zub machine control AG
Buzibachstrasse 31
6023 Rothenburg
Schweiz

Telefon +41 (0)41 541 50 40
info@zub.ch
www.zub.ch

EPOS4 / ROS2

ros2_canopen

Date	27. Oktober 2024
Document number	12906999
Document index	01
Specific use	Document for external use

maxon | zub

Application notes on how to use maxon EPOS4 Positioning Controllers with ROS2 using the CANopen package ros2_canopen

Important notes:
This document and all provided sample code has been developed at maxon. Any installation steps and code samples are intended for testing purposes only.
Please adapt the code to the needs of your concrete application.
Any warranty for proper functionality is excluded and has to be ensured based on tests within the concrete system environment.
Take care of the wiring and safety instructions mentioned by the "Hardware Reference" of the corresponding controller!
Please submit a request on maxon's Support Center (-> http://support.maxongroup.com) in case of any questions concerning the controller's setup or wiring. No ROS support can be provided on this channel.

Content

1 About

1.1 About this Document

1.1.1 Intended Purpose

The purpose of this document is to help you integrate maxon EPOS4 controllers with ROS2 (Robot Operating System: https://www.ros.org/) using the software package **ros2_canopen** (https://ros-industrial.github.io/ros2_canopen/manual/rolling/index.html). This package is part of ROS-Industrial (https://rosindustrial.org/) which aims to bring ROS to the industry.
The present documentation goes along with the ROS2 package **maxon_epos4_ros2** as an example to get started.

1.1.2 Target Audience

This document is written for users who have basic knowledge of CANopen (https://www.can-cia.org/can-knowledge/canopen) and ROS2 (https://docs.ros.org/en/rolling/). It contains all the necessary links to extend your knowledge on the topic, but it mainly focuses on specificities related to the EPOS4 and **ros2_canopen** setup.
You can get more knowledge about CANopen on the following link from maxon's Technical Support: https://support.maxongroup.com/hc/en-us/articles/360017882660

1.1.3 Use Cases

ROS2 is suitable for applications where it is beneficial to have modular architecture and support for distributed systems in robotics and automation. It officially runs on the Ubuntu Linux distribution.

With the help of this document and the examples in the **maxon_epos4_ros2** package, users have a practical guide for setting up a motion control system in ROS2 that connects to EPOS4 motor controllers via CAN bus. The goal is to enable motor control directly from the ROS2 environment.

The code examples contained in the **maxon_epos4_ros2** package describe simple use cases, which can be easily extended to more controllers for a specific application.
At this point the example supports the drive operation mode "Profile Position Mode" (PPM), "Profile Velocity Mode" (PVM), "Cyclic Synchronous Position Mode" (CSP) and "Cyclic Synchronous Velocity Mode" (CSV).

The practical cases described at the end of the documentation focus on the integration with a NVIDIA Jetson Orin Nano computer and a Raspberry Pi 5 coupled with a CAN HAT.

To get a working system the following steps are required:

1) Setup the EPOS4

2) Setup the ROS2 environment
3) Setup the CAN driver and SocketCAN
4) Setup the ros2_canopen package
5) Install and run the “maxon_epos4_ros2” examples

1.1.4 Conventions

All the commands that need to be executed in a terminal are highlighted in in grey. Some commands might take two lines on the documentation but should still be entered as one line. Here is an example of a command:

```
git clone https://github.com/ros-industrial/ros2_canopen.git
```

File names, source code, ROS package names and parameters are written in bold, for example:

maxon_epos4_ros2, **/cia402_device_1**

1.1.5 Symbols & Signs

Type	Symbol	Meaning
DANGER safety alert		Indicates an **imminent hazardous situation**. If not avoided, it **will result in death or serious injury**.
WARNING		Indicates a **potential hazardous situation**. If not avoided, it **can result in death or serious injury**.
CAUTION		Indicates a **probable hazardous situation** or calls the attention to unsafe practices. If not avoided, it **may result in injury**.

Prohibited action		Indicates a dangerous action. Hence, **you must not!**
Mandatory action		Indicates a mandatory action. Hence, **you must!**
Requirement, Note, Remark		Indicates an activity you must perform prior continuing, or gives information on a particular item you need to observe.
Best practice		Indicates an advice or recommendation on the easiest and best way to further proceed.
Material Damage		Indicates information particular to possible damage of the equipment.

1.1.6 Abbreviations

The following abbreviations are used in the documentation:

- ROS: Robot Operating System (if not specified ROS2 is meant)
- CAN: Controller Area Network
- SDO: Service Data Object
- PDO: Process Data Object
- EDS: Electronic Data Sheet
- DCF: Device Configuration File
- PPM: Profile Position Mode
- PVM: Profile Velocity Mode
- HM: Homing Mode
- CSP Mode: Cyclic Synchronous Position Mode
- CSV Mode: Cyclic Synchronous Velocity Mode
- YAML: YAML Ain't Markup Language
- CiA402: CANopen device profile for motion control

2 EPOS4 Setup

The first step is to have a working EPOS4 that is configured correctly to the connected motor and sensors (encoder and/or hall). This whole step can be done and tested with maxon **EPOS Studio**.

2.1 Hardware Setup

For wiring the EPOS4, please refer to the official maxon EPOS4 Hardware Reference manual corresponding to your controller. You will find it in the Download section after selecting your controller: http://epos.maxongroup.com/

2.2 Software Setup with EPOS Studio

2.2.1 Installation

The installation file can be downloaded from the maxon web shop under the download section of the chosen EPOS4 controller.

Extract the .zip file and execute the file “EPOS-IDX-setup.exe”

2.2.2 Run the wizard for the EPOS4

Just run the wizard to do the correct configuration of the EPOS4.

If you have more than one EPOS4 with the same motor configuration you can export the parameters to a file a load the same configuration to the other EPOS4.

2.2.3 Autotuning

Use the autotune functionality in EPOS Studio to tune the Current, velocity and position control loop.

2.3 EPOS4 CAN settings

It is important that basic CAN communication works through the entire CAN network. Therefore a few settings must be done beforehand.

2.3.1 CAN bitrate

In our example we want use the CAN bus with its maximal performance, The bitrate is 1Mbaud (1000kB) for the whole CAN network. Also set the CAN bitrate for the EPOS4 fix to 1000kB. The EPOS4 has activated the auto bitrate detection by default. Because we start to communicate with SDO telegrams directly, you must turn off this feature by setting the according DIP switch to OFF. When you change the bitrate, it is important that you do a power cycle on the device.

2.3.2 CAN Node-ID

All devices on the CAN network must have unique Node-ID. The Node-ID is set via DIP switch.
The actual Node-ID can be verified in the EPOS Studio. For the example described in this document we use Node-ID 1 and 2 (the example with a single EPOS4 Node-ID is used).

2.3.3 PDO mapping

For exchanging process data between CAN master and all the slave devices PDO communication is used. Because the PDO contains plain data only, master and slave must specify what data are exchanged with its location and size within the telegram. There are PDOs from and to each slave device.

There is no need to do the PDO mapping configuration with EPOS Studio.
The CAN master in the ros2_canopen package does the PDO mapping in the initialization phase. So, the master writes to the mapping objects in the slave device using SDO communication. Previous settings are overwritten. All the mapping information is given in the **bus.yml** file.

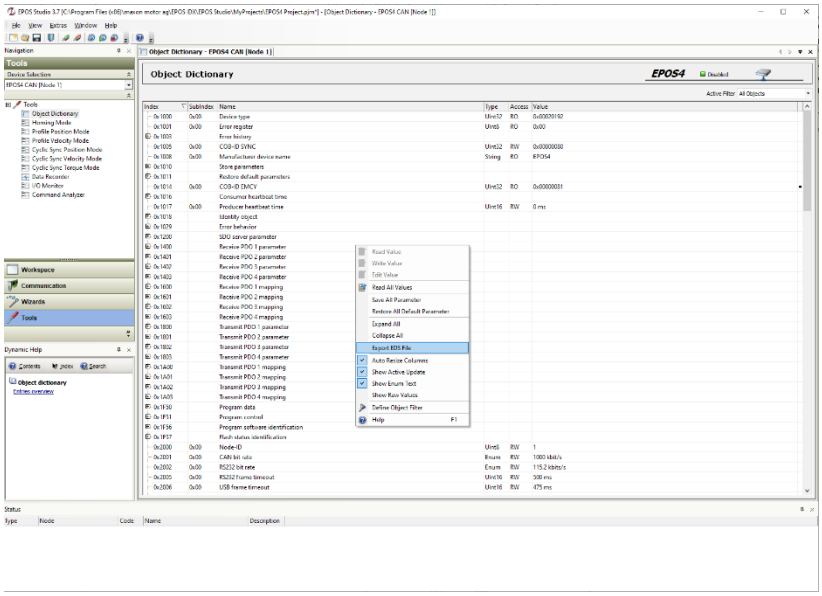
2.4 EPOS4 device description file (.eds)

For the use with ros2_canopen package we must provide a .eds file for each slave device (EPOS4). The .eds file holds all the drive object information in a single file. It lists also all supported SDO objects with additional information on whether the object can be mapped into a PDO. This information is needed by ROS2 CAN master.

