

# Aurora link with OPAL-RT via SFP

PN110 | Posted on February 2, 2026 | Updated on May 6, 2026



**François LEDENT**

Development Engineer

imperix • in

---

## Table of Contents

- [Case study](#)
- [Required software](#)
- [Downloads](#)
- [OP4510 project](#)
  - [Simulink model](#)
  - [Two SFP interfaces](#)
  - [Import in RT-LAB](#)
- [Communication chain](#)
  - [Overview](#)
  - [Vivado project](#)
  - [Modules/IPs description](#)
  - [Aurora parameters](#)
- [Experimental validation](#)
  - [Physical setup](#)
  - [Software-side setup](#)
  - [Real-time monitoring](#)

This page presents a practical example of Aurora communication with OPAL-RT simulators, specifically the OP4510 and OP4512. It provides a ready-to-use user application, along with an FPGA design specifically configured for interfacing OPAL-RT simulators.

To demonstrate the setup, this page also includes a pre-configured and fully functional RT-LAB project for the OPAL-RT simulator implementing a simple loopback example. In this example, a three-phase sine wave is transmitted from the imperix controller, multiplied by two in the simulator, and sent back to the controller.

To cover a broader range of use cases, the RT-LAB project deployed on the OPAL-RT target instantiates two SFP interfaces. The first interface is managed by [eHS](#), OPAL-RT's

FPGA-based electrical toolbox and solver, while the second interface is controlled by the simulator's CPU.

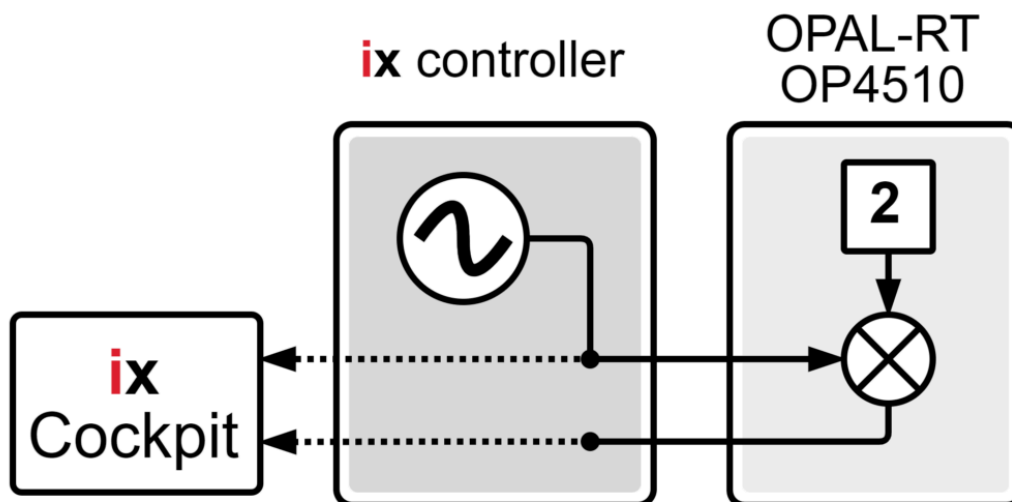
For broader technical background, a general introduction to [SFP communication with third-party devices](#) is available. That page covers fundamental SFP considerations, describes the full communication chain, and provides guidance on implementing the imperix-side drivers.

Similar pages are available for Plexim and Typhoon devices in [Aurora link with Plexim via SFP](#) and [Aurora link with Typhoon via SFP](#).

## Case study

This case study demonstrates a straightforward signal processing loop:

- The controller generates a three-phase sine wave and transmits it to an OPAL-RT OP4510 via SFP.
- Upon receipt, the OP4510 applies a gain of two to the three signals – doubling the amplitude of the sine wave – and returns them to the controller, also via SFP.
- Finally, the original transmitted values and the received return signals are compared in real-time in Cockpit, showing the proper operation of the system.



## Required software

- **Vivado Design Suite** (version **2022.1** is recommended)  
The [Xilinx installation page](#) details the installation procedure.
- **FPGA sandbox template 3.10** or later.  
Available on the [FPGA download](#) page.
- **C++ or ACG SDK version 2024.3** or later.  
Available on the [SDK download](#) page.

