Safety Certified ROS-native Industrial Manipulator

PILZ
THE SPIRIT OF SAFETY

https://wiki.ros.org/pilz_robots

ROSCon 2019
Macau, November 1st 2019

Christian Henkel
Advanced Development

Outline

PILZ
THE SPIRIT OF SAFETY

- Company Profile
- Goals
- Development Progress
- System Overview
- Component Aspects
- Example Application

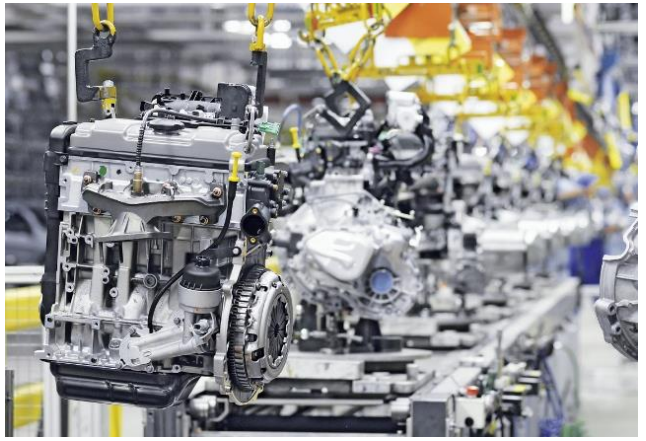
Company Profile

Pilz GmbH & Co. KG

- Independent automation technology company founded in 1948
- Headquarters in Ostfildern near Stuttgart
- Employees:
  - 2 515 worldwide
- 2018 turnover:
  - 345 Million Euro
  - 73 % export

Packaging technology

Automotive



Railway engineering

Wind energy

Manipulator Module PRBT

- **Number of axes: 6**
- Max. load capacity: 6 kg
- Repetition accuracy: +/- 0.15 mm
- Mounting direction: any
- Weight: 19 kg
- Max. operating range: 741 mm

- **Power supply: 24 V DC**
- Interface: CANopen, ROS
- Safety functions:
  - STO (safe torque off)
  - SBC (safe brake control)
- No proprietary controller needed

Previous work in ROS

PILZ
THE SPIRIT OF SAFETY

Driver

- Based on ros_canopen
- Safety functions

Industrial planners

- Using moveit
- 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

```
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))
```

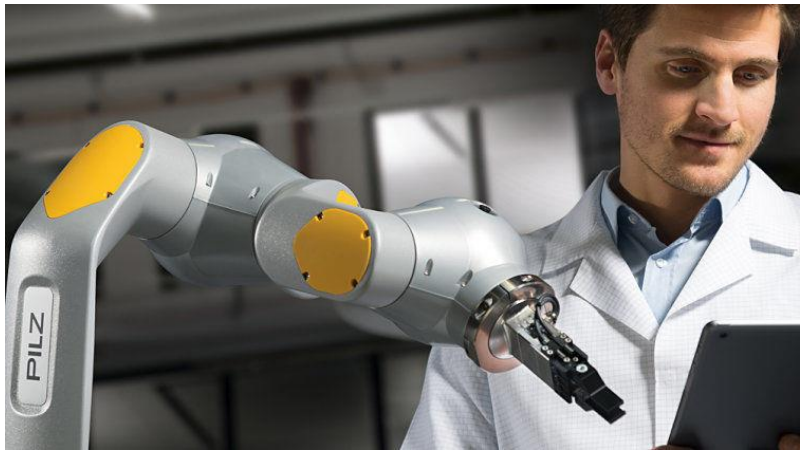
ROSin

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.