2.4.1 Creating a .eds file

- Start EPOS Studio
- Connect your EPOS4 to EPOS Studio

- Open the **"Object Dictionary" tool**.
- **Right-click** on any line in the "Object Dictionary".
- Select **"Export EDS File"**



https://support.maxongroup.com/hc/en-us/articles/360012778713-EPOS-IDX-Export-of-the-eds-resp-esi-file-Electronic-Data-Sheet

In the EPOS4.eds file of an EPOS4 all RxPDO are valid by default. This confuses the ros2_canopen stack and enable all those PDOs so that they are sent by the master. To avoid that, you can disable the additional RxPDOs by setting the NO VALID bit in objects [1401sub1], [1402sub1] and [1403sub1].
For example: (DefaultValue=$NODEID+0x**8**0000400)

```
586  [1401sub1]
587  ParameterName=COB-ID used by RxPDO 2
588  ObjectType=0x7
589  DataType=0x7
590  AccessType=rw
591  DefaultValue=$NODEID+0x80000300   ;before $NODEID+0x00000300
592  PDOMapping=0
593  ObjFlags=0
```

2.5 Getting help

More information and further knowledge you can find on the maxon support website.

https://support.maxongroup.com/hc/en-us/categories/360000006434-Technical-Support

If there are any questions regarding setting up the EPOS4 or questions about EPOS Studio the maxon support team is pleased to help you.

You can contact us by submitting a support request.

https://support.maxongroup.com/hc/en-us/requests/new

3 Mode of operation explained

There are different modes of operation that can be used on an EPOS4.

3.1 Profile Position Mode (PPM)

The EPOS4 supports the Profile Position Mode (PPM). In this mode a single target position is sent to the EPOS4 with a start command in the control word. The trajectory with a trapezoidal ramp type is calculated in the EPOS4 and the active position controller makes the motor follow the profile to its commanded position.

This mode is simple to use for the master and ros2_canopen can handle this mode with its standard configuration by providing a simple bus.yml file.

3.2 Cyclic Synchronous Position (CSP) Mode

In one of the examples, we use CSP (cyclic synchronous position) mode. With this mode the “target position” is updated in every cycle. This allows the master to command the motor with any given trajectory that is calculated in the master itself. The position control loop is active on the EPOS4.

In case you are using CSP mode, it is important to adjust the CAN bus cycle time of updated “Target position” data properly by setting the EPOS4 object “Interpolation time period value” (0x60C2/01). It is recommended to use a value of 10ms.
The master does write this value to the EPOS4, when specified in the bus.yml file.

Please find more information about this by maxon’s public Support Center document.

EPOS4 / IDX: CSP or CSV Mode -> Relevance of "Interpolation Time Period Value"? – maxon Support

3.3 Profile Velocity Mode (PVM)

The EPOS4 supports the Profile Velocity Mode (PVM). In this mode a target velocity is sent to the EPOS4 with a start command in the control word. The velocity is then increased with the given ramp until the target velocity is reached. The default unit for velocities is rpm. In the EPOS4 a velocity controller is activated.

This mode is simple to use for the master and ros2_canopen can handle this mode with its standard configuration by providing a simple bus.yml file.

3.4 Cyclic Synchronous Velocity (CSV) Mode

In one of the examples, we use CSV (cyclic synchronous velocity) mode. With this mode the “target velocity” is updated in every cycle. This allows the master to command the motor with any given trajectory that is calculated in the master itself. The velocity control loop is active on the EPOS4.

In case you are using CSV mode, it is important to adjust the CAN bus cycle time of updated “Target velocity” data properly by setting the EPOS4 object “Interpolation time period value” (0x60C2/01). It is recommended to use a value of 10ms.
The master does write this value to the EPOS4, when specified in the bus.yml file.

Please find more information about this by maxon’s public Support Center document.

EPOS4 / IDX: CSP or CSV Mode -> Relevance of "Interpolation Time Period Value"? – maxon Support

4 CAN Setup

Setting up a CAN bus is highly device dependent. Please refer to the chapter:

Practical case with NVIDIA Jetson Orin Nano (chapter 8)

or

Practical case with Raspberry Pi 5 and CAN-HAT (chapter 9)

For the general steps you can continue with the next chapter.

5 ROS2 Setup

The next step is to have a working ROS2 installation on your target Linux system. The target must provide SocketCAN support (hardware and Linux drivers).

5.1 ROS2 installation

An installation of a ROS2 version goes along with a specific Linux Ubuntu version.

ROS2 Humble requires Ubuntu 22.04 LTS “Jammy Jellyfish”
ROS2 Jazzy requires Ubuntu 24.04 LTS “Noble Numbat”

On some hardware only a specific Ubuntu version is supported. That is the reason that we use

- Ubuntu 22.04 on the Nvidia Jetson Nano with ROS2 “Humble”
- Ubuntu 24.04 on the Raspberry Pi 5 with ROS2 “Jazzy”

The ROS2 installation is a straightforward process described in detail here:

https://docs.ros.org/en/jazzy/Installation/Ubuntu-Install-Debs.html

For the examples it is fine to use ROS-base install without GUI tools. Desktop install works fine, too.

5.2 Setting up the ROS2 workspace

The ROS2 workspace is just a folder on the Linux system to where you can clone github repositories. It is recommended to use the suggested “ros2_ws” folder in the home directory.

5.3 Additional installations

Some ROS packages are available as apt repository and can be installed with
“sudo apt install NAME”

5.3.1 Install ros2_control

We need ros2_control:

```
sudo apt install ros2_control
```

6 Setup ros2_canopen

In this chapter we make sure we install the correct branch of the **ros2_canopen** package on the target system. Additional information and instructions can be found on the GitHub package site of ros-industrial:

https://ros-industrial.github.io/ros2_canopen/manual/rolling/user-guide/operation.html

6.1 ros2_canopen installation

ros2_canopen is provided as a package that must be cloned into the workspace and then build with **colcon**. Before clone go to the /src (source) folder inside the workspace.

```
cd ~/ros2_ws/src
```

For **Jazzy** you can clone the package by using the command:

```
git clone https://github.com/ros-industrial/ros2_canopen.git
```

For **Humble** use the humble branch:

```
git clone -b humble https://github.com/ros-industrial/ros2_canopen.git
```

go back to the ros2_ws folder, check dependencies with **rosdep** and build the package with **colcon**:

```
cd ..
source /opt/ros/jazzy/setup.bash    or    source /opt/ros/humble/setup.bash

sudo rosdep init
rosdep update

rosdep install --from-paths src/ros2_canopen --ignore-src -r -y
colcon build
```

