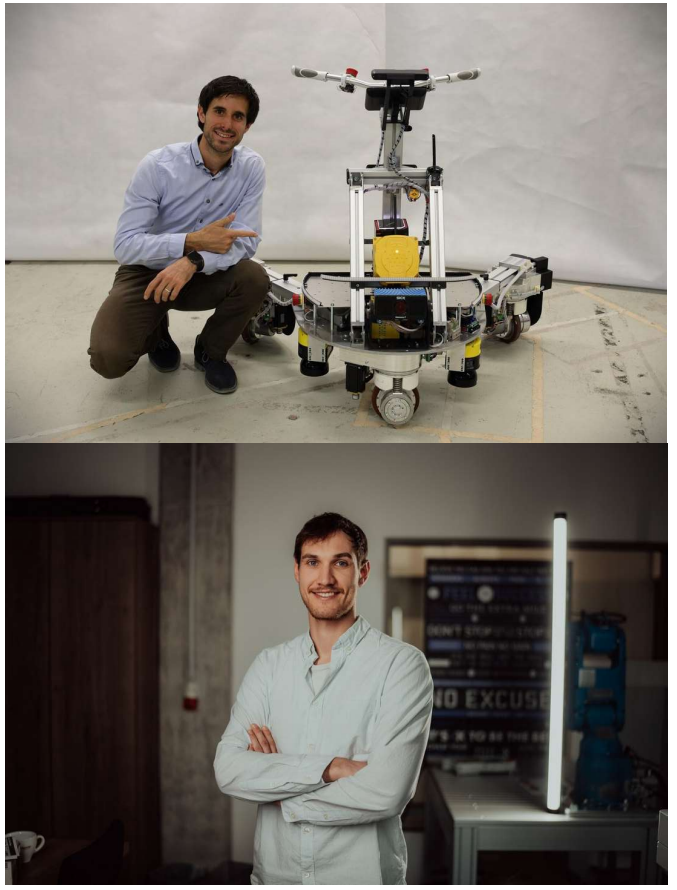
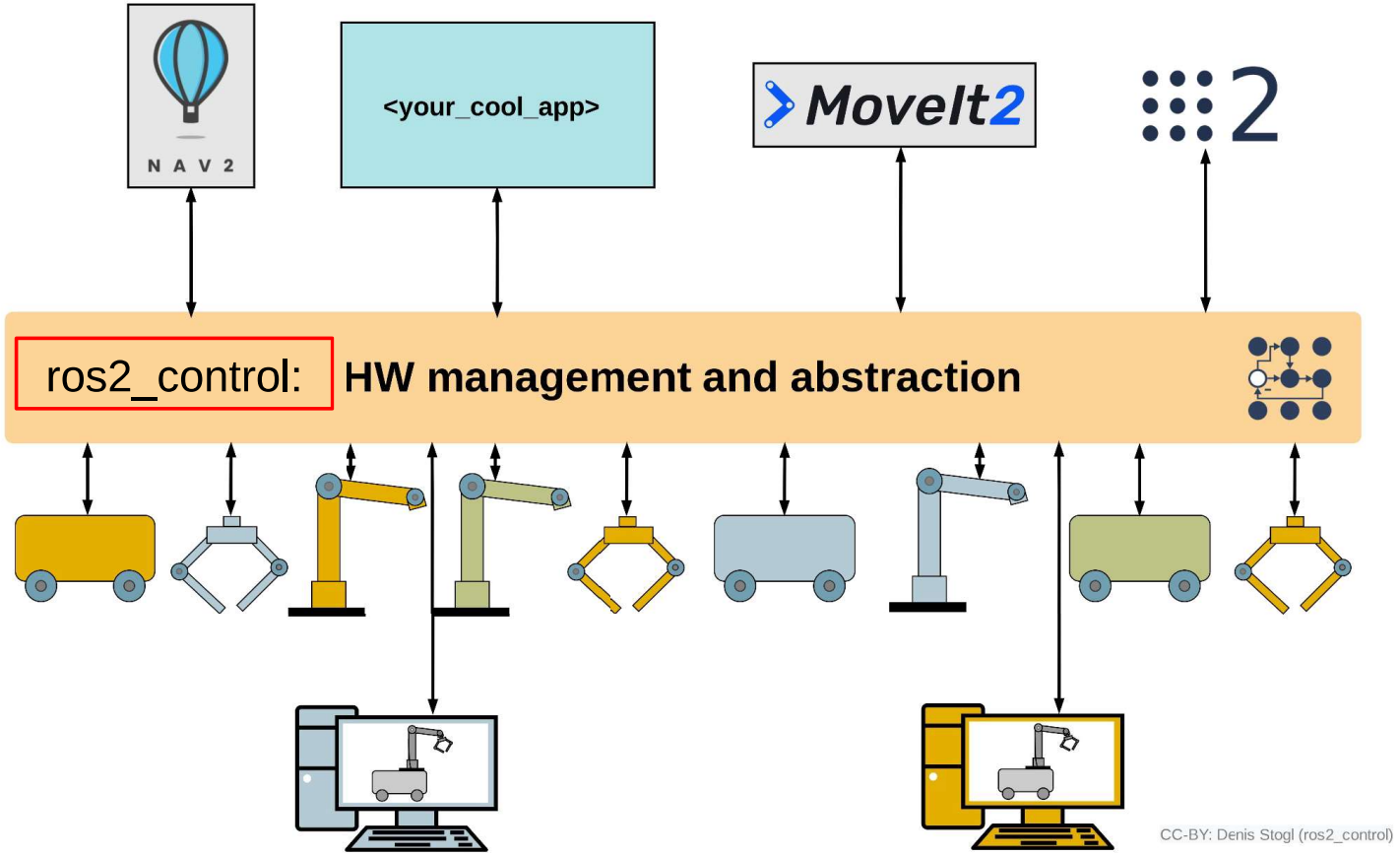
# Von Hardware-Anbindung Bis Workspace-Management: Produktivitätsbooster Für ROS2