On the third-party side, this project has been tested with an **OP4510, RT-LAB 2024.1.6.55** and **Matlab 2022B**.

## Downloads

As explained in the [setup overview](#), the SFP communication requires the three following software parts:

- The **user application**, running in the imperix controller's CPU and provided as a Simulink or PLECS script.
- The **FPGA bitstream**, loaded in the imperix controller's FPGA and provided as generation scripts.
- The **third-party application**, running in the OP4510 and provided as an RT-LAB archive.

User application	FPGA bitstream	OP4510 application
<a href="#">aurora_ix_template.slx</a> <a href="#">aurora_ix_template.plecs</a>	<a href="#">aurora_ix_opalrt_gen_scripts.zip</a>	<a href="#">aurora_ix_opalrt_rtlab.zip</a>

Further details on Aurora communication for the OP4510 and OP4512 from OPAL-RT are provided below.

[Download OPAL\\_RT\\_Aurora\\_Documentation.pdf](#)

## OP4510 project

The OP4510 project is provided as a ZIP archive that can be imported in RT-Lab, which contains the OP4510's model, firmware, eHS model and configuration files.

## Simulink model

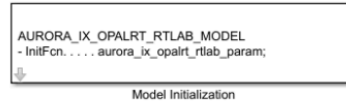
The Simulink model contains the two main subsystems:

- **SC\_Console** (left) to control and monitor signals in real-time.
- **SM\_Controller** (right) that contains the OP4510 control program.

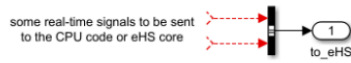


## Aurora link with OPAL-RT over SFP

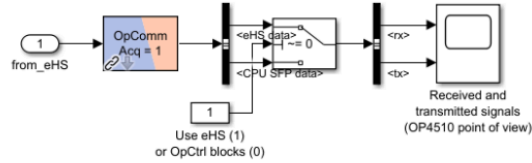
Basic loopback example between an imperix controller and an OP4510 from OPAL-RT



### Inputs



### Outputs

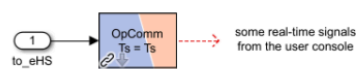
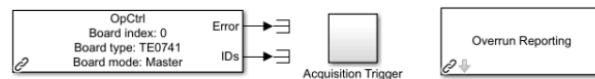
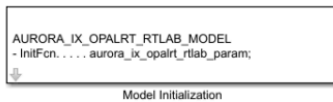


## SC\_Console subsystem in the OP4510's Simulink model

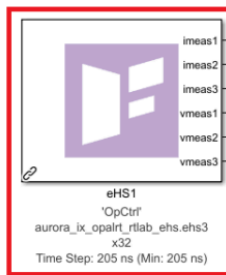


## Aurora link with OPAL-RT over SFP

Basic loopback example between an imperix controller and an OP4510 from OPAL-RT



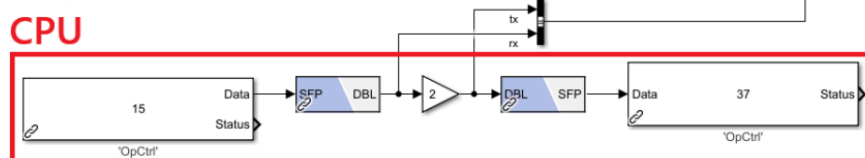
### eHS



#### Note

- eHS is connected to SFP00 and handles the Aurora communication at the eHS rate
- OpCtrl blocks are connected to SFP02 and handle the Aurora communication at the CPU rate

### CPU



## SM\_Controller subsystem in the OP4510's Simulink model

The SM\_Controller subsystem contains two SFP interfaces to access SFP either from eHS (on port SFP CH00) or from the simulator's CPU (on port SFP CH02). These interfaces are described in the subsequent sections.

Real-time variables can be defined in the SC\_Console subsystem as constants and passed to the control logic of the SM\_Controller subsystem (see the *to\_eHS* ports in the screenshots above). Although this example doesn't use that mechanism, the related ports and blocks are kept for easier future expansion of the model.