There are a few warnings during the build and **colcon** doesn't finish with the "success" message, but this should not be a major issue. It is ok when the output messages look similar to this:

```
--- stderr: canopen_ros2_controllers
In file included from /home/mk/ros2_ws/src/ros2_canopen/canopen_ros2_controllers/src/cia402_device_controller.cpp:15:
/home/mk/ros2_ws/src/ros2_canopen/canopen_ros2_controllers/include/canopen_ros2_controllers/cia402_device_controller.hpp: In lamb
da function:
/home/mk/ros2_ws/src/ros2_canopen/canopen_ros2_controllers/include/canopen_ros2_controllers/cia402_device_controller.hpp:100:45:
warning: ignoring return value of ‘bool hardware_interface::LoanedCommandInterface::set_value(const T&, unsigned int) [with T = d
ouble]’, declared with attribute ‘nodiscard’ [-Wunused-result]
  100 |           command_interfaces_[cmd].set_value(kCommandValue);
      |           ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~
In file included from /opt/ros/jazzy/include/controller_interface/controller_interface/controller_interface_base.hpp:27,
                 from /opt/ros/jazzy/include/controller_interface/controller_interface/controller_interface.hpp:21,
                 from /home/mk/ros2_ws/src/ros2_canopen/canopen_ros2_controllers/include/canopen_ros2_controllers/canopen_proxy_c
ontroller.hpp:26,
                 from /home/mk/ros2_ws/src/ros2_canopen/canopen_ros2_controllers/include/canopen_ros2_controllers/cia402_device_c
ontroller.hpp:19:
/opt/ros/jazzy/include/hardware_interface/hardware_interface/loaned_command_interface.hpp:113:22: note: declared here
  113 |   [[nodiscard]] bool set_value(const T & value, unsigned int max_tries = 10)
      |                      ^~~~~~~~~
/home/mk/ros2_ws/src/ros2_canopen/canopen_ros2_controllers/include/canopen_ros2_controllers/cia402_device_controller.hpp:102:63:
warning: ‘double hardware_interface::LoanedCommandInterface::get_value() const’ is deprecated: Use std::optional<T> get_optional(
) instead to retrieve the value. This method will be removed by the ROS 2 Kilted Kaiju release. [-Wdeprecated-declarations]
  102 |           while (std::isnan(command_interfaces_[fbk].get_value()) && rclcpp::ok())
      |                             ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~^~
/opt/ros/jazzy/include/hardware_interface/hardware_interface/loaned_command_interface.hpp:134:10: note: declared here
  134 |   double get_value() const
      |          ^~~~~~~~~
/home/mk/ros2_ws/src/ros2_canopen/canopen_ros2_controllers/include/canopen_ros2_controllers/cia402_device_controller.hpp:108:83:
warning: ‘double hardware_interface::LoanedCommandInterface::get_value() const’ is deprecated: Use std::optional<T> get_optional(
) instead to retrieve the value. This method will be removed by the ROS 2 Kilted Kaiju release. [-Wdeprecated-declarations]
  108 |           response->success = static_cast<bool>(command_interfaces_[fbk].get_value());
      |                                                 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~^~
/opt/ros/jazzy/include/hardware_interface/hardware_interface/loaned_command_interface.hpp:134:10: note: declared here
  134 |   double get_value() const
      |          ^~~~~~~~~
```

The final output window of the **colcon** build is:

```
mk@mk-pi: ~/ros2_ws 130x24
e ‘nodiscard’ [-Wunused-result]
  334 |     command_interfaces_[CommandInterfaces::TPDO_DATA].set_value(
      |     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~^
  335 |       static_cast<double>((*current_cmd)->data));
      |       ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/opt/ros/jazzy/include/hardware_interface/hardware_interface/loaned_command_interface.hpp:113:22: note: declared here
  113 |   [[nodiscard]] bool set_value(const T & value, unsigned int max_tries = 10)
      |                      ^~~~~~~~~
/home/mk/ros2_ws/src/ros2_canopen/canopen_ros2_controllers/src/canopen_proxy_controller.cpp:337:63: warning: ignoring return value
 of ‘bool hardware_interface::LoanedCommandInterface::set_value(const T&, unsigned int) [with T = double]’, declared with attribut
e ‘nodiscard’ [-Wunused-result]
  337 |     command_interfaces_[CommandInterfaces::TPDO_ONS].set_value(kCommandValue);
      |     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~
/opt/ros/jazzy/include/hardware_interface/hardware_interface/loaned_command_interface.hpp:113:22: note: declared here
  113 |   [[nodiscard]] bool set_value(const T & value, unsigned int max_tries = 10)
      |                      ^~~~~~~~~
---
Finished <<< canopen_ros2_controllers [56.9s]
Starting >>> canopen_tests
Finished <<< canopen_tests [9.96s]

Summary: 15 packages finished [5min 36s]
  3 packages had stderr output: canopen_ros2_control canopen_ros2_controllers lely_core_libraries
mk@mk-pi:~/ros2_ws$
```

6.1.1 ros2_canopen Links

For the **ros2_canopen** documentation use:
https://ros-industrial.github.io/ros2_canopen/manual/rolling/index.html

The source of the **ros2_canopen** package can be found in github under:

https://github.com/ros-industrial/ros2_canopen/tree/master

6.1.2 Testing ros2_canopen

For testing the **ros2_canopen** you can do a few simple test that uses a virtual can bus (vcan0) and also virtual can drives.

The virtual can bus must be setup with the next following commands:

```
sudo modprobe vcan

sudo ip link add dev vcan0 type vcan

sudo ip link set vcan0 txqueuelen 1000

sudo ip link set up vcan0
```

Follow the instructions on in the **ros2_canopen** documentation in the chapter “Running Examples” following the link below. Because the **ros2_canopen** package was built locally, you must make sure you source the installation as well. Go to a new terminal and enter the three commands.

```
source /opt/ros/jazzy/setup.bash

cd ros2_ws

source install/setup.bash
```

Running the examples:

Running Examples — ros2_canopen 0.0.1 documentation

7 maxon EPOS4 ROS2 package (maxon_epos4_ros2)

That you can use the **ros2_canopen** package with all the settings that are used for your hardware configuration. The concept of ros2_canopen is that you have a launch file that uses a configuration folder with a bus.yml and a .eds or .dcf file of the device. Because we have motion contrroller, we use the cia402 functionality that are provided by ros2_canopen. During the colcon build, there is a mechanism with a tool “cogen_dcf” that comes along with ros2_canopen. “cogen_dcf” takes the configuration folder (see in CMakeLists.txt) and generates a master.dcf and master.bin file for the CAN master. So, there is no need to provide the later files by the user.

7.1 general description

The **maxon_epos4_ros2** package is a ROS2 package that holds examples on how to use **ros2_canopen** with EPOS4 motor controller over CAN bus.
It contains four different examples, showing different modes of operation:

- ppm (profile position mode)
- csp (cyclic synchronous position mode)
- pvm (profile velocity mode)
- csv (cyclic synchronous velocity mode)

Note that position examples and velocity examples use different configuration settings.

The following steps are needed:

- Clone the package with git clone command
- Provide the correct .eds file
- Adjust the configuration file
- Build with colcon
- Test the examples

The **maxon_epos4_ros2** package holds two different configurations. Both use the cia402 configuration. One sets up all the configuration for position operation and the other one for velocity operation. The difference between position configuration and velocity configuration is the PDO mapping and some start configuration.
It is not possible to start the position configuration and velocity configuration at the same time.

In this example configuration we use the following CAN driver.

maxon **EPOS4 Compact 24/1.5 CAN** motor controller with the part number 546714. The product code of this specific controller is 0x6050.

If you are using a different controller, make sure you provide the .eds file in the config folder and change the **bus.yml** file.
For this starting example we use one EPOS4 motor controller only. It has the node_id 1. Later it should be no problem to add a second EPOS4. In the **bus.yml** file there is already a block which is

commented out. This shows how to add another drive. In the **bus.yml** there is also a node-id setting for the master. This node_id value must be different from any connected device. It is recommended that all EPOS4 are used in the same operation mode.

The name of the .eds file that is used for the EPOS4 and given in the **bus.yml** file is maxon_epos4_0x6050.eds. I you use a different EPOS4 you must replace the name in the **bus.yml** and make sure you have a copy in the folder config/bus_config_cia402_epos4_pos and config/bus_config_cia402_epos4_vel.

The directory tree of the **maxon_epos4_ros2** package looks like this:

```
maxon_epos4_ros2/
    ├─ documentation/
    │   └── EPOS4_ROS2_canopen_12906999-01.pdf
    ├─ maxon_epos4_ros2/
    │   ├── config/
    │   │   ├── bus_config_cia402_epos4_pos
    │   │   │   ├── bus.yml
    │   │   │   └── maxon_epos4_0x6050.eds
    │   │   └── bus_config_cia402_epos4_vel
    │   │       ├── bus.yml
    │   │       └── maxon_epos4_0x6050.eds
    │   ├── include/
    │   ├── launch/
    │   │   ├── bus_config_cia402_epos4_pos.launch.py
    │   │   └── bus_config_cia402_epos4_vel.launch.py
    │   ├── src/
    │   │   ├── epos4_pos.cpp
    │   │   └── epos4_vel.cpp
    │   ├── CMakeLists.txt
    │   └── package.xml
    └── LICENCE
    └── README.md
```

If you plan to do add any additional configuration yourself then it is important that you have a launch file with the same name as the configuration folder.

