



# Real-Time Motion Control in ROS: Uniting Machinekit HAL with Tormach's ZA6 Robot

John Morris

Software Engineering

Brandon Duarte

Software Engineering



# TORMACH®

Headquartered in Madison WI

100% employee-owned

## Our Customers

Hobbyist

Small businesses

Researchers







# ZA6 Robot

6kg industrial arm

Natural fit with machine tool products

Powered by ROS and fully open source





“Each new robotic platform we integrate with, we have to evaluate RT expectations and requirements. We would much rather have that solved out of the box and use that engineering time building more sophisticated robots!”

Nathan Brooks, CTO



# Machinekit HAL cuts machine control development time





# Machinekit HAL: Real Time Threading Environment

## Real-time thread programming

### Linux RT thread “flavor” support

- [RT\\_PREEMPT](#)
- and/or [POSIX](#) (non-RT)
- or [Xenomai](#)

## Real Time Scheduling

- scheduling algorithm e.g. POSIX ``SCHED_FIFO``
- elevated scheduling priority

## More real-time programming

### RT <-> non-RT communication

- lock-free ring buffers
- shm
- mutexes

## Arch-dependent programming

- Atomic 64-bit access
- hi-res timers

## Other

- ``mlockall()``: VM page faults

## Fix real-time jitter sources

### OS and hardware tuning

- Intel CPU C-States (power saving)
- CPU frequency scaling (Intel; AMD)