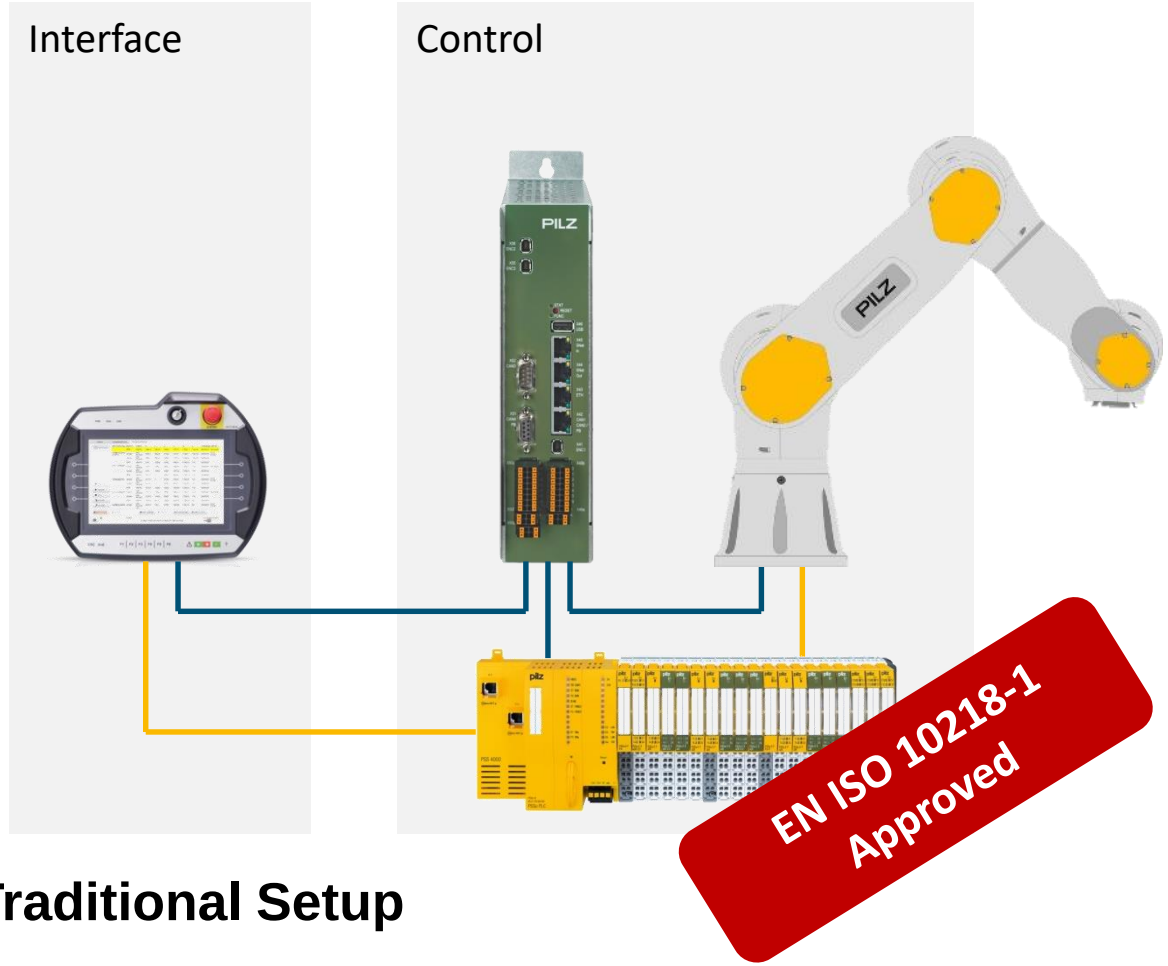
More on our planner at MoveIt Workshop
tomorrow @ Sheraton Grand Macao Hotel