7.2 Installation and build

To install and build the **maxon_epos4_ros2** package you first must clone the package. Make sure you clone/copy the package into the folder /ros2_ws/src/

```
cd ~/ros2_ws/src
```

```
git clone https://github.com/MattMax319/maxon_epos4_ros2.git
```

After installing the package, we must build it with colcon. Before the build command go back to the working folder /ros2_ws

```
cd ..

colcon build --packages-select maxon_epos4_ros2
```

Before you run any launch file or node from the **maxon_epos4_ros2** package make sure you also source the working folder as:

```
cd ros2_ws
source install/setup.bash
```

7.3 Testing bus_config_cia402_epos4_pos

It is important that the canopen master with the configuration (bus_config_cia402_epos4_pos) is run before the epos4 commander node (**epos4_pos**). Also, before testing any of the command line service calls.

```
ros2 launch maxon_epos4_ros2 bus_config_cia402_epos4_pos.launch.py
```

The terminal output looks like this:

```
mk@mk-pi:~/ros2_ws$ ros2 launch maxon_epos4_ros2 bus_config_cia402_epos4_pos.launch.py
[INFO] [launch]: All log files can be found below /home/mk/.ros/log/2025-10-27-00-21-05-186588-mk-pi-2752
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [launch.user]: /home/mk/ros2_ws/install/maxon_epos4_ros2/share/maxon_epos4_ros2/config/bus_config_cia402_epos4_pos/bus.yml
[INFO] [launch.user]: /home/mk/ros2_ws/install/maxon_epos4_ros2/share/maxon_epos4_ros2/config/bus_config_cia402_epos4_pos/master.dcf
[INFO] [launch.user]: /home/mk/ros2_ws/install/maxon_epos4_ros2/share/maxon_epos4_ros2/config/bus_config_cia402_epos4_pos/master.bin
[INFO] [launch.user]: can0
[INFO] [device_container_node-1]: process started with pid [2769]
[device_container_node-1] [INFO] [1761520866.254629508] [device_container_node]: Starting Device Container with:
[device_container_node-1] [INFO] [1761520866.254791101] [device_container_node]:  master_config
/home/mk/ros2_ws/install/maxon_epos4_ros2/share/maxon_epos4_ros2/config/bus_config_cia402_epos4_pos/master.dcf
[device_container_node-1] [INFO] [1761520866.254819767] [device_container_node]:  bus_config
/home/mk/ros2_ws/install/maxon_epos4_ros2/share/maxon_epos4_ros2/config/bus_config_cia402_epos4_pos/bus.yml
[device_container_node-1] [INFO] [1761520866.254839267] [device_container_node]:  can_interface_name can0
[device_container_node-1] [INFO] [1761520866.256305990] [device_container_node]: Loading Master Configuration.
[device_container_node-1] [INFO] [1761520866.256722841] [yaml-cpp]: Failed to load entry "namespace" for device "master"
[device_container_node-1] [INFO] [1761520866.259201082] [device_container_node]: Load Library: /home/mk/ros2_ws/install/canopen_master_driver/lib/libmaster_driver.so
[device_container_node-1] [INFO] [1761520866.331505582] [device_container_node]: Found class: rclcpp_components::NodeFactoryTemplate<ros2_canopen::MasterDriver>
[device_container_node-1] [INFO] [1761520866.331585897] [device_container_node]: Instantiate class:
rclcpp_components::NodeFactoryTemplate<ros2_canopen::MasterDriver>
[device_container_node-1] [WARN] [1761520866.331665360] [device_container_node]: Namespace not provided for node 'master', using container namespace: /
[device_container_node-1] [INFO] [1761520866.345009230] [master]: NodeCanopenBasicMaster
[device_container_node-1] [INFO] [1761520866.345622397] [device_container_node]: Load master component.
[device_container_node-1] [INFO] [1761520866.345832545] [device_container_node]: Added /master to executor
[device_container_node-1] [INFO] [1761520866.354392378] [master]: Master boot timeout set to 1500ms.
[device_container_node-1] [INFO] [1761520866.388977730] [device_container_node]: Loading Driver Configuration.
[device_container_node-1] [INFO] [1761520866.389139415] [device_container_node]: Found device cia402_device_1 with driver ros2_canopen::Cia402Driver
[device_container_node-1] [INFO] [1761520866.390179304] [device_container_node]: Load Library: /home/mk/ros2_ws/install/canopen_402_driver/lib/libcia402_driver.so
[device_container_node-1] [INFO] [1761520866.568378471] [device_container_node]: Found class: rclcpp_components::NodeFactoryTemplate<ros2_canopen::Cia402Driver>
[device_container_node-1] [INFO] [1761520866.568469026] [device_container_node]: Instantiate class:
rclcpp_components::NodeFactoryTemplate<ros2_canopen::Cia402Driver>
[device_container_node-1] [INFO] [1761520866.568527026] [device_container_node]: Namespace set for node 'cia402_device_1': /
[device_container_node-1] [INFO] [1761520866.593274638] [device_container_node]: Load driver component.
[device_container_node-1] [INFO] [1761520866.593428582] [device_container_node]: Added /cia402_device_1 to executor
[device_container_node-1] [INFO] [1761520866.673655989] [cia402_device_1]: Non transmit timeout100ms
[device_container_node-1] [WARN] [1761520866.673823212] [cia402_device_1]: Could not polling from config, setting to true.
[device_container_node-1] [WARN] [1761520866.673986582] [cia402_device_1]: Could not read enable diagnostics from config, setting to false.
[device_container_node-1] [INFO] [1761520866.674547008] [cia402_device_1]: scale_pos_to_dev_ 1.000000
[device_container_node-1] scale_pos_from_dev_ 1.000000
[device_container_node-1] scale_vel_to_dev_ 1000.000000
[device_container_node-1] scale_vel_from_dev_ 0.001000
[device_container_node-1] offset_pos_to_dev_ 0.000000
[device_container_node-1] offset_pos_from_dev_ 0.000000
[device_container_node-1] homing_timeout_seconds_ 10
[device_container_node-1]
[device_container_node-1] [INFO] [1761520866.681508582] [cia402_device_1]: eds file /home/mk/ros2_ws/install/maxon_epos4_ros2/share/maxon_epos4_ros2/config/
bus_config_cia402_epos4_pos/maxon_epos4_0x6050.eds
[device_container_node-1] [INFO] [1761520866.681600675] [cia402_device_1]: bin file /home/mk/ros2_ws/install/maxon_epos4_ros2/share/maxon_epos4_ros2/config/
bus_config_cia402_epos4_pos/cia402_device_1.bin
[device_container_node-1] Found rpdo mapped object: index=6040 subindex=0
[device_container_node-1] Found rpdo mapped object: index=607a subindex=0
[device_container_node-1] Found rpdo mapped object: index=6060 subindex=0
[device_container_node-1] Found rpdo mapped object: index=6040 subindex=0
[device_container_node-1] Found rpdo mapped object: index=6060 subindex=0
[device_container_node-1] Found rpdo mapped object: index=6040 subindex=0
[device_container_node-1] Found rpdo mapped object: index=607a subindex=0
[device_container_node-1] Found rpdo mapped object: index=6040 subindex=0
[device_container_node-1] Found rpdo mapped object: index=60ff subindex=0
[device_container_node-1] Found tpdo mapped object: index=6041 subindex=0
[device_container_node-1] Found tpdo mapped object: index=6064 subindex=0
[device_container_node-1] Found tpdo mapped object: index=6061 subindex=0
[device_container_node-1] Found tpdo mapped object: index=6041 subindex=0
[device_container_node-1] Found tpdo mapped object: index=6061 subindex=0
[device_container_node-1] Found tpdo mapped object: index=6041 subindex=0
[device_container_node-1] Found tpdo mapped object: index=6064 subindex=0
[device_container_node-1] Found tpdo mapped object: index=6041 subindex=0
[device_container_node-1] Found tpdo mapped object: index=606c subindex=0
[device_container_node-1] [WARN] [1761520866.694005100] [cia402_device_1]: Wait for device to boot...
[device_container_node-1] [INFO] [1761520868.393564748] [cia402_device_1]: Driver booted and ready.
[device_container_node-1] [INFO] [1761520868.394435322] [cia402_device_1]: Starting with polling mode.
[device_container_node-1] [INFO] [1761520870.393532246] [cia402_device_1]: Slave 0x1: Switched NMT state to START
```