# ros2_control @b-robotized.com

- Professionelle Integration und Entwicklung von ROS2 Robotikanwendungen
  - Echtzeit Hardware Regelung
  - Prototypen- und Produktentwicklung
  - Industrieautomatisierung mit ROS2
  - Consulting und Booster für Robotik-teams
- 4 Vollzeitetwickler
  - + Großes Netzerk aus 10+ Robotikexperten

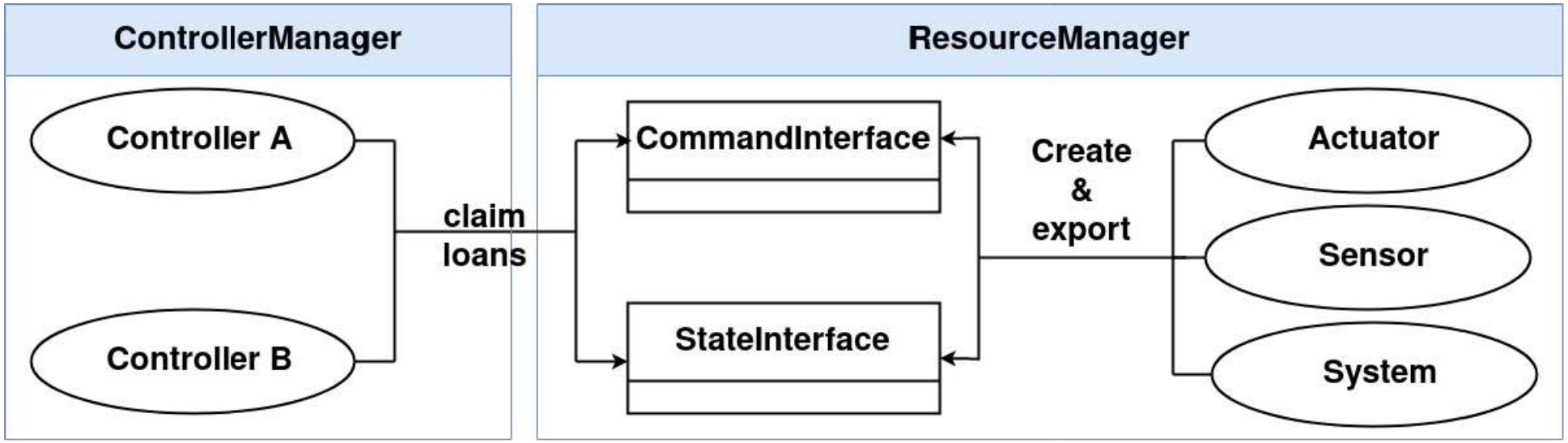


# ros2_control - kernel for ROS2



# ros2_control - kernel for ROS2

- ControllerManager → Management der Regler
- ResourceManager → Management der Hardwarekomponenten