## Two SFP interfaces

To support different usage scenarios, this project provides two simultaneous SFP interfaces, enabling communication either with [eHS](#) – the FPGA-based electrical toolbox and solver from OPAL-RT – or with the simulator’s CPU. The characteristics of each interface are summarized in the table below.

Additional details on using both interfaces can be found in the OPAL-RT documentation available in the [Downloads](#) section.

Interface name	Location	Timestep	Use case(s)	SFP port
eHS	FPGA	A few FPGA cycles (depends on the simulated circuit)	Simulation of power circuits only (in <a href="#">eHS</a> )	SFP CH00
CPU	CPU	CPU period	Simulation of power circuits or any other use (in Simulink)	SFP CH02

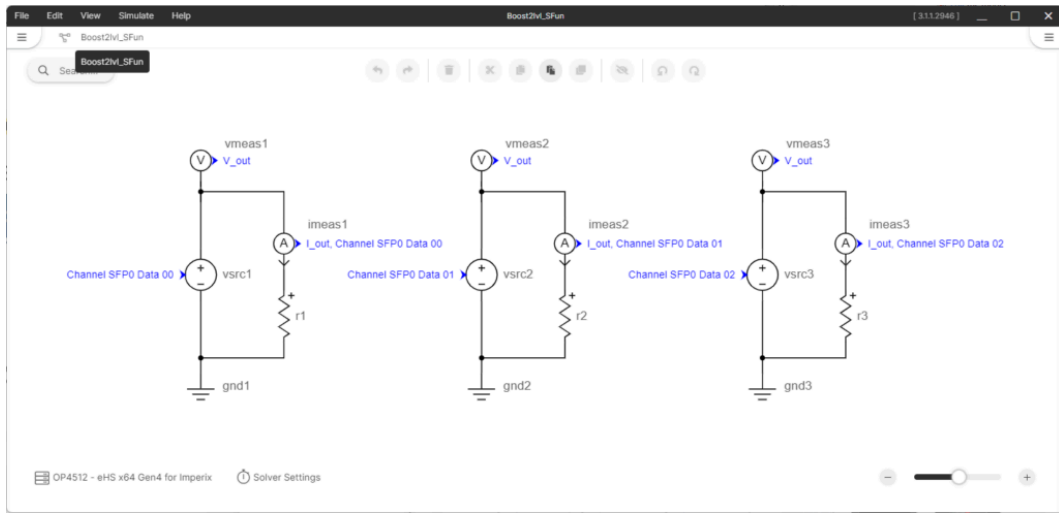
From the imperix controller perspective, the choice of interface has no impact. In both cases, the values are expected to be doubled in the OP4510.

Both communication interfaces run simultaneously. In this example, the imperix controller has only one port dedicated to the Aurora communication with the OP4510, meaning that one of the two OP4510’s ports should be left unconnected and simply returns zeroes as incoming values.

**eHS interface**

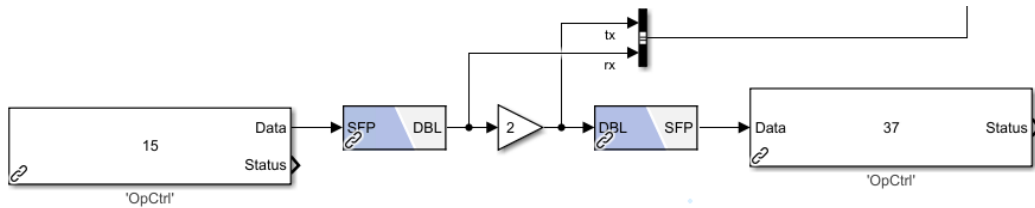
The eHS circuit is designed from the [Schematic Editor](#) of OPAL-RT and directly embedded in the Simulink model, as shown in the right capture of the [Simulink model](#) section above. Double-click on the **eHS ‘OpCtrl’** block to open the eHS circuit in the Schematic Editor.

In the eHS circuit, controlled voltage sources are fed by the incoming SFP values and placed in series with 0.5Ω resistors. The current flowing through the resistors is sent back to the imperix controller.