Goals



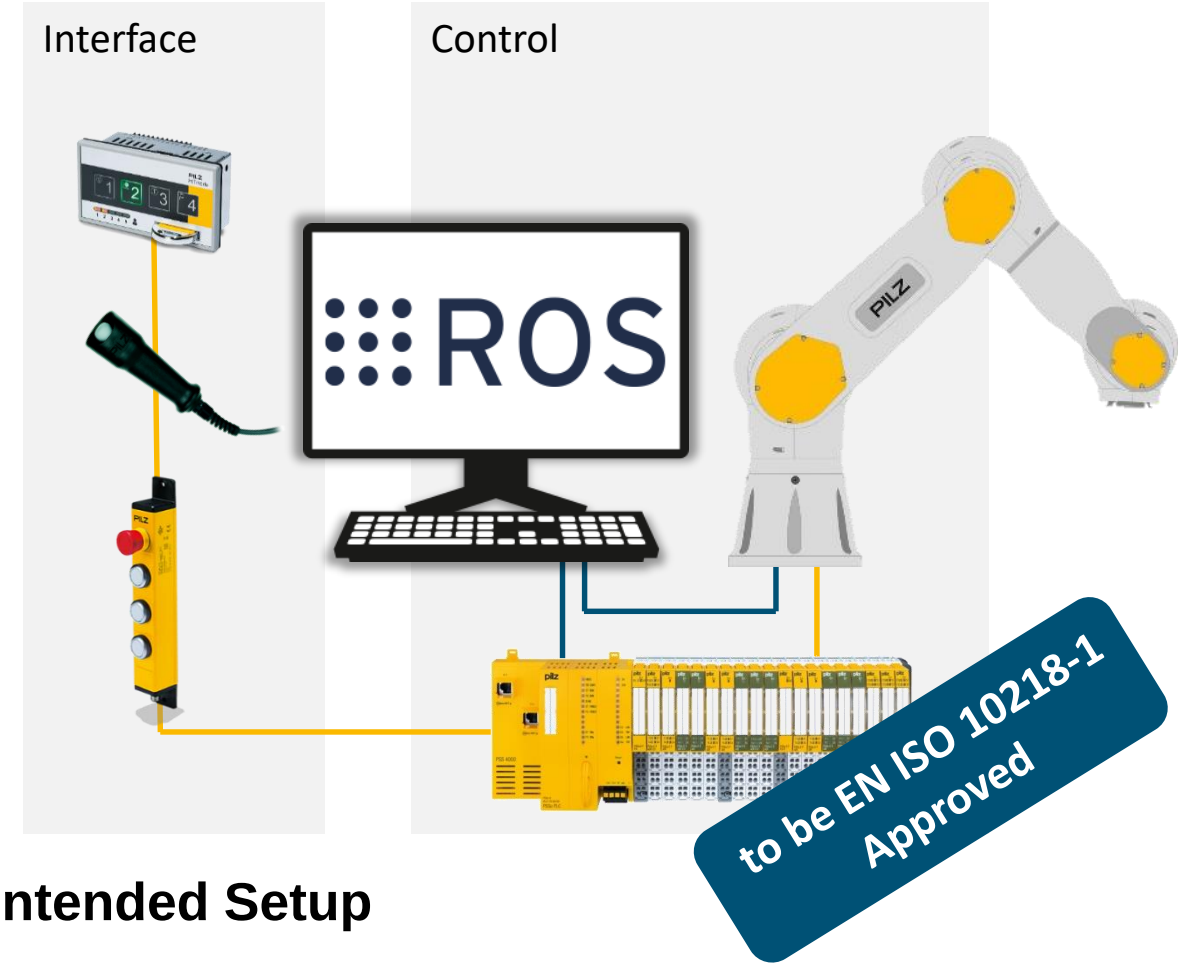
- **Establish ROS in Industrial Applications**
- Robot can be used with ROS
  No Proprietary Controller
  No Proprietary Teach Pendant
- → As much functionality as possible implemented in ROS
  \+ Safety Controller for Safety Functionality
- *Robot certifiable under EN ISO 10218-1*
  Applications are build purely in ROS
- *Integrator can focus on application*
  Safety provided with the robot
- Pilz offering ROS Product Training from next Year

System Overview



Traditional Setup

ROS would be merely an afterthought



Intended Setup

ROS as core component

ISO 10218-1:2011 Robots and robotic devices — Safety requirements for industrial robots — Part 1: Robots

Exemplary Aspects of the Standard:

Operational modes

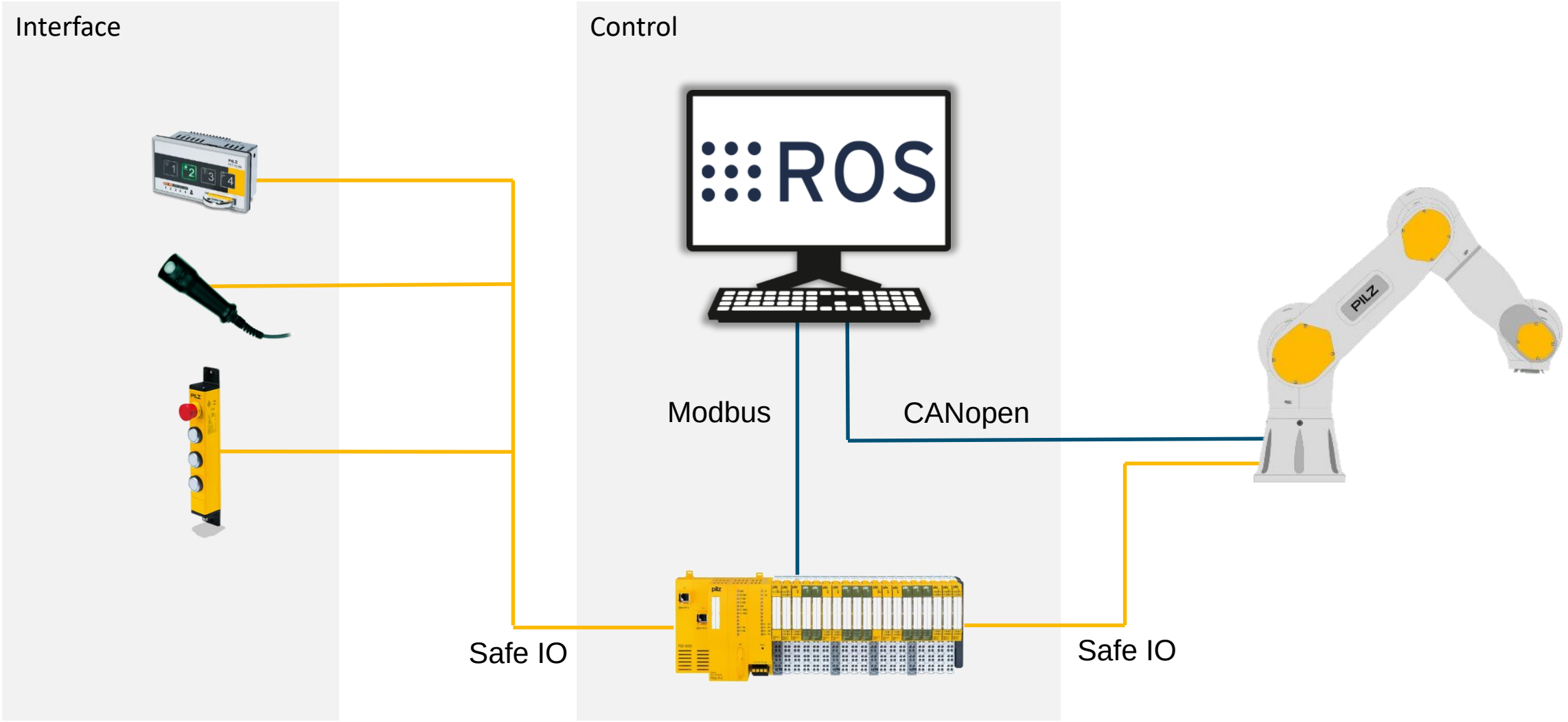
- Automatic / Manual Reduced Speed
  - Display of Mode
  - Monitoring of Reduced Speed