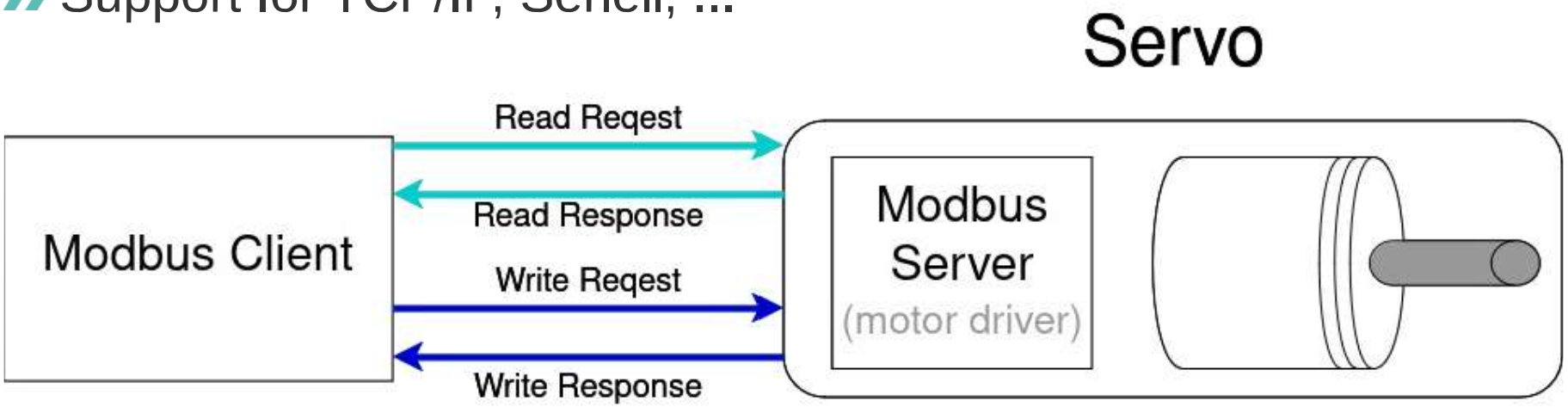


# Modbus Hardware Interface

# Modbus

- Kommunikations Protokoll
- Klient-Server (Request/Response)

- Support for TCP/IP, Seriell, ...



# Modbus

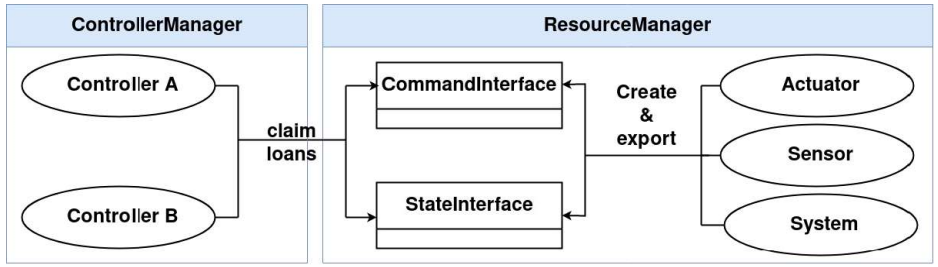
- Unterschiedliche Typen

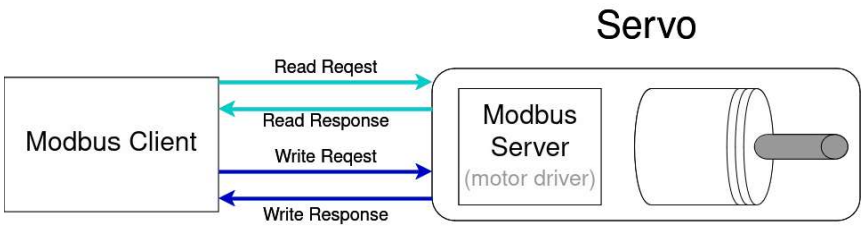
| | Read/Write access | |
|---|---|---|
| | R/W | R |
| Type | • Bits/Coils (read/write) | • Input Bits/Coils or Discrete Input (read only) |
| | • Register/Holding Register (read/ write) | • Input Register (read only) |

