Safety Certified ROS-native Industrial Manipulator

https://wiki.ros.org/pilz_robots

7th ROS-Industrial Conference
December 2019

Dr.-Ing. Manuel Schön
Product Management
Service Robotic Modules
Manipulator Module PRBT 6

- **Number of axes:** 6
  - Max. load capacity: 6 kg
  - Repetition accuracy: +/- 0.1 mm
  - Mounting direction: any
  - Weight: 19 kg
  - Max. operating range: 741 mm

- **Power supply:** 24 V DC
  - Interface: CANopen
  - Safety functions:
    - STO (safe torque off)
    - SBC (safe brake control)

---

No proprietary controller needed
Why using ROS?

Because it is the answer to Service Robotic in industrial and non-industrial environment!
Previous work in ROS

Driver
- based on ros_canopen
- safety functions

Industrial planners
- using Movelt!
- industrial requirements
- deterministic behavior
- basic movements: Linear, Point-to-Point, Circular
- blending of the above

Python API
- easy to use interface to aforementioned planners
- no extensive training required

Example: Moving a Robot with Python API

```python
r = Robot()

# Simple ptp movement in joint space
r.move(Ptp(goal=[0, 0.5, 0.5, 0, 0, 0],
         vel_scale=0.4))
start_joint_values = r.get_current_joint_states()

# Relative ptp movement
r.move(Ptp(goal=[0.1, 0, 0, 0, 0, 0],
           relative=True,
           vel_scale=0.2))

# Simple cartesian Lin movement
r.move(Lin(goal=Pose(position=Point(0.2, 0, 0.8)),
           vel_scale=0.1,
           acc_scale=0.1))

# Circ movement
r.move(Circ(goal=Pose(position=Point(0.2, -0.2, 0.8)),
            center=Point(0.1, -0.1, 0.8),
            acc_scale=0.4))

# Move robot with stored pose
r.move(Ptp(goal=pose_after_relative,
           vel_scale=0.2))
```

Supported by ROSIN - ROS-Industrial Quality-Assured Robot Software Components. More information: rosin-project.eu

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement no. 732287.
Application vs. Safety

ISO 10218-2
Goals for a safe Robot-Application in ROS

Establish ROS in Industrial Applications

- **Robot can be used with ROS**
  - no Proprietary Controller
  - no Proprietary Teach-Pendant

- As much functionality as possible implemented in ROS
  - Safety Controller for Safety Functionality

- **Robot is certifiable under EN ISO 10218-1**
  - Applications are build purely in ROS

- **Integrator can focus on application**
  - Safety is provided “with the Robot”
How do we do this?

Traditional Setup – the industrial way

Intended Setup – the Future

EN ISO 10218-1 Approved

to be EN ISO 10218-1 Approved
Exemplary Aspects of the Standard:

**Operational modes**
- Automatic / Manual Reduced Speed
  - display of Mode
  - monitoring of reduced speed

**Robot stopping functions**
- Emergency stop
  - smoothly stopping
  - brakes in emergency
  - brake Test
  - within time limit
  - triggered from external device

Full Standard ➔ [https://www.iso.org/standard/51330.html](https://www.iso.org/standard/51330.html)
Technical Overview

Dr.-Ing. Manuel Schön

Safety Certified ROS-native Industrial Manipulator
Development Progress

Start of Project: 04/2019
Implementation Functional Modules: 06/2019
Certification: 06/2020

Dr. - Ing. Manuel Schön

Safety Certified ROS-native Industrial Manipulator
Brake Test

- Robot must be able to brake safely
  - when emergency stop is pressed
  - when speed limit is violated
  - when other safety sensors are trigger

- Equipped with brakes
  - regular testing is required
  - Safety Controller ensures test is preformed

- ROS can
  - ask when test is required
  - execute test at any point before time limit

- Safety Controller
  - disables drives if test is not performed within limit
  - ensures Safety

Example: Performing a brake test with Python API

```python
r = Robot()

if r.is_brake_test_required():
    # Move robot to the pose where the brake test should be executed
    r.move(Ptp(goal=BRAKE_TEST_POSE))
    try:
        # Execute brake test
        r.execute_brake_test()
    except RobotBrakeTestException as e:
        # Handle error
        rospy.logerr(e)
```
Operation Modes

- **Automatic**
  - automatic execution of predefined program
  - e.g. Script written in our API

- **Manual reduced speed**
  - limit of speed to 250 mm/s
  - for teaching
  - ROS will monitor any TF frame
  - Robot can be controlled by any method in ROS

- **Manual high speed**
  - limit start at 250 mm/s but can be increased
  - control from ROS
  - for testing
Example Application: Visual Inspection

Task: Inspect part features for large number of product variants

Approach: Robot on-board camera supported on database to lookup poses and save results

Strengths of ROS:
- high-level control based on the adaption of State-Machine packages
- interface with other software components
- use of workspace based (OMPL) and deterministic (pilz_industrial_motion) motion planners

Current Setup

Inspection Poses

Demo Setup

Machine setup
We want to help establish ROS in industrial applications

Our robot RPBT6 supports ROS natively

We provide the safety, so you can focus on the application
Let me answer your questions!
Dr.- Ing. Manuel Schön

Pilz GmbH & Co. KG
Felix-Wankel-Straße 2
73760 Ostfildern, Germany
Tel.: +49 711 3409-7860

Please visit
https://github.com/pilzde/pilz_robots