ROS2 CANopen

—

Supporting CANopen in ROS2
Design goals
Why complete reimplementation for ROS2?

- Understandability and Extendability
- Maintainability
- Robust Parallel Requests
- Plain ROS2 Interface
- ros2_control Interface
- Extensive Documentation
Overview

**General**
- Based on the Lely Core CANopen Stack
  ([https://gitlab.com/lely_industries/lely-core](https://gitlab.com/lely_industries/lely-core))
- Based on ROS2 components concept, CANopen master and drivers are components
- Don’t hardcode, configure
- Licensed under Apache 2.0 where possible (currently only canopen_402_driver needs to be under LGPLv3)
Configuration
Different Configuration Options

Master
- Definition node id
- Definition of component that provides the master driver
- Further master configuration options such as sync period or heartbeat are available

Driver
- Definition of name
- Definition of node id
- Definition of component that provides the master
- Definition of SDO calls that are automatically executed after device boot
- Definition of rpdo and tpdo configuration that is automatically configured after device boot
Bus Configuration
Automatic configuration artefact generation

@Designtime

- **OEM** creates **Device EDS**
- **User** creates **Bus Description**
- **CMAKE** generates **Master DCF**, **Master BIN**, **Device BIN**

@Runtime

- **User** launches **Device Container**
- **CMAKE** loads **CANopen Master**, **CANopen Drivers**
- **Device Container** loads **CANopen Drivers**
Operation
Different Operating Options

- **Working**
- **Under Development**

- **Real-time Control**
  - Driver’s functionality is exposed via ros2_control system interface

- **ROS2 control based**
- **Service based**
  - All drivers expose their functionality via ROS interface
  - All drivers and master are lifecycle nodes
  - Lifecycle device manager enables managing overall lifecycle

- **Lifecycle service based**
  - All drivers expose their functionality via ROS interface
  - All drivers and master are lifecycle nodes
  - Lifecycle device manager enables managing overall lifecycle

- **Robustness**
Service-based interface

Simplicity

On Launch

Configuration Files
- BIN
- DCF
- YAML

Device Container Node
- Loads
- BIN
- DCF

Master Node
- Configures
- Loads

Driver Node
- Loads

Executor
- Device Container Node
- Master Node
- Driver Node

Loads

YAML
DCF
BIN
Lifecycle service-based interface

Robustness
ros2_control System Interface
Controlling – Under Development

On Creation

Robot Description

ROS2 CANopen System Interface (ros2_control::SystemInterface)

Executor

Master Node

Driver Node

Loads from URDF:

YAML
BIN
DCF
Video
ros2_canopen with lifecycle service interface
Further Developments

• Final integration of ros2_control interface

• Streamlining of the different interfaces (removing code duplications etc.)

• Extensive testing (Pilz PRBT, Care-O-Bot and others planned)
Thank you for your attention!
Contact

Harsh Deshpande and Christoph Hellmann Santos
Fraunhofer IPA – Robots and assistive systems
Phone +49 711 970-1097
Christoph.hellmann.santos@ipa.fraunhofer.de

Fraunhofer IPA
Nobelstraße 12
70569 Stuttgart
Germany
www.ipa.fraunhofer.de

Future is our product
Sustainable. Personalized. Smart.