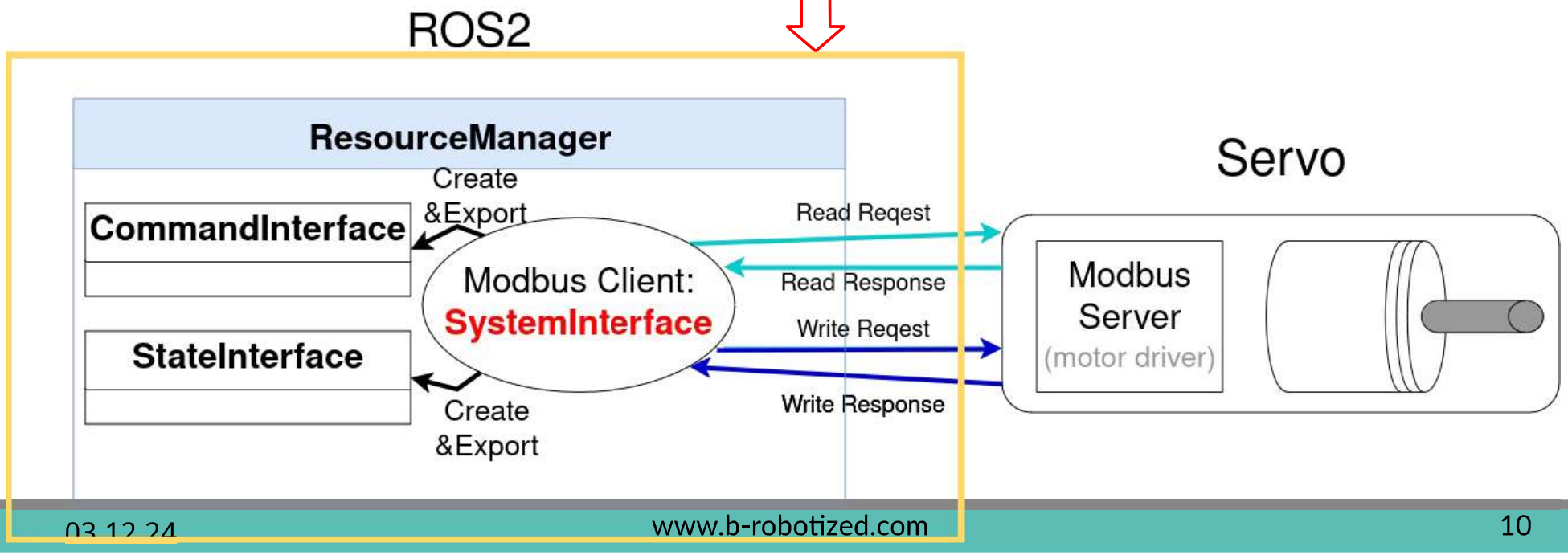
# Modbus

- Aufbauend auf: **libmodbus**
  - Open-Source Bibliothek für Kommunikaition mit modbusfähigen Geräten
  - Unterstütz TCP & RTU
  - Verfügbar auf Linux, FreeBSD, Mac und Windows
- Stellt uns Funktionalität für:
  - Management der Verbindung (Verbinden/Verbindung trennen)
  - Lesen und Schreiben der Bits & Register
- Register: uint_16t → n register: reads n*sizeof(uint_16t)
- Bits: uint_8t → n bits reads: n*sizeof(uint_8t)

# Konzept


ControllerManager
Controller A
Controller B
claim
loans
ResourceManager
CommandInterface
StateInterface
Create
&
export
Actuator
Sensor
System


Servo
Modbus Client
Read Reqest
Read Response
Write Reqest
Write Response
Modbus
Server
(motor driver)


ROS2
ResourceManager
Create
&Export
CommandInterface
StateInterface
Modbus Client:
SystemInterface
Create
&Export
Read Reqest
Read Response
Write Reqest
Write Response
Servo
Modbus
Server
(motor driver)

# Funktionalität

- Automatische Handhabung der Verbindung
  - Aufbau/Abbau
  - Überwachung ob die Verbindung abgebrochen ist
- Einfache Definition welche Register/Bits gelesen/geschrieben werden sollen
  - Erstellen der StateInterfaces
    - Lesen der Bits/Register
    - Konvertierung von uint_8t/uint_16t → double
  - Erstellen der CommandInterfaces
    - Schreiben der Bits/Register
    - Konvertierung von double → uint_8t/uint_16t