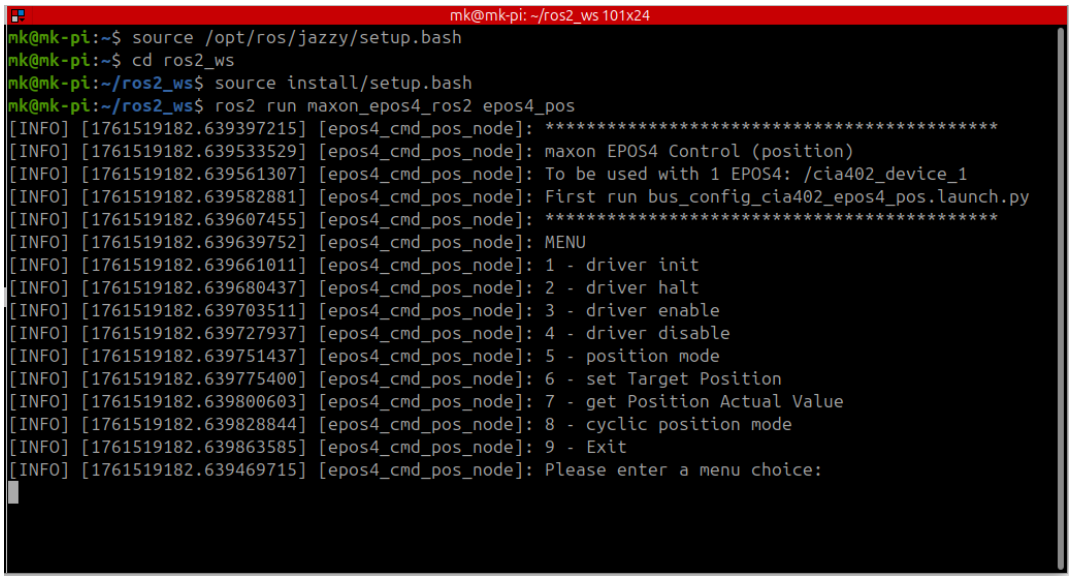
7.3.1 Testing with EPOS4 control node (pos)

This package provides a control node **epos4_pos** that demonstrates the use of service calls and publishes topics used by the cia402 device.
Service calls and publishing topics can also be done with ros2 command line but it is a bit annoying. But see below some examples for command line service calls.

Cyclic velocity mode uses a Sine wave generator to send velocity target values every 10ms via the topic **/cia402_device_1/tpdo** . Transmission PDO (tpdo) is here from the perspective of the CAN master.
You can leave the CSP mode and its moving by changing the mode back to position_mode, by pressing 5.

Run the **epos4_pos** node:

```
ros2 run maxon_epos4_ros2 epos4_pos
```



7.4 Testing bus_config_cia402_epos4_vel

It is important that the canopen master with the configuration (bus_config_cia402_epos4_pos) is run before the epos4 commander node (**epos4_vel**). Also, before testing any of the command line service calls. Just start the

```
ros2 launch maxon_epos4_ros2 bus_config_cia402_epos4_vel.launch.py
```

The terminal output looks like this:

```
mk@mk-pi:~/ros2_ws$ ros2 launch maxon_epos4_ros2 bus_config_cia402_epos4_vel.launch.py
[INFO] [launch]: All log files can be found below /home/mk/.ros/log/2025-10-27-00-26-19-854983-mk-pi-2979
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [launch.user]: /home/mk/ros2_ws/install/maxon_epos4_ros2/share/maxon_epos4_ros2/config/bus_config_cia402_epos4_vel/bus.yml
[INFO] [launch.user]: /home/mk/ros2_ws/install/maxon_epos4_ros2/share/maxon_epos4_ros2/config/bus_config_cia402_epos4_vel/master.dcf
[INFO] [launch.user]: /home/mk/ros2_ws/install/maxon_epos4_ros2/share/maxon_epos4_ros2/config/bus_config_cia402_epos4_vel/master.bin
[INFO] [launch.user]: can0
[INFO] [device_container_node-1]: process started with pid [2995]
[device_container_node-1] [INFO] [1761521180.408807525] [device_container_node]: Starting Device Container with:
[device_container_node-1] [INFO] [1761521180.408941562] [device_container_node]:  master_config
/home/mk/ros2_ws/install/maxon_epos4_ros2/share/maxon_epos4_ros2/config/bus_config_cia402_epos4_vel/master.dcf
[device_container_node-1] [INFO] [1761521180.408966710] [device_container_node]:  bus_config
/home/mk/ros2_ws/install/maxon_epos4_ros2/share/maxon_epos4_ros2/config/bus_config_cia402_epos4_vel/bus.yml
[device_container_node-1] [INFO] [1761521180.408978673] [device_container_node]:  can_interface_name can0
[device_container_node-1] [INFO] [1761521180.410190414] [device_container_node]: Loading Master Configuration.
[device_container_node-1] [INFO] [1761521180.410473802] [yaml-cpp]: Failed to load entry "namespace" for device "master"
[device_container_node-1] [INFO] [1761521180.411049321] [device_container_node]: Load Library: /home/mk/ros2_ws/install/canopen_master_driver/lib/libmaster_driver.so
[device_container_node-1] [INFO] [1761521180.431987710] [device_container_node]: Found class: rclcpp_components::NodeFactoryTemplate<ros2_canopen::MasterDriver>
[device_container_node-1] [INFO] [1761521180.432560951] [device_container_node]: Instantiate class:
rclcpp_components::NodeFactoryTemplate<ros2_canopen::MasterDriver>
[device_container_node-1] [WARN] [1761521180.433081062] [device_container_node]: Namespace not provided for node 'master', using container namespace: /
[device_container_node-1] [INFO] [1761521180.456819951] [master]: NodeCanopenBasicMaster
[device_container_node-1] [INFO] [1761521180.457297265] [device_container_node]: Load master component.
[device_container_node-1] [INFO] [1761521180.457537099] [device_container_node]: Added /master to executor
[device_container_node-1] [INFO] [1761521180.469891062] [master]: Master boot timeout set to 1500ms.
[device_container_node-1] [INFO] [1761521180.504019321] [device_container_node]: Loading Driver Configuration.
[device_container_node-1] [INFO] [1761521180.504952988] [device_container_node]: Found device cia402_device_1 with driver ros2_canopen::Cia402Driver
[device_container_node-1] [INFO] [1761521180.507630562] [device_container_node]: Load Library: /home/mk/ros2_ws/install/canopen_402_driver/lib/libcia402_driver.so
[device_container_node-1] [INFO] [1761521180.555187154] [device_container_node]: Found class: rclcpp_components::NodeFactoryTemplate<ros2_canopen::Cia402Driver>
[device_container_node-1] [INFO] [1761521180.555342950] [device_container_node]: Instantiate class:
rclcpp_components::NodeFactoryTemplate<ros2_canopen::Cia402Driver>
[device_container_node-1] [INFO] [1761521180.555423580] [device_container_node]: Namespace set for node 'cia402_device_1': /
[device_container_node-1] [INFO] [1761521180.590742710] [device_container_node]: Load driver component.
[device_container_node-1] [INFO] [1761521180.592014247] [device_container_node]: Added /cia402_device_1 to executor
[device_container_node-1] [INFO] [1761521180.685682247] [cia402_device_1]: Non transmit timeout100ms
[device_container_node-1] [WARN] [1761521180.685884191] [cia402_device_1]: Could not polling from config, setting to true.
[device_container_node-1] [WARN] [1761521180.686079617] [cia402_device_1]: Could not read enable diagnostics from config, setting to false.
[device_container_node-1] [INFO] [1761521180.686712469] [cia402_device_1]: scale_pos_to_dev_ 1.000000
[device_container_node-1] scale_pos_from_dev_ 1.000000
[device_container_node-1] scale_vel_to_dev_ 1.000000
[device_container_node-1] scale_vel_from_dev_ 1.000000
[device_container_node-1] offset_pos_to_dev_ 0.000000
[device_container_node-1] offset_pos_from_dev_ 0.000000
[device_container_node-1] homing_timeout_seconds_ 10
[device_container_node-1]
[device_container_node-1] [INFO] [1761521180.690310228] [cia402_device_1]: eds file /home/mk/ros2_ws/install/maxon_epos4_ros2/share/maxon_epos4_ros2/config/
bus_config_cia402_epos4_vel/maxon_epos4_0x6050.eds
[device_container_node-1] [INFO] [1761521180.690375969] [cia402_device_1]: bin file /home/mk/ros2_ws/install/maxon_epos4_ros2/share/maxon_epos4_ros2/config/
bus_config_cia402_epos4_vel/cia402_device_1.bin
[device_container_node-1] Found rpdo mapped object: index=6040 subindex=0
[device_container_node-1] Found rpdo mapped object: index=60ff subindex=0
[device_container_node-1] Found rpdo mapped object: index=6060 subindex=0
[device_container_node-1] Found rpdo mapped object: index=6040 subindex=0
[device_container_node-1] Found rpdo mapped object: index=6060 subindex=0
[device_container_node-1] Found rpdo mapped object: index=6040 subindex=0
[device_container_node-1] Found rpdo mapped object: index=607a subindex=0
[device_container_node-1] Found rpdo mapped object: index=6040 subindex=0
[device_container_node-1] Found rpdo mapped object: index=60ff subindex=0
[device_container_node-1] Found tpdo mapped object: index=6041 subindex=0
[device_container_node-1] Found tpdo mapped object: index=6064 subindex=0
[device_container_node-1] Found tpdo mapped object: index=6061 subindex=0
[device_container_node-1] Found tpdo mapped object: index=6041 subindex=0
[device_container_node-1] Found tpdo mapped object: index=6061 subindex=0
[device_container_node-1] Found tpdo mapped object: index=6041 subindex=0
[device_container_node-1] Found tpdo mapped object: index=6064 subindex=0
[device_container_node-1] Found tpdo mapped object: index=6041 subindex=0
[device_container_node-1] Found tpdo mapped object: index=606c subindex=0
[device_container_node-1] [WARN] [1761521180.707220024] [cia402_device_1]: Wait for device to boot...
[device_container_node-1] [INFO] [1761521182.512723653] [cia402_device_1]: Driver booted and ready.
[device_container_node-1] [INFO] [1761521182.513533505] [cia402_device_1]: Starting with polling mode.
[device_container_node-1] [INFO] [1761521184.512659837] [cia402_device_1]: Slave 0x1: Switched NMT state to START
```