### Dedicated RT CPU

- ``isolcpus``
- `cgroup `cpuset``
- CPU affinity





# Machinekit HAL: Modular components for common tasks

## Hardware drivers

- EtherCAT
- “hostmot” FPGA f/w
- Beaglebone PRUSS
- Modbus
- GPIO

## Motor control

- PID
- PWM generator
- Step/dir generator
- Quadrature

## Logic

- and/or/not/xor
- flipflop
- mux/demux
- oneshot
- lut
- timedelay

## Instrumentation

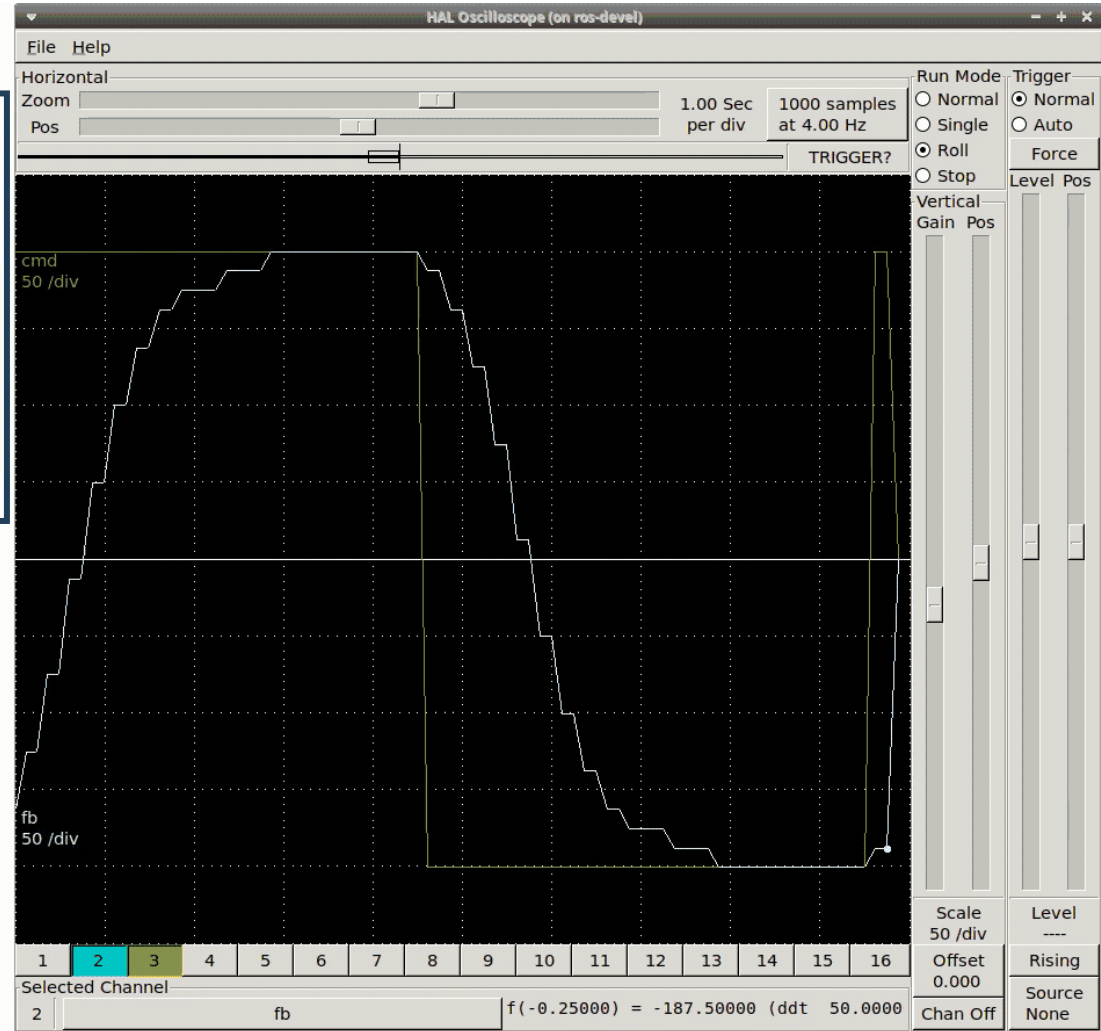
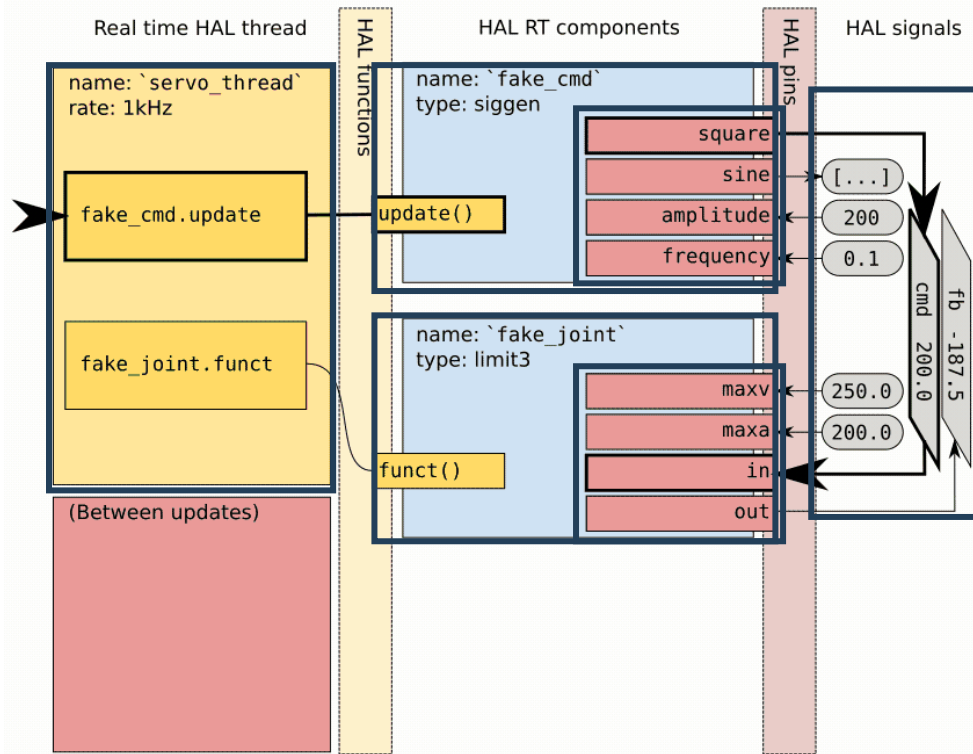
- halscope
- sampler
- streamer
- latencybins
- histobins

## Arithmetic & math

- ddt/integrator
- biquad/lowpass/kalman filters
- limit1/limit2/limit3
- offset/scale
- sum/mult
- siggen