# Beispiel

```
<robot xmlns:xacro="http://wiki.ros.org/xacro">
  <xacro:macro name="my_servo_motor_ros2_control" params="
               name
               modbus_server_ip:=10.150.1.4
               modbus_server_port:=502
               modbus_use_persistent_connection:=true"
               >
    <ros2_control name="${name}" type="system">
      <hardware>
        <plugin>modbus_hardware_interface/ModbusHardwareInterface</plugin>
        <param name="modbus_server_ip">${modbus_server_ip}</param>
        <param name="modbus_server_port">${modbus_server_port}</param>
        <param name="use_persistent_connection">${modbus_use_persistent_connection}</param>
      </hardware>
    </ros2_control>
  </xacro:macro>
</robot>
```

# Beispiel

```
<robot xmlns:xacro="http://wiki.ros.org/xacro">
  <xacro:macro name="my_servo_motor_ros2_control" params="
                name
                modbus_server_ip:=10.150.1.4
                modbus_server_port:=502
                modbus_use_persistent_connection:=true"
                >
    <ros2_control name="${name}" type="system">
      <hardware>
        <plugin>modbus_hardware_interface/ModbusHardwareInterface</plugin>
        <param name="modbus_server_ip">${modbus_server_ip}</param>
        <param name="modbus_server_port">${modbus_server_port}</param>
        <param name="use_persistent_connection">${modbus_use_persistent_connection}</param>
      </hardware>
    </ros2_control>
  </xacro:macro>
</robot>
```

# Beispiel

| Reg | Name | Size | Access | Range | Default | Unit | Description | |
|---|---|---|---|---|---|---|---|---|
| 7 | RUN_ CURRENT | 32bit | R/W | 0-1533 | 511 | C: 5.87 mA<br>B: 3.91 mA<br>A: 1.96 mA | Current to use when the motor is running. The unit depends on the driver: C = 9 A, B = 6 A, A = 3 A. | Running Current |

```
  <xacro:macro name="my_servo_motor_ros2_control" params="
                name
                modbus_server_ip:=10.150.1.4
                modbus_server_port:=502
                modbus_use_persistent_connection:=true"
                >
    <ros2_control name="${name}" type="system">
      <hardware>
        <plugin>modbus_hardware_interface/ModbusHardwareInterface</plugin>
        <param name="modbus_server_ip">${modbus_server_ip}</param>
        <param name="modbus_server_port">${modbus_server_port}</param>
        <param name="use_persistent_connection">${modbus_use_persistent_connection}</param>
      </hardware>

      <joint name="${prefix}linear_joint">
        <state_interface name="running_current">
          <param name="register">7</param>
          <param name="bits_to_read">32</param>
          <param name="conversion_fn">to_int_to_float</param>
          <param name="read_function">register</param>
          <param name="factor">0.00587</param> # 9A / 1533
        </state_interface>
      </joint>
    </ros2_control>
  </xacro:macro>
</robot>
```

conversion_fn:
- konvertierung von uint_xt → double

read_functions:
- register
- input_register
- bits
- input_bits

# Beispiel

| Reg | Name | Size | Access | Range | Default | Unit | Description | |
|---|---|---|---|---|---|---|---|---|
| 3 | P_SOLL | 32bit | R/W | $(-2^{31})$-$(2^{31}-1)$ | 0 | Steps | The desired position. When in position mode, the motor will move to this position. This value can be changed at any time. | Position |

```
<robot xmlns:xacro="http://wiki.ros.org/xacro">
  <xacro:macro name="my_servo_motor_ros2_control" params="
                name
                modbus_server_ip:=10.150.1.4
                modbus_server_port:=502
                modbus_use_persistent_connection:=true"
                >
    <ros2_control name="${name}" type="system">
      <hardware>
        <plugin>modbus_hardware_interface/ModbusHardwareInterface</plugin>
        <param name="modbus_server_ip">${modbus_server_ip}</param>
        <param name="modbus_server_port">${modbus_server_port}</param>
        <param name="use_persistent_connection">${modbus_use_persistent_connection}</param>
      </hardware>

      <joint name="${prefix}linear_joint">
        <command_interface name="position">
          <param name="register">3</param>
          <param name="bits_to_read">32</param>
          <param name="conversion_fn">to_int_to_modbus</param>
          <param name="write_function">register</param>
          <param name="factor">${409600 / 0.005}</param> # tics per revolution / screw pitch in mm
        </command_interface>
      </joint>
    </ros2_control>
  </xacro:macro>
</robot>
```