Robot stopping functions

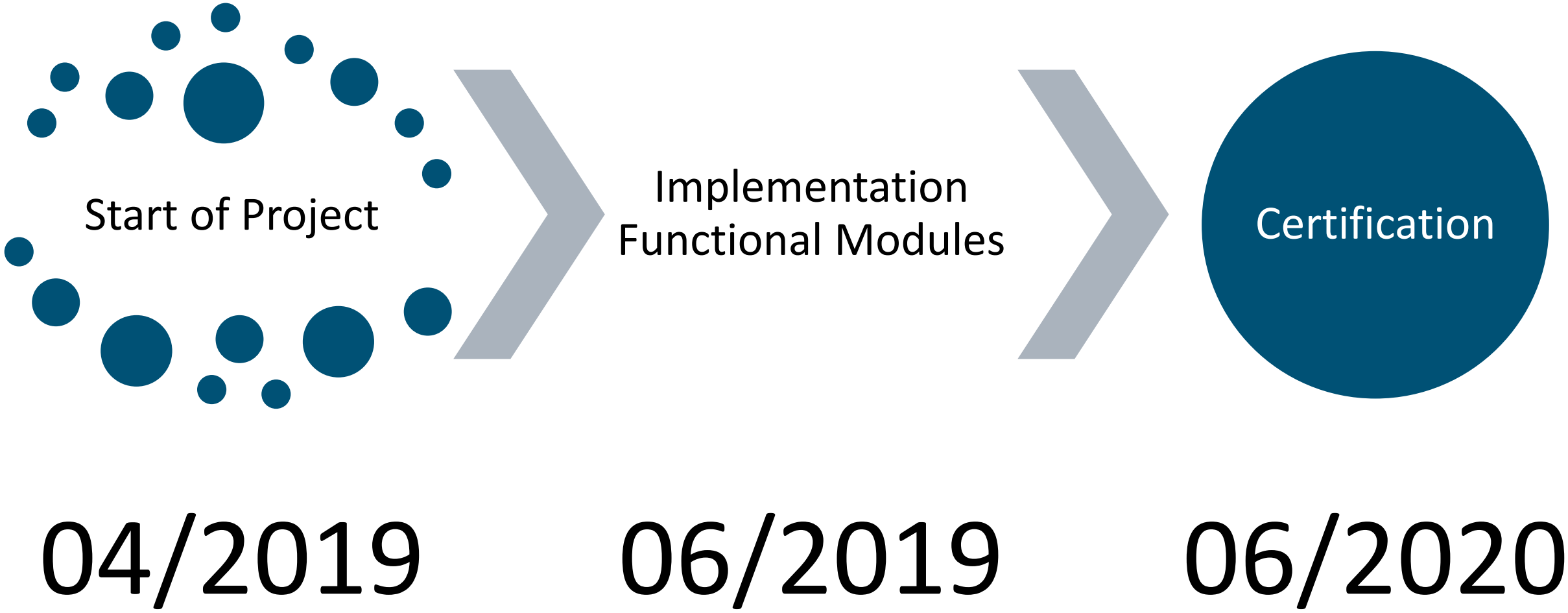
- Emergency stop
  - Smoothly stopping
  - Brakes in emergency
  - -> Brake Test
  - within time limit
  - Triggered from Pendant