# Machinekit HAL: Basic concepts





# Machinekit HAL: Basic configuration

1. load "siggen" comp

2. load "limit3" comp

3. net "cmd" & "fb" signals

4. set up real time thread

5. start the scope!

```
1 # Fake "command": signal generator component square wave
2 loadrt siggen names=fake_cmd
3 setp fake_cmd.amplitude 200
4 setp fake_cmd.frequency 0.1
5
6 # Fake "joint": limit3 component
7 #   output is accel + velocity limited input
8 newinst limit3 fake_joint
9 setp fake_joint.maxv 250.0
10 setp fake_joint.maxa 200.0
11
12 # Connect "cmd" and "fb" signals
13 net cmd fake_cmd.square => fake_joint.in
14 net fb <= fake_joint.out
15
16 # Define 4Hz RT thread, "pokey_thread", add functions and start
17 loadrt threads name1=pokey_thread period1=250000000
18 addf fake_cmd.update pokey_thread
19 addf fake_joint.funct pokey_thread
20 start
21
22 # Start halscope (wait for it to exit, then exit HAL)
23 loadusr -w halscope
```

Save to `demo.hal` and run:  
`halrun -f demo.hal`



## Machinekit HAL: Easy C and Python APIs

```
1 component and2 "Two-input AND gate";
2 pin in bit in0;
3 pin in bit in1;
4 pin out bit out
5     "TRUE if both in0 & in1 are TRUE, else FALSE";
6 function _nofp;
7 license "GPL";
8 ;;
9
10 FUNCTION(_)
11 {
12     out = in0 && in1;
13     return 0;
14 }
```

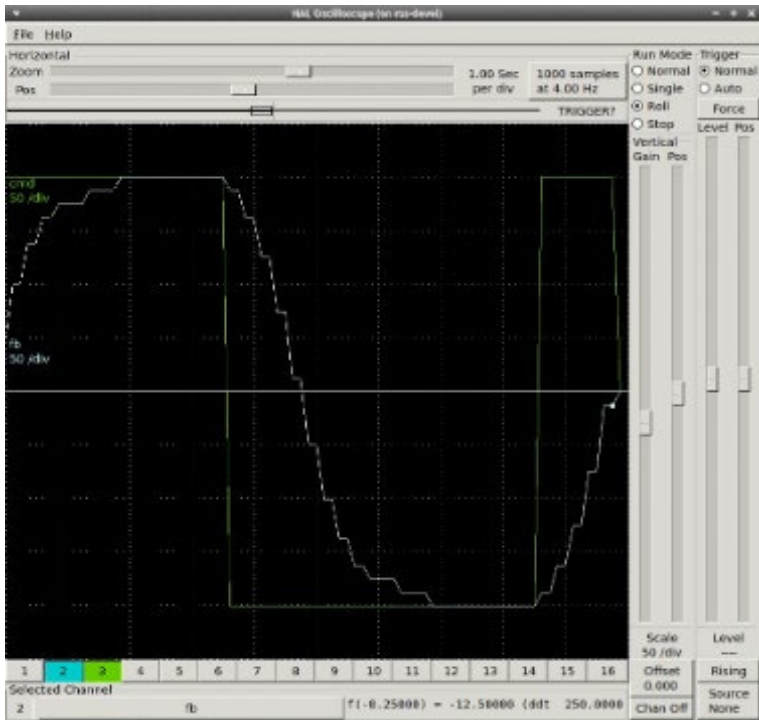
```
1 #!/usr/bin/python
2 import hal, time
3
4 # Set up HAL component and mark it "ready"
5 h = hal.component("and2_py")
6 h.newpin("in0", hal.HAL_FLOAT, hal.HAL_IN)
7 h.newpin("in1", hal.HAL_FLOAT, hal.HAL_IN)
8 h.newpin("out", hal.HAL_FLOAT, hal.HAL_OUT)
9 h.ready()
10
11 try:
12     while 1:
13         # Main update loop
14         time.sleep(1)
15         h["out"] = h["in0"] and h["in1"]
16
17 except KeyboardInterrupt:
18     raise SystemExit
```