conversion_fn:
- konvertierung von double → uint_xt

write_functions:
- register
- input_register
- bits
- input_bits

# Beispiel

```
<robot xmlns:xacro="http://wiki.ros.org/xacro">
  <xacro:macro name="my_servo_motor_ros2_control" params="
               name
               modbus_server_ip:=10.150.1.4
               modbus_server_port:=502
               modbus_use_persistent_connection:=true"
               >
    <ros2_control name="${name}" type="system">
      <hardware>
        <plugin>modbus_hardware_interface/ModbusHardwareInterface</plugin>
        <param name="modbus_server_ip">${modbus_server_ip}</param>
        <param name="modbus_server_port">${modbus_server_port}</param>
        <param name="use_persistent_connection">${modbus_use_persistent_connection}</param>
      </hardware>

      <joint name="${prefix}linear_joint">
        <command_interface name="position">
          <param name="register">3</param>
          <param name="bits_to_read">32</param>
          <param name="conversion_fn">to_int_to_modbus</param>
          <param name="write_function">register</param>
          <param name="factor">${409600 / 0.005}</param> # tics per revolution / screw pitch in mm
        </command_interface>
        <state_interface name="running_current">
          <param name="register">7</param>
          <param name="bits_to_read">32</param>
          <param name="conversion_fn">to_int_to_float</param>
          <param name="read_function">register</param>
          <param name="factor">0.00587</param> # 9A / 1533
        </state_interface>
      </joint>
    </ros2_control>
  </xacro:macro>
</robot>
```

# Wie Erweitern?

```
class MyServo : public modbus_hardware_interface::ModbusHardwareInterface

hardware_interface::CallbackReturn MyServo::on_init(
  const hardware_interface::HardwareInfo & info)
{
  if (modbus_hardware_interface::ModbusHardwareInterface::on_init(info) != CallbackReturn::SUCCESS)
  {
    return CallbackReturn::ERROR;
  }
  ... // some hardware specific handling
}

hardware_interface::return_type MyServo::read(
  const rclcpp::Time & time, const rclcpp::Duration & period)
{
  auto ret = modbus_hardware_interface::ModbusHardwareInterface::read(time, period);
  if (ret != hardware_interface::return_type::OK)
  {
    return ret;
  }
  ... // some hardware specific handling
}
```

# Wie Erweitern?

```
<robot xmlns:xacro="http://wiki.ros.org/xacro">
  <xacro:macro name="my_servo_motor_ros2_control" params="
....
                ">
    <ros2_control name="${name}" type="system">
      <hardware>
        <plugin>my_servo_package/MyServo</plugin>
.....
      </hardware>
....
    </ros2_control>
  </xacro:macro>
</robot>
```

# Open-Source

- ModbusHardwareInterface:
- https://github.com/StoglRobotics/modbus_hardware_interface
- ros2_control:
- https://github.com/ros-controls/ros2_control
- https://control.ros.org/rolling/index.html