Full Standard → https://www.iso.org/standard/51330.html

Technical overview



Development Progress



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

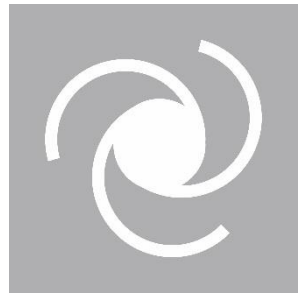
Example: Performing a brake test with Python API

```
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

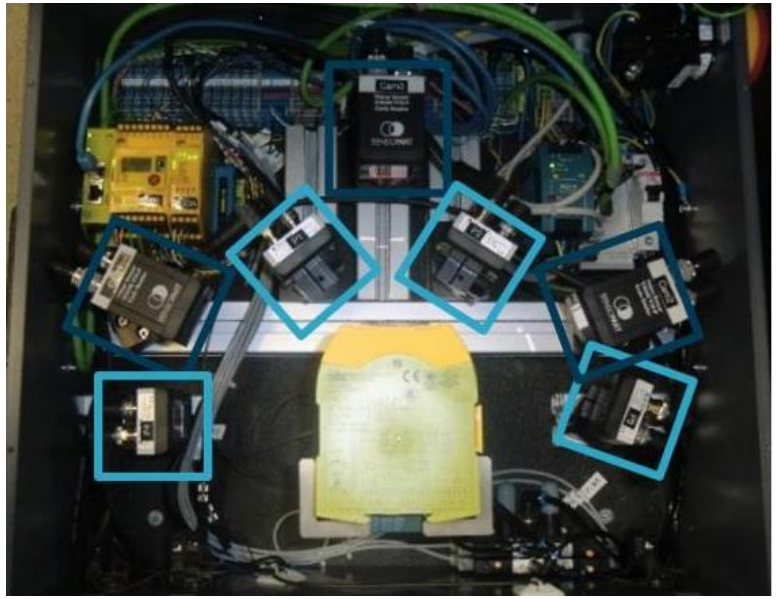
- Automatic
  - Automatic execution of predefined program
    - E.g. Script written in our API
- Manual reduced speed
  - Limit of speed to 250 mm/s
  - ROS will monitor any TF frame
  - Robot can be controlled by any method in ROS
- Manual high speed
  - Limit start at 250 mm/s but can be increased
  - Control from ROS
  - For teaching etc.



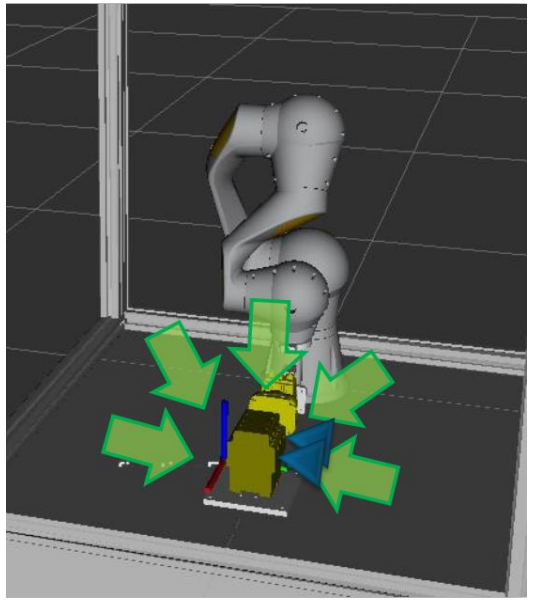
Example Application: Visual Inspection

- *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

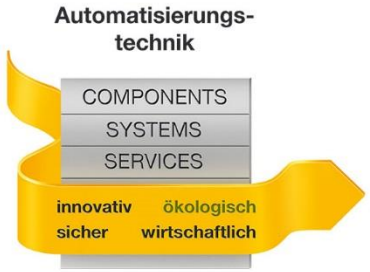
Machine setup

Summary

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



Christian Henkel

Pilz GmbH & Co. KG
Felix-Wankel-Straße 2
73760 Ostfildern, Germany
Tel.: +49 711 3409-7860
c.henkel@pilz.de
post@henkelchristian.de

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

PILZ
THE SPIRIT OF SAFETY