# Machinekit HAL: Tools for inspecting running systems

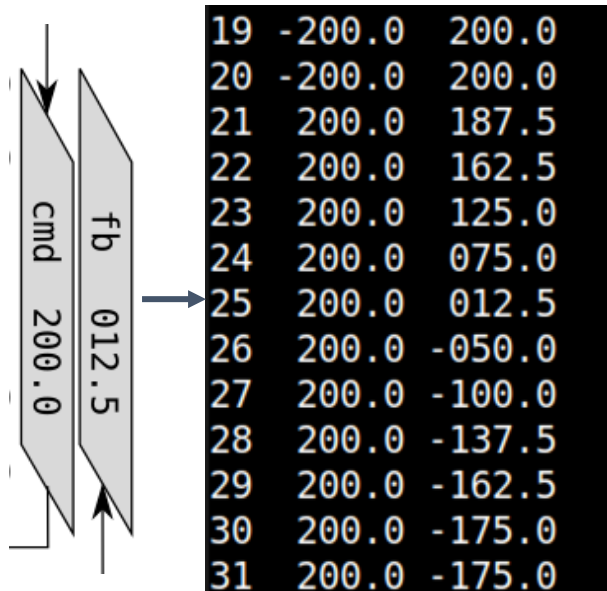
## halscope

visualize live data



## halsampler

data logging



## halcmd

update live system

```
$ halcmd show pin fake_cmd.frequency
Component Pins:
  Comp Inst Type Dir      Value Name
    78   float IN         0.1  fake_cmd.frequency

$ halcmd show pin fake_cmd.amplitude
Component Pins:
  Comp Inst Type Dir      Value Name
    78   float IN         200  fake_cmd.amplitude

$ halcmd show sig cmd
Signals:
Type      Value flags Name  linked to:
float     200   --  cmd    <== fake_cmd.square
                                     ==> fake_joint.in

$ halcmd show pin fake_joint.maxv
Component Pins:
  Comp Inst Type Dir      Value Name
    93   95 float IN         250  fake_joint.maxv

$ halcmd show pin fake_joint.maxa
Component Pins:
  Comp Inst Type Dir      Value Name
    93   95 float IN         200  fake_joint.maxa

$ halcmd show sig fb
Signals:
Type      Value flags Name  linked to:
float     -200  --  fb    <== fake_joint.out
```



`hal\_ros\_control` cuts robot control development time



**TORMACH®**



# ros2\_control + HAL: Controller Manager in HAL RT thread

```
- hal_config:  
  param:  
    - name: hal_debug_output  
      value: $(var hal_debug_output)  
    - name: hal_debug_level  
      value: $(var hal_debug_level)  
  actions:  
    - hal_rt_node:  
      pkg: hal_hw_interface  
      component: hal_control_node  
      param:  
        - name: robot_description  
          value: $(var robot_description_content)  
        - from: $(var hal_hw_interface_yaml_path)  
        - from: $(var ros2_controllers_yaml_path)  
      wait_timeout: $(var hal_wait_timeout)  
    - hal_files:  
      hal_files:  
        - $(var hal_file)  
      param:  
        - name: sim_mode  
          value: $(arg sim_mode)  
        - from: $(arg hal_config_yaml_path)  
        - from: $(arg joint_limits_yaml_path)
```