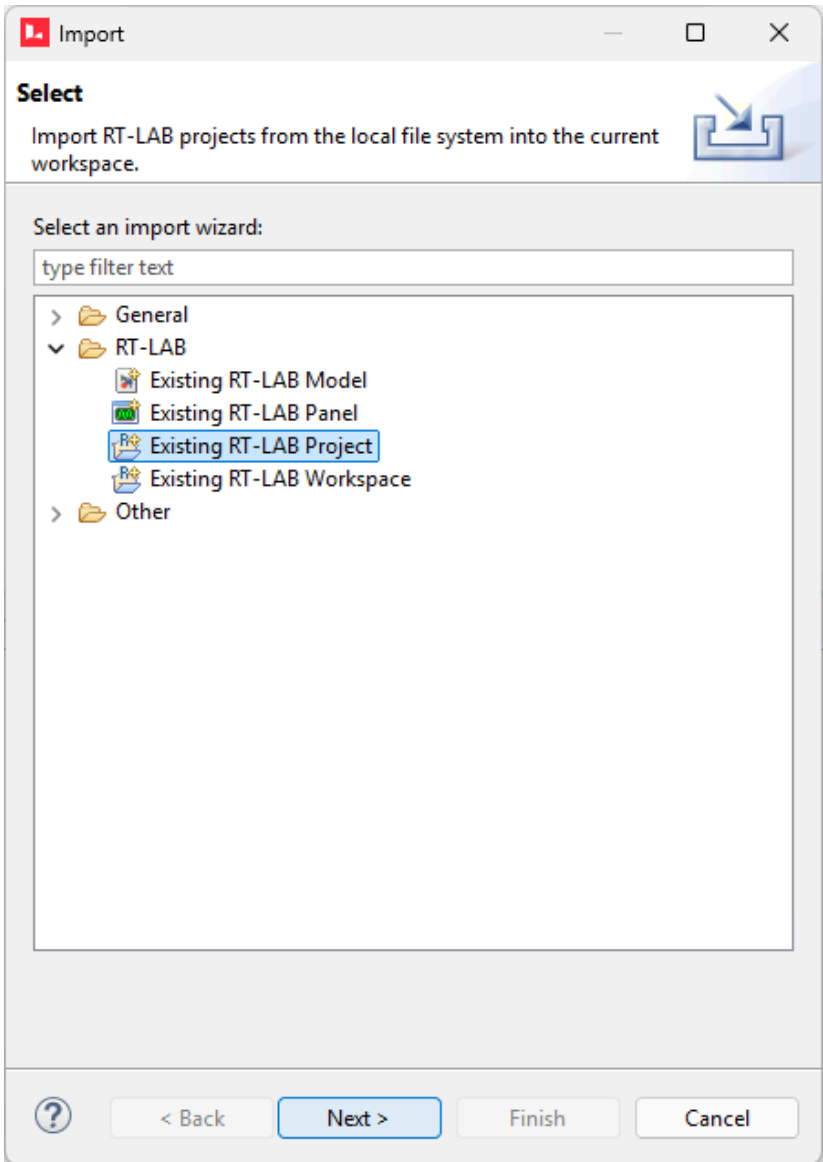
## CPU interface

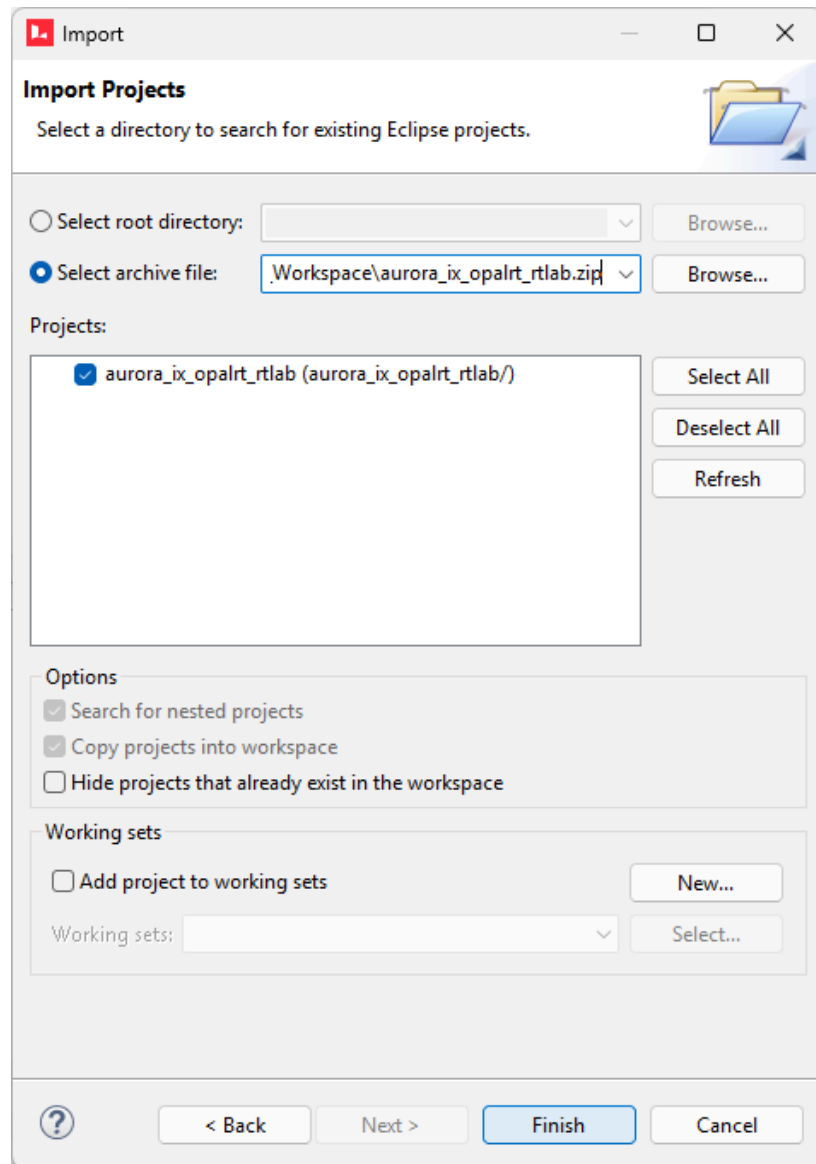
For the CPU-based SFP control method, the incoming values are simply manually multiplied by two. As the 'DataOut Recv' and 'DataIn Send' blocks manipulate the data as unsigned integers, a datatype conversion is applied upon reception and before transmission in the OP4510.



## Import in RT-LAB

To import the project, click on **File > Import**. Then, in the Import window, expand **RT-LAB**, select **Existing RT-LAB Project** and click **Next**. In the next window, select **Select archive