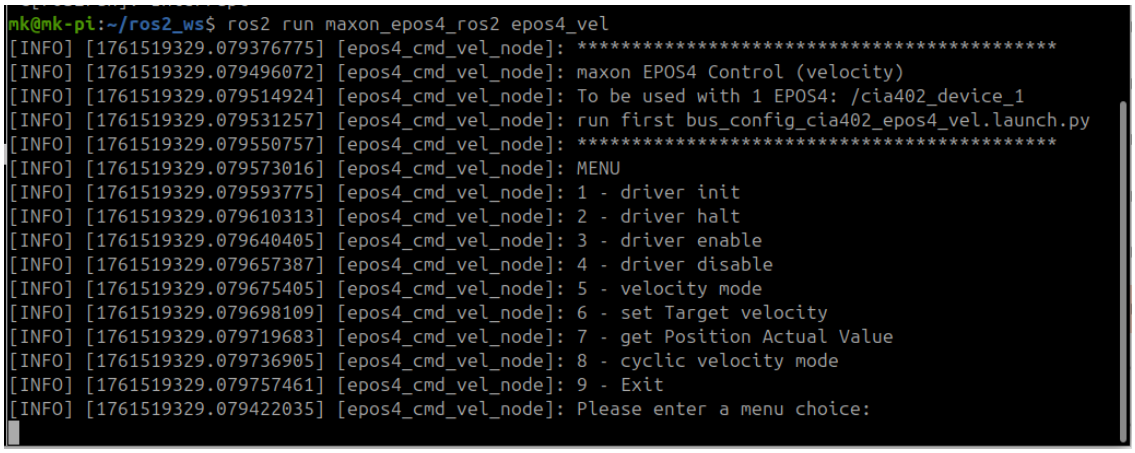
7.4.1 Testing with EPOS4 control node (vel)

This package provides a control node **epos4_vel** that demonstrates the use of service calls and publishes topics used by the cia402 device.
Service calls and publishing topics can also be done with ros2 command line but it is a bit annoying. But see below some examples for command line service calls.

Cyclic velocity mode uses a Sine wave generator to send velocity target values every 10ms via the topic **/cia402_device_1/tpdo** . Transmission PDO (tpdo) is here from the perspective of the CAN master.
You can leave the CSV mode and its moving by changing the mode back to velocity_mode, by pressing 5.

Run the **epos4_vel** node:

```
ros2 run maxon_epos4_ros2 epos4_vel
```



7.4.2 Known issues

At the moment there is an issue in the **ros2_canopen** when using the **/init** service command. Somehow the homing mode timeouts and it returns failed. But the drive is activated anyway and you can go on with testing.
Another issue is that a service call /position_mode fails when the position mode is already active. I hope these problems are fixed in a later version of **ros2_canopen**.

7.4.3 Testing with service calls in the command line

Make sure you do source the work space as well. This is because some of the service types are defined in the **canopen_interfaces** package.

```
cd ros2_ws
source install/setup.bash
```

Here is a list of possible service calls:

```
ros2 service call /cia402_device_1/init std_srvs/srv/Trigger
```

```
ros2 service call /cia402_device_1/position_mode std_srvs/srv/Trigger
```

move the motor to a target position:

```
ros2 service call /cia402_device_1/target canopen_interfaces/srv/COTargetDouble "{ target: 20000.0 }"
```

and back to zero position:

```
ros2 service call /cia402_device_1/target canopen_interfaces/srv/COTargetDouble "{ target: 0.0 }"
```

It is possible to use **/sdo_read** and **/sdo_write** services

```
ros2 service call /cia402_device_1/sdo_read canopen_interfaces/srv/CORead "{ index: 0x6064, subindex: 0 }"
```

```
ros2 service call /cia402_device_1/sdo_write canopen_interfaces/srv/COWrite "{ index: 0x6098, subindex: 0, data: 27 }"
```

Note that “data” is of type uint32. Therefore, negative numbers must be casted to this type beforehand.

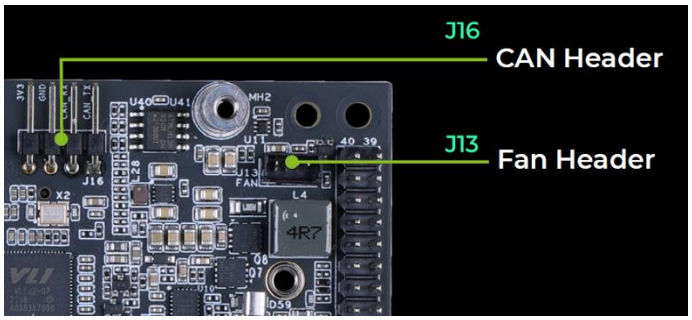
```
ros2 service call /cia402_device_1/sdo_read canopen_interfaces/srv/CORead "{ index: 0x6098, subindex: 0 }"
```

8 Practical case with NVIDIA Jetson Orin Nano

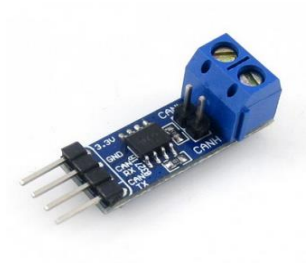
This chapter focuses exclusively on the steps that are specific to the NVIDIA Jetson Orin Nano. These primarily involve configuring the system to enable socket CAN functionality, which is a prerequisite for using the ros2_canopen package.

8.1 Hardware setup

CAN controller is available on the Jetson Orin Nano but with no CAN driver IC.



An additional CAN physical driver is used. A SN65HVD230 CAN Board (waveshare) can be easily attached. Any other CAN driver should work as well.



https://www.waveshare.com/sn65hvd230-can-board.htm

8.2 Software Setup

8.2.1 Flash Jetpac 6.x with Ubuntu 22.04

Flash the Jetpac with Ubuntu 22.04 to the NVIDIA Jetson Orin Nano with its carrier board.