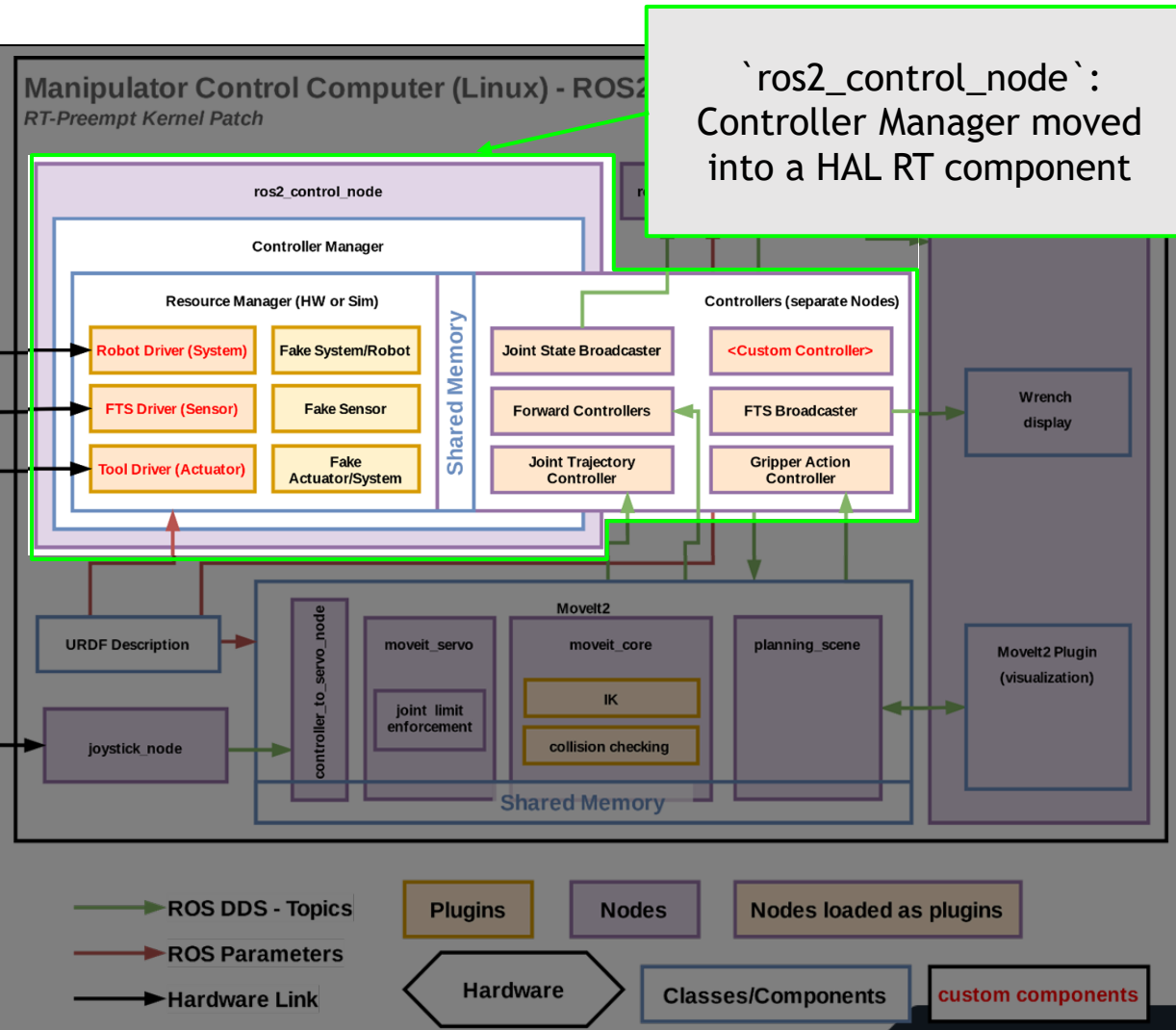
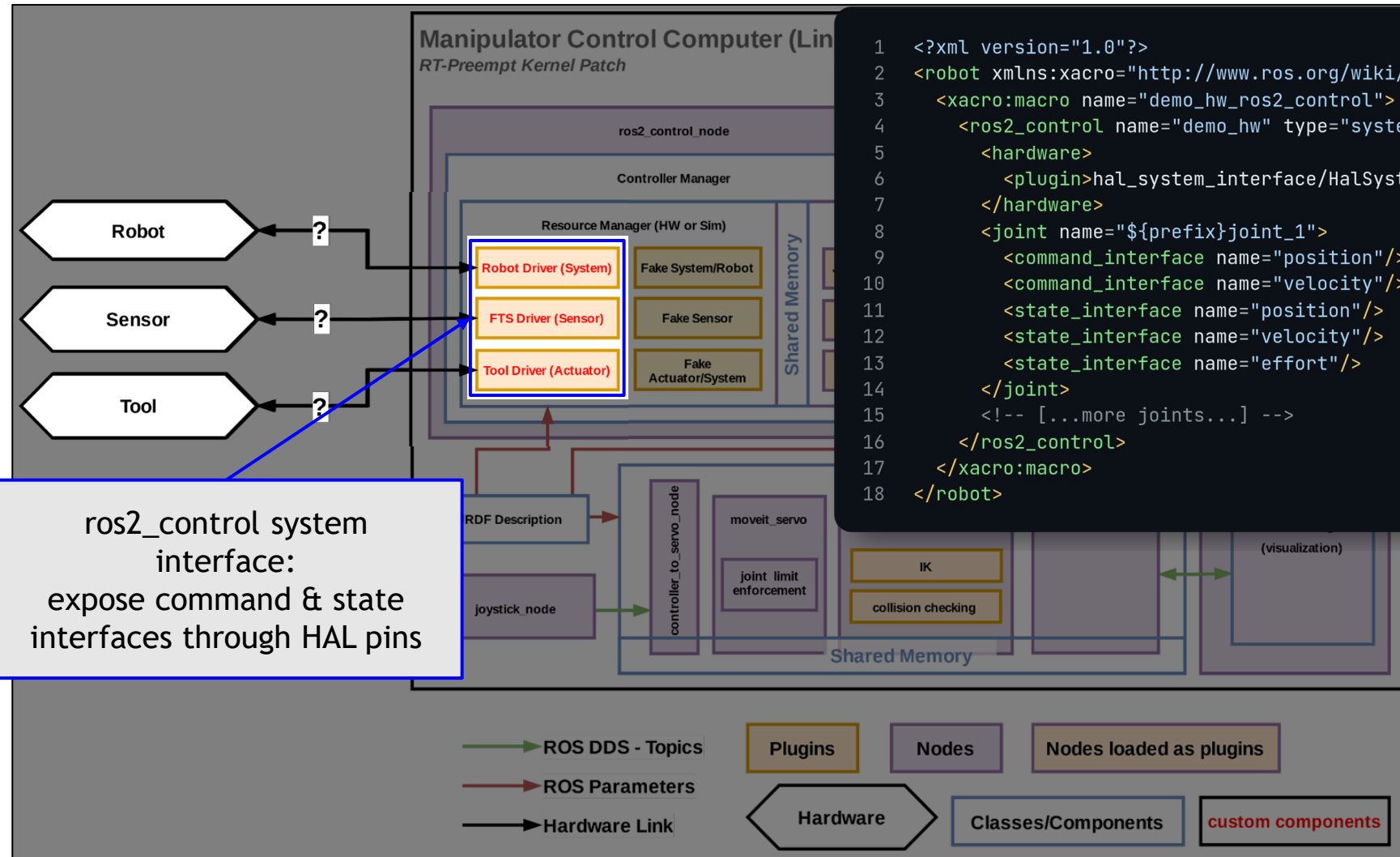