file** and navigate to the location where the archive is locally stored. Finally, press **Finish**.





Once the model is imported, double-click on it in the **Project Explorer** to open it. Navigate to **Models** and double-click on the model. Then, in the main panel, successively click on **Build**, **Load** and **Execute**.

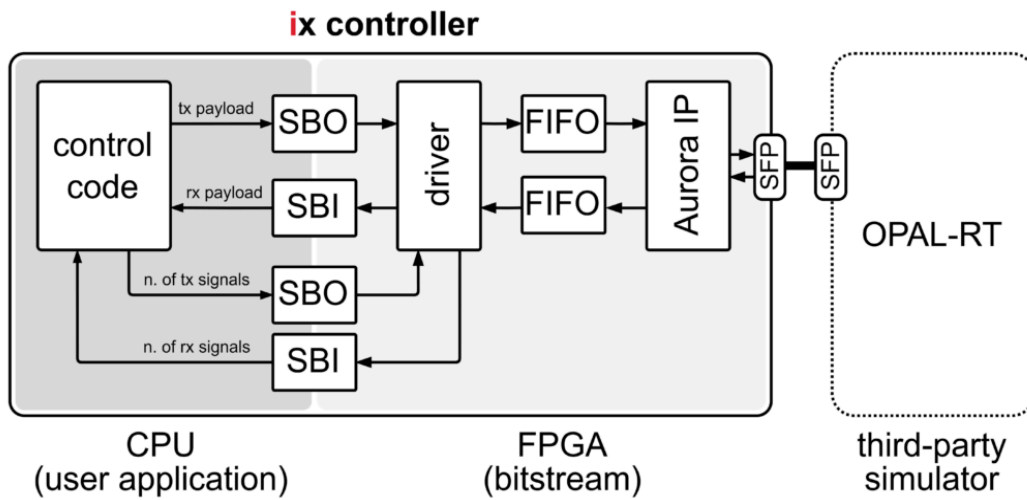
During the loading phase, a new Simulink model is automatically created based on the SC\_Console subsystem. This model is intended to be used when the code is running to control the constants and monitor data in real-time.