Follow the Steps of the installation guide in the link:
https://wiki.seeedstudio.com/reComputer_J4012_Flash_Jetpack/

8.2.2 SocketCAN installation

To use SocketCAN on an Nvidia Jetson Orin Nano with Ubuntu 22.04, follow these steps:

1. Install Required Packages: Open a terminal and install the necessary packages:

   ```
   sudo apt-get update
   sudo apt-get install can-utils
   ```

2. Load Kernel Modules: Load the CAN kernel modules:

   ```
   sudo modprobe can
   sudo modprobe can_raw
   sudo modprobe mttcan
   ```

3. Configure CAN Interface: Identify your CAN interface (e.g., can0). You can use ip link to list network interfaces. Then, bring up the CAN interface:

   ```
   sudo ip link set can0 up type can bitrate 1000000
   ```

8.2.3 Testing CAN bus

With the command **ifconfig** you should be able to see the CAN interface listed as shown below.

```
ifconfig
```

```
can0: flags=193<UP,RUNNING,NOARP>  mtu 16
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 65536  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Verify CAN Interface and check the status of the CAN interface:

```
ip -details link show can0
```

Test CAN communication with the commands **candump** and **cansend**:

In the first terminal:

```
candump can0
```

In the second terminal:

```
cansend can0 000#11.22.33.44
```

8.2.4 Make it run on boot

For bringing up the can0 interface on boot, use a systemd service.

Create a file "can_networks.service" in /etc/systemd/system/

```
sudo gedit /etc/systemd/system/ can_networks.service
```

The content of the file is:

```
***
[Unit]
Description=Set up CAN interface
After=network.target

[Service]
Type=oneshot
RemainAfterExit=yes
#ExeStart=/usr/local
ExecStart=/sbin/ip link set can0 up type can bitrate 1000000
ExecStart=/sbin/ifconfig can0 txqueuelen 65536
ExecStop=/sbin/ip link set can0 down

[Install]
WantedBy=multi-user.target
#WantedBy=default.target
***
```

Activate the systemd service:

```
sudo systemctl daemon-reload
```

```
sudo systemctl enable can_networks.service
```

```
sudo systemctl start can_networks.service
```

9 Practical case with Raspberry Pi 5 and CAN-HAT

This chapter focuses exclusively on the steps that are specific to the Raspberry Pi. These primarily involve configuring the system to enable socket CAN functionality, which is a prerequisite for using the ros2_canopen package.

Raspberry Pi 5 allows you to install a Linux Ubuntu 24.04 Noble. Therefore, the ROS2 version “Jazzy” must be use.

9.1 Hardware Setup

Because the Raspberry Pi has no built-in CAN controller, we must provide an external CAN controller and driver.

For our tests we used a 2-CH_CAN_HAT+ with dual CAN interfaces. So, there is CAN_1 and CAN_2. This board holds two Microchip CAN controllers of type MCP2515. The communication to the Raspberry Pi 5 is done via SPI.

2-CH_CAN_HAT+

https://www.waveshare.com/wiki/2-CH_CAN_HAT+

Any other CAN HAT should work, as well.

9.2 Software Setup

9.2.1 Install Ubuntu 24.04 on Raspberry Pi 5

That ROS2 can be used on a Raspberry Pi 5 you must install Ubuntu 24.04 first.

You can use the **Raspberry Pi Imager** Tool. It can be found here:
https://www.raspberrypi.com/software/

Once you have opened the **Raspberry Pi Imager** you can use the button “Choose Device” to select the Raspberry Pi 5. Next click “Choose OS” -> “Other general-purpose OS” -> “Ubuntu” -> “24.04 LTS 64bit”.

9.2.2 CAN HAT driver installation

Install BCM2835, open the Raspberry Pi terminal, and run the following commands:

```
wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.60.tar.gz
tar zxvf bcm2835-1.60.tar.gz
cd bcm2835-1.60/
sudo ./configure
sudo make
sudo make check
sudo make install
```

Follow the steps for a 64-bit Raspberry Pi System
Copy the resource package to the Raspberry Pi using the command:

```
wget https://files.waveshare.com/upload/8/8c/WiringPi-master.zip
sudo apt-get install unzip
unzip WiringPi-master.zip
cd WiringPi-master/
sudo ./build
chmod +x ./build
```

Install a few python3 dependencies:

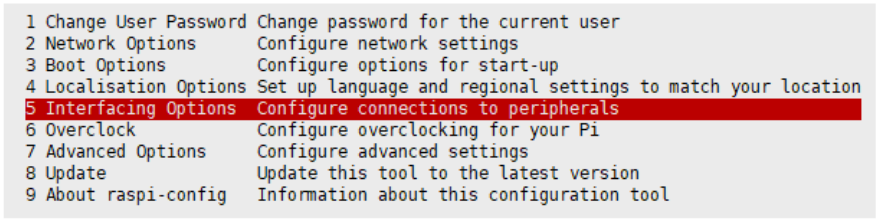
```
sudo apt-get update
sudo apt-get install python3-pip
sudo apt-get install python3-pil
sudo apt-get install python3-numpy
sudo pip3 install RPi.GPIO
sudo pip3 install spidev
sudo pip3 install python-can
```

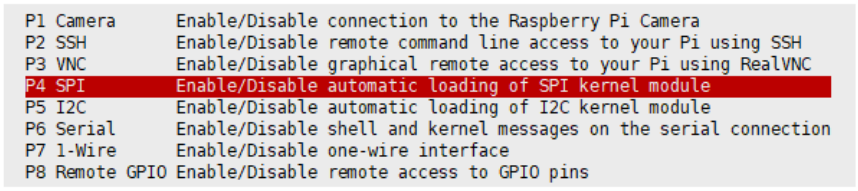
9.2.3 Setting up the SPI interface

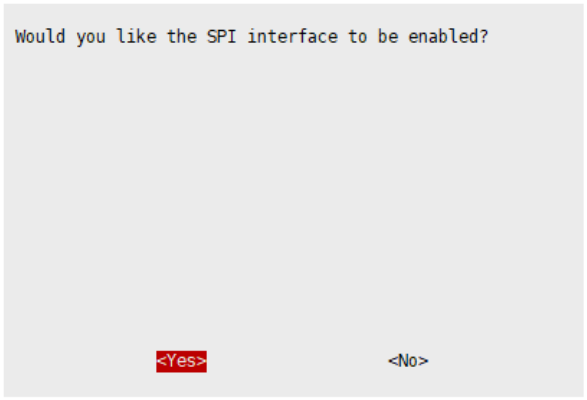
Next step is to enable the SPI interface on the Raspberry Pi 5. Open the Raspberry Pi terminal, and input the following commands to enter the configure interface:

```
(sudo apt install raspi-config)
sudo raspi-config
```

Select “5 Interfacing Options” -> “P4 SPI” -> “<Yes>” to enable SPI interface
Then leave with <Finish>








```
sudo reboot
sudo nano /boot/firmware/config.txt
```

Add these statements in the config.txt file at the last line (below[ALL]):

```
dtparam=spi=on
dtoverlay=i2c0
dtoverlay=spi1-3cs
dtoverlay=mcp2515,spi1-1,oscillator=16000000,interrupt=22
dtoverlay=mcp2515,spi1-2,oscillator=16000000,interrupt=13
```

Perform a reboot again:

```
sudo reboot
```

9.2.4 Testing the SPI interface

After rebooting the Raspberry Pi, we check the SPI information:

```
sudo dmesg | grep spi1
```

```
mk@mk-pi:~$ sudo dmesg | grep spi1
[sudo] password for mk:
[    4.425678] mcp251x spi1.2 can0: MCP2515 successfully initialized.
[    4.445421] mcp251x spi1.1 can1: MCP2515 successfully initialized.
mk@mk-pi:~$
```

9.2.5 Enable the CAN bus (Socket CAN)

The CAN interface ie enabled with the following commands

```
sudo ip link set can0 up type can bitrate 1000000
```

```
sudo ip link set can1 up type can bitrate 1000000
```

```
sudo ifconfig can0 txqueuelen 65536
```

```
sudo ifconfig can1 txqueuelen 65536
```

9.2.6 Testing the CAN bus

With the command **ifconfig** you should be able to see the CAN interface(s) listed as shown below.

```
Ifconfig
```

```
can0: flags=193<UP,RUNNING,NOARP>  mtu 16
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 65536  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

can1: flags=193<UP,RUNNING,NOARP>  mtu 16
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 65536  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Before we can test the communication on the CAN bus, we must install the **can-utils** with**:**

```
sudo apt-get install can-utils
```

In case you use also a 2 channel CAN HAT you can connect the CAN_H of CAN0 to CAN1 and CAN_L of CAN0 to CAN1. Otherwise make sure you have any other CAN device attached with the same bitrate. You can then use candump and cansend commands on the same CAN interface (e.g. can0) or on can0 and can1.