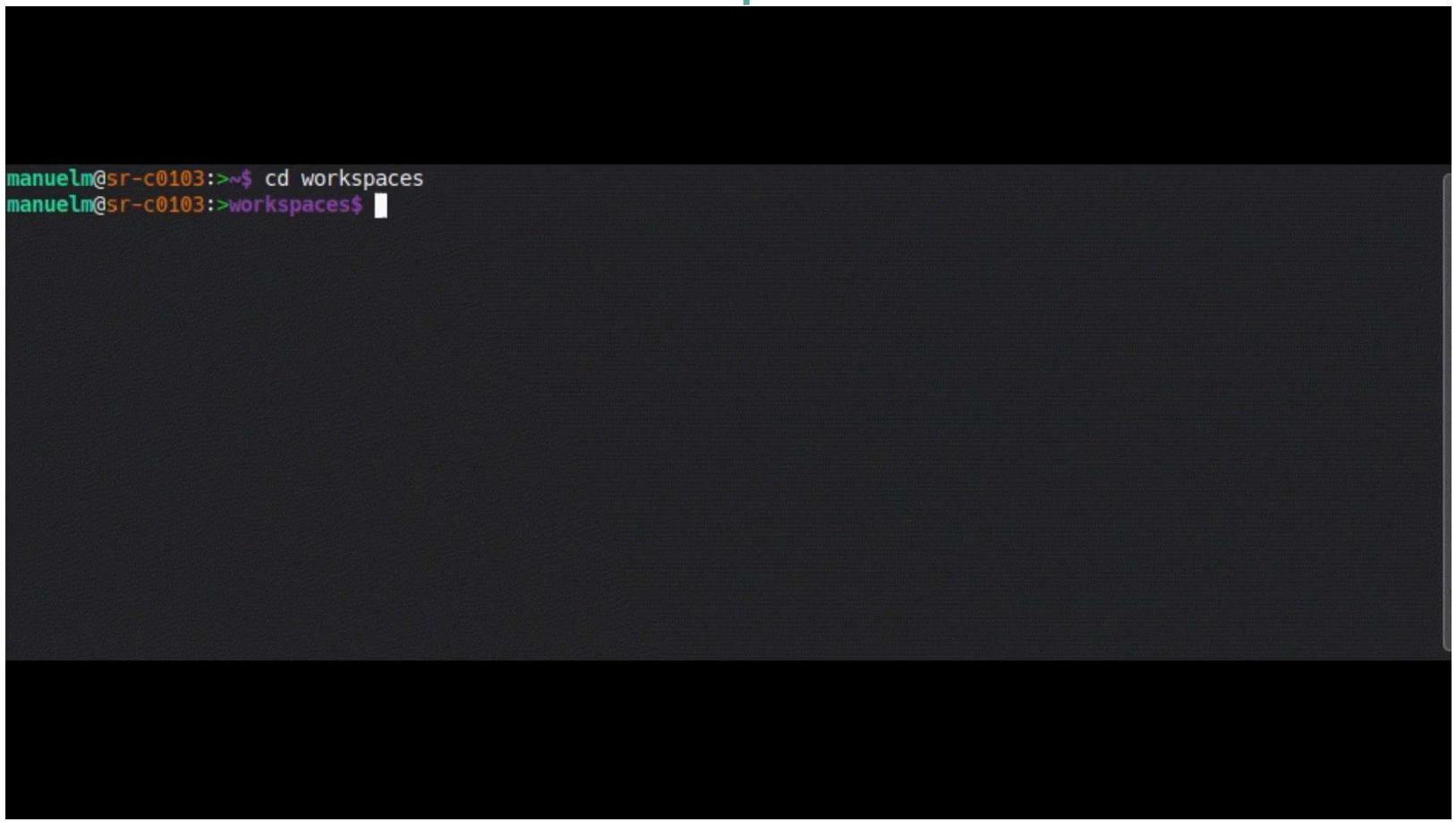
# RosTeamWS (rtw)

# RosTeamWS

- Mehr als nur ein open-source Workspace-Management-Tool

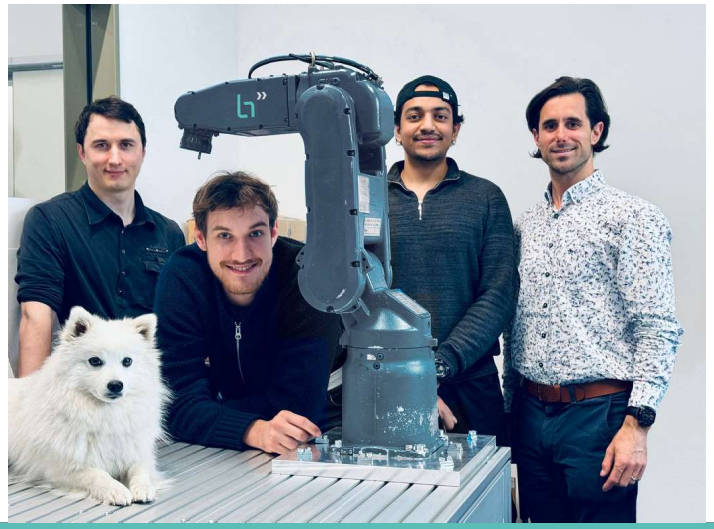
  https://github.com/StoglRobotics/ros_team_workspace
- Workspace-Management+
  - Support für ROS und ROS2 (Foxy, Humble, Jazzy, Rolling)
  - Erstellen und löschen von Workspaces
  - Option für erstellen der Workspaces im Docker
  - Support für Repository + Branch + .repos file
  - Automatisches Sourcen und viele nützliche Kommandos
- Erstellen von Packeten und Templates
  - Regler und Hardware
  - Robot Description (urdf)
  - Robot Bringup (launch)

# RosTeamWS – create workspace



# RosTeamWS – use workspace

b»robotized
BY STOGL ROBOTICS



Reach out to us! https://www.b-robotized.com/