Open the scope when the code is running to observe the data exchanged between the imperix controller and the OP4510. Change the monitored interface (eHS or CPU-based) by setting the dedicated constant in the console.

## Communication chain

### Overview

The overview of the communication chain is presented in the [setup overview](#).

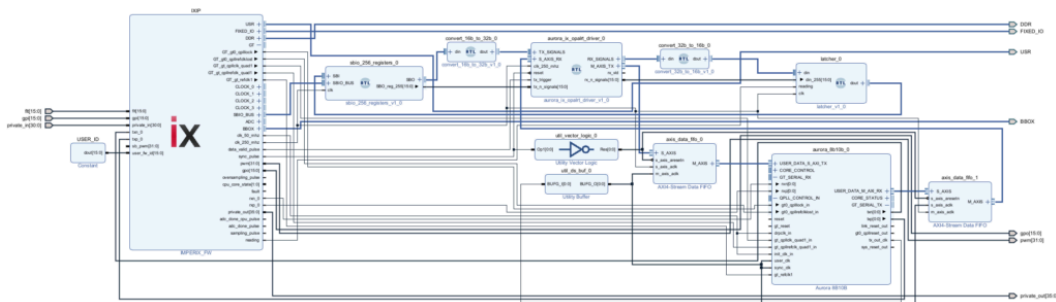


The main clock domain always runs at 250 MHz, while the frequency of the Aurora clock domain varies with the configuration of the Aurora IP. In this example, the frequency is 125 MHz with the configuration presented in the [Aurora parameters](#) section.

## Vivado project

The Vivado project is provided in the form of generation scripts. As explained in the [PN109](#), the scripts automatically create and open the project illustrated below. The bitstream can be directly generated by simply pressing **Generate Bitstream** in the left navigation bar in the Vivado environment.

As provided in the source files, the driver supports the exchange of up to **50 signals** in each direction and the Aurora communication is linked to the **SFP 0 (UP)** port of the imperix controller.



Once generated, the bitstream can be loaded onto the imperix controller using [Cockpit](#). A reboot is required for the bitstream change to take effect.

To increase the number of exchanged signals or change the SFP port used for the Aurora communication, please refer to [How to exchange more signals](#) and [How to assign a different SFP port](#).

## Modules/IPs description

The Vivado project contains the following VHDL modules and IPs.

Module name	Type	Description
sbio_256_registers	VHDL module	Instantiates and provides access to SBIO bus registers in the FPGA. More information is provided <a href="#">here</a> .
convert_16b_to_32b	VHDL module	Converts the 16-bit words of the SBIO bus back into the 32-bit words of the payload.
aurora_ix_opalrt_driver	VHDL module	Custom driver provided by imperix to communicate with the OP4510 from OPAL-RT ; mainly acts as a parallel-to-serial transmitter and serial-to-parallel receiver.
convert_32b_to_16b	VHDL module	Converts the 32-bit words received from the OP4510 through the driver into 16-bit words compatible with the SBIO bus.
latcher	VHDL module	Ensures data coherency by preventing the update of the SBI registers while the CPU is reading.
AXI4-Stream Data FIFO	Vivado IP (Xilinx)	Handles the clock domain crossing between the main 250 MHz domain of the imperix firmware and the Aurora clock domain ; buffers the frame in the transmission direction.
Aurora 8B10B	Vivado IP (Xilinx)	Handles the Aurora communication and interfaces with the underlying hardware logic.
Utility Vector Logic	Vivado IP (Xilinx)	Converts the active-high reset signal from the sync