In one Terminal call:

```
candump can0
```

In another terminal call:

```
cansend can1 000#11.22.33.44
```

Test the interface with more details:

```
ip -details link show can0
ip -details link show can1
```

9.2.7 Make it run on boot

For bringing up the can0 interface on boot, use a systemd service.

Create a file "can_networks.service" in /etc/systemd/system/

```
sudo gedit /etc/systemd/system/ can_networks.service
```

The content of the file is:

***
[Unit]
Description=Set up CAN interface
After=network.target

[Service]

```
Type=oneshot
RemainAfterExit=yes

#ExeStart=/usr/local
ExecStart=/sbin/ip link set can0 up type can bitrate 1000000
ExecStart=/sbin/ifconfig can0 txqueuelen 65536
ExecStart=/sbin/ip link set can1 up type can bitrate 1000000
ExecStart=/sbin/ifconfig can1 txqueuelen 65536
ExecStop=/sbin/ip link set can0 down
ExecStop=/sbin/ip link set can1 down

[Install]
WantedBy=multi-user.target
#WantedBy=default.target
***
```

Activate the systemd service:

```
sudo systemctl daemon-reload

sudo systemctl enable can_networks.service

sudo systemctl start can_networks.service
```

10 Troubleshooting

Sometimes things do not work as expected. In such a case it is important that you can check some settings and find out where something goes wrong.

10.1 Useful tools and commands

10.1.1 Linux command

Shows some detailed information about the interface “can0”

```
ip -details link show can0
```

10.1.2 ROS2 tools

rqt_graph
rqt_plot

10.1.3 ROS2 commands

ros2 topic list
ros2 service list

ros2 topic echo /{topic_name}

10.1.4 CAN tools

Install can-utils and monitor the CAN bus with cansend and candump

Monitor all bus taffic:

```
candump can0
```

Send an SDO read request, reading object 0x6064/0 on node_id 1 (COB-Id =0x601):

```
cansend can0 601#40.64.60.00.00.00.00.00
```

The startup sequence on the CAN bus when starting (bus_config_cia402_epos4_pos) looks like this:

```
mk@mk-pi:~/ros2_ws$ candump can0
  can0  701   [1]  05
  can0  701   [1]  05
  can0  701   [1]  05
  can0  701   [1]  05
  can0  706   [1]  00
  can0  000   [2]  82 00
  can0  701   [1]  00
  can0  601   [8]  40 00 10 00 00 00 00 00
  can0  581   [8]  43 00 10 00 92 01 02 00
  can0  601   [8]  40 18 10 01 00 00 00 00
  can0  581   [8]  43 18 10 01 FB 00 00 00
  can0  601   [8]  40 18 10 02 00 00 00 00
  can0  581   [8]  43 18 10 02 00 00 50 60
  can0  601   [8]  40 18 10 03 00 00 00 00
  can0  581   [8]  43 18 10 03 00 00 70 01
  can0  601   [8]  2B 17 10 00 D0 07 00 00
  can0  701   [1]  7F
  can0  581   [8]  60 17 10 00 00 00 00 00
  can0  601   [8]  23 00 14 01 01 02 00 80
  can0  581   [8]  60 00 14 01 00 00 00 00
  can0  601   [8]  2F 00 14 02 01 00 00 00
  can0  581   [8]  60 00 14 02 00 00 00 00
  can0  601   [8]  2F 00 16 00 00 00 00 00
  can0  581   [8]  60 00 16 00 00 00 00 00
  can0  601   [8]  23 00 16 01 10 00 40 60
  can0  581   [8]  60 00 16 01 00 00 00 00
  can0  601   [8]  23 00 16 02 20 00 7A 60
  can0  581   [8]  60 00 16 02 00 00 00 00
  can0  601   [8]  23 00 16 03 08 00 60 60
  can0  581   [8]  60 00 16 03 00 00 00 00
  can0  601   [8]  2F 00 16 00 03 00 00 00
  can0  581   [8]  60 00 16 00 00 00 00 00
  can0  601   [8]  23 00 14 01 01 02 00 00
  can0  581   [8]  60 00 14 01 00 00 00 00
  can0  601   [8]  23 00 18 01 81 01 00 80
  can0  581   [8]  60 00 18 01 00 00 00 00
  can0  601   [8]  2F 00 18 02 01 00 00 00
  can0  581   [8]  60 00 18 02 00 00 00 00
  can0  601   [8]  2F 00 1A 00 00 00 00 00
  can0  581   [8]  60 00 1A 00 00 00 00 00
  can0  601   [8]  23 00 1A 01 10 00 41 60
  can0  581   [8]  60 00 1A 01 00 00 00 00
  can0  601   [8]  23 00 1A 02 20 00 64 60
  can0  581   [8]  60 00 1A 02 00 00 00 00
  can0  601   [8]  23 00 1A 03 08 00 61 60
  can0  581   [8]  60 00 1A 03 00 00 00 00
  can0  601   [8]  2F 00 1A 00 03 00 00 00
  can0  080   [0]
  can0  581   [8]  60 00 1A 00 00 00 00 00
  can0  201   [7]  00 00 00 00 00 00 00
  can0  601   [8]  23 00 18 01 81 01 00 00
  can0  581   [8]  60 00 18 01 00 00 00 00
  can0  601   [8]  2B 40 60 00 80 00 00 00
  can0  581   [8]  60 40 60 00 00 00 00 00
  can0  601   [8]  2B 40 60 00 06 00 00 00
  can0  581   [8]  60 40 60 00 00 00 00 00
  can0  601   [8]  2B 40 60 00 07 00 00 00
  can0  581   [8]  60 40 60 00 00 00 00 00
  can0  601   [8]  2F C2 60 01 32 00 00 00
  can0  581   [8]  60 C2 60 01 00 00 00 00
  can0  601   [8]  2F C2 60 02 FD 00 00 00
  can0  581   [8]  60 C2 60 02 00 00 00 00
  can0  601   [8]  23 81 60 00 E8 03 00 00
  can0  581   [8]  60 81 60 00 00 00 00 00
  can0  601   [8]  23 83 60 00 D0 07 00 00
  can0  581   [8]  60 83 60 00 00 00 00 00
  can0  601   [8]  2F 98 60 00 25 00 00 00
  can0  581   [8]  60 98 60 00 00 00 00 00
  can0  601   [8]  2F 60 60 00 01 00 00 00
  can0  581   [8]  60 60 60 00 00 00 00 00
  can0  080   [0]
  can0  181   [7]  40 02 0E EE FF FF 01
  can0  201   [7]  00 00 00 00 00 00 00
  can0  080   [0]
  can0  181   [7]  40 02 0E EE FF FF 01
  can0  201   [7]  00 00 00 00 00 00 00
  can0  080   [0]
  can0  181   [7]  40 02 0E EE FF FF 01
  can0  201   [7]  00 00 00 00 00 00 00
  can0  080   [0]
  can0  181   [7]  40 02 0E EE FF FF 01
  can0  201   [7]  00 00 00 00 00 00 00
  can0  080   [0]
  can0  181   [7]  40 02 0E EE FF FF 01
  can0  201   [7]  00 00 00 00 00 00 00
  can0  080   [0]
  can0  181   [7]  40 02 0E EE FF FF 01
  can0  201   [7]  00 00 00 00 00 00 00
```

10.2 Troubleshooting ros2_canopen

10.2.1 Configuration mismatch

ros2_canopen is very strict when it comes to connecting real devices on the CAN bus. If the device or some settings in the given .eds file do not match the connected device, it can lead to a failing system.
During initialization phase the master reads back objects from the device.

To check what is read back from the device on the CAN bus, you can use the **candump** command. Note that you have a SDO request on the COB-Id 0x600+node_id and the response on the COB-Id 0x580+node_id.

10.2.2 Colcon build

Sometimes it can be helpful to double check what the output files are of the colcon build or the master.dcf generated by “cogen_dcf”.
Navigate to the path ros2_ws/
Path to the master.dcf and master.bin is:

~/ros2_ws/build/maxon_epos4_ros2/config/bus_config_cia402_epos4_pos

10.3 Links

http://epos.maxongroup.com/

https://www.can-cia.org/can-knowledge/canopen

https://www.can-cia.org/can-knowledge/pdo-protocol