Diagram from [ros2\\_control docs](#)



# ros2\_control + HAL: Hardware interface



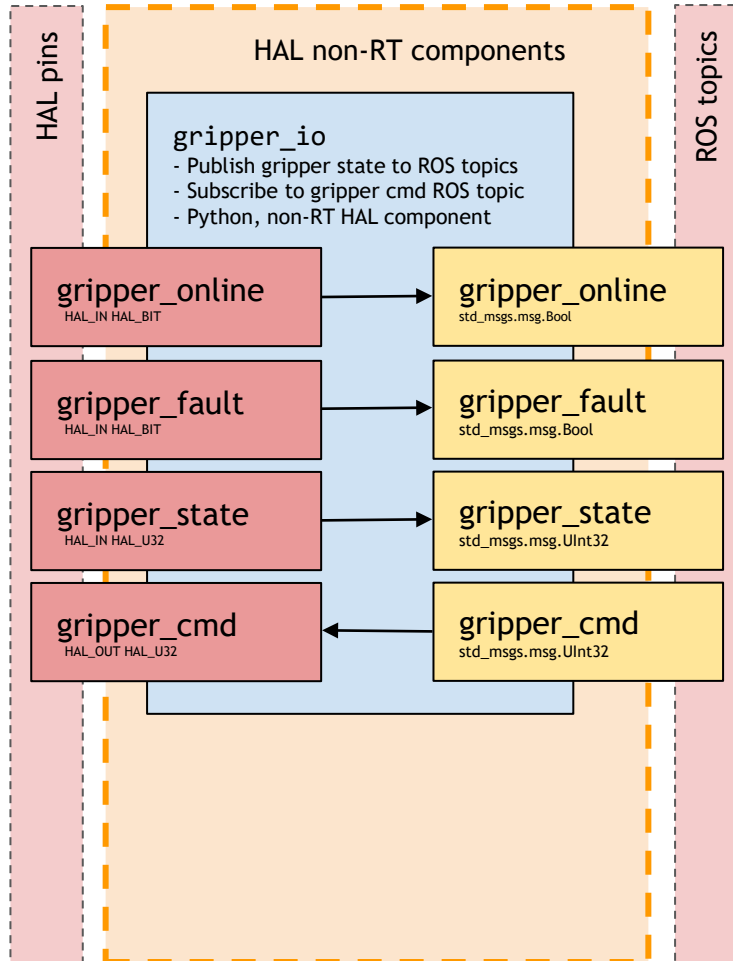
```
1 <?xml version="1.0"?>
2 <robot xmlns:xacro="http://www.ros.org/wiki/xacro">
3   <xacro:macro name="demo_hw_ros2_control">
4     <ros2_control name="demo_hw" type="system">
5       <hardware>
6         <plugin>hal_system_interface/HalSystemInterface</plugin>
7       </hardware>
8       <joint name="${prefix}joint_1">
9         <command_interface name="position"/>
10        <command_interface name="velocity"/>
11        <state_interface name="position"/>
12        <state_interface name="velocity"/>
13        <state_interface name="effort"/>
14      </joint>
15      <!-- [...more joints...] -->
16    </ros2_control>
17  </xacro:macro>
18 </robot>
```

Diagram from [ros2\\_control docs](#)





# `hal\_ros\_control` and ROS: Other interfaces



```
from hal_hw_interface.ros_hal_component import RosHalComponent
from hal_hw_interface.ros_hal_pin import RosHalPinSubscriber, RosHalPinPublisher

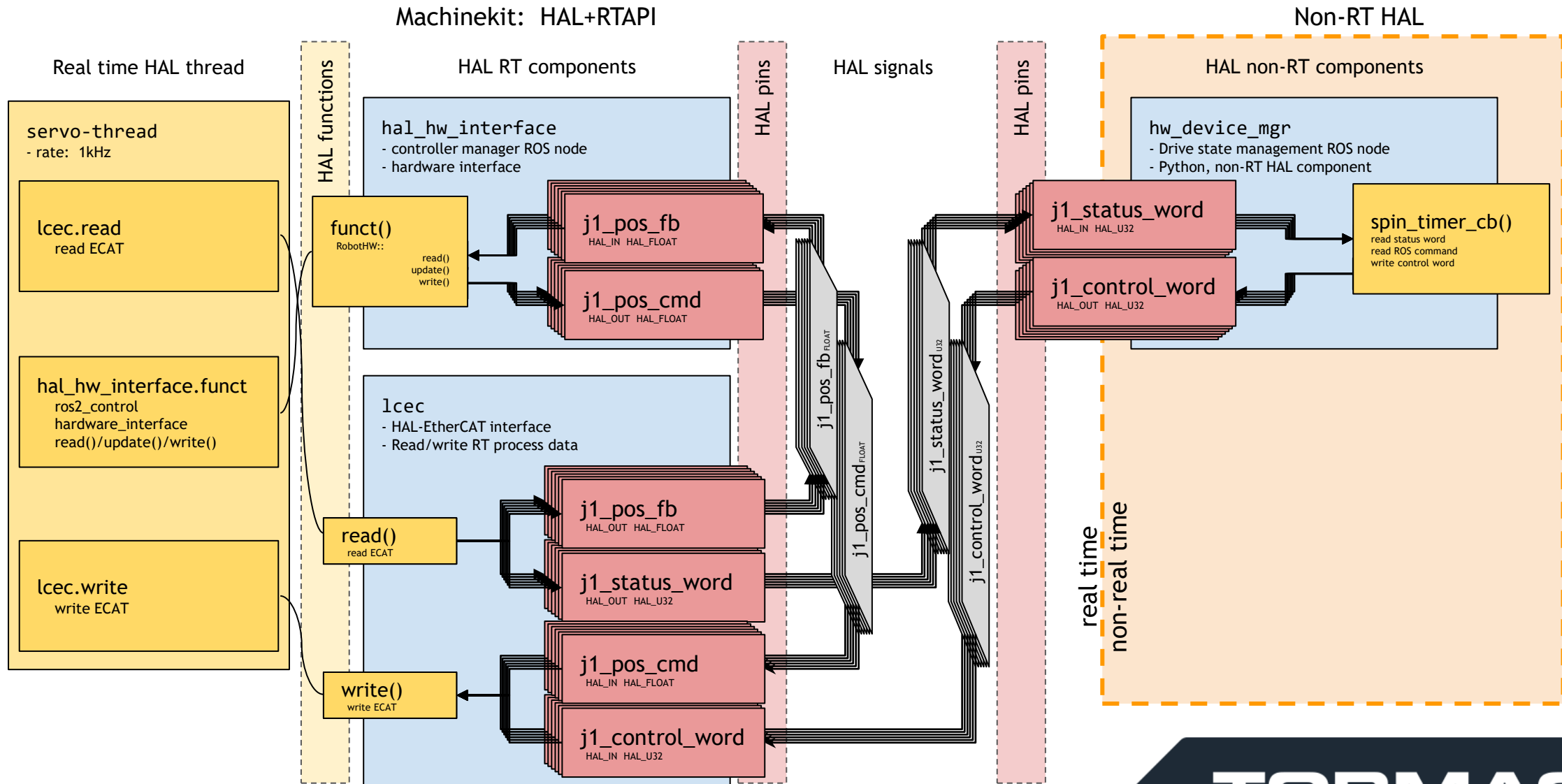
class GripperIO(RosHalComponent):
    compname = "gripper_io"

    def setup_component(self):
        self.pins = [
            RosHalPinPublisher("gripper_online", hal_type="BIT"),
            RosHalPinPublisher("gripper_fault", hal_type="BIT"),
            RosHalPinPublisher("gripper_state", hal_type="U32"),
            RosHalPinPublisher("gripper_cmd", hal_type="U32"),
        ]

    def update(self):
        for p in self.pins:
            p.update()
```



# Basic `hal\_ros\_control` robot hardware configuration





# ZA6 Robot: ROS-based, fully open source control stack

The screenshot displays the PathPilot control interface for the ZA6 robot. It is divided into several sections:

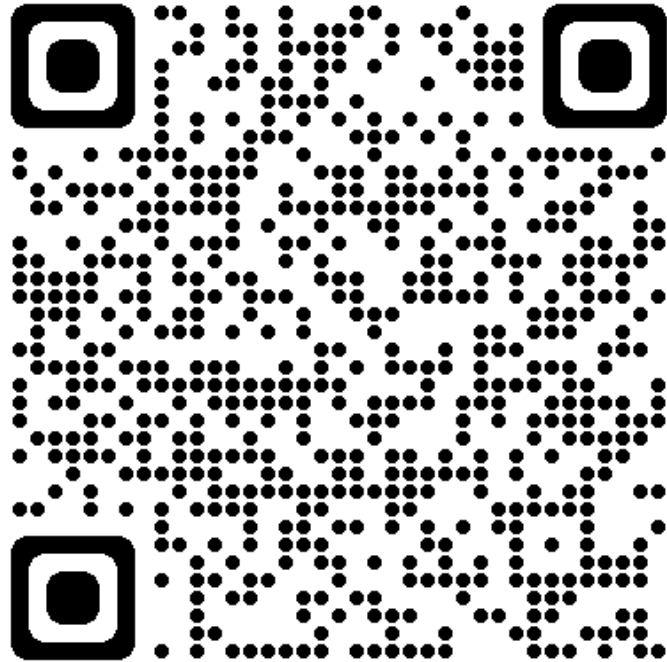
- Code Editor (top left):** Shows a Python script named `j6_exercise.py` with the following content:

```
1 from robot_command.rpl import *
2 set_units("mm", "deg", "s")
3 waypoint_1 = [[2.591, 57.071, -47.425,
4 waypoint_2 = [[2.591, 57.071, -47.425,
5
6 def main():
7     movej(waypoint_1)
8     movej(waypoint_2)
9
```
- 3D Model (top center):** A 3D rendering of the robot arm with a red tool tip and green/green tool axes.
- Control Panel (bottom left):** Includes buttons for **CYCLE START**, **SINGLE BLOCK**, **FEEDHOLD**, **OPTIONAL PAUSE**, **STOP**, and **RESET**. It also features a feed rate slider set to 100% and a max velocity slider set to 21%.
- Coordinate Display (bottom center):** A table showing the current position of the robot's joints in millimeters and degrees.

Axis	Position	Joint	Position
X	923.029	J1	2.591
Y	35.796	J2	57.071
Z	596.774	J3	-47.425
A	-123.206	J4	-7.255
B	-3.430	J5	23.710
C	-88.643	J6	94.453
- Status (bottom center):** Shows the robot is **Running** with units set to **mm / deg / second**.
- HAL Oscilloscope (on pathpilot-controller) (right):** A multi-channel oscilloscope showing:
  - joint6\_pos\_cmd** (purple): A slowly increasing step function.
  - joint6\_vel\_cmd** (green): A constant step function.
  - joint6\_torque\_fb** (red): A noisy signal fluctuating around a mean value.
  - joint6\_pos\_err** (cyan): A signal showing small oscillations around zero.



# HAL-ROS Control



[https://github.com/tormach/hal\\_ros\\_control](https://github.com/tormach/hal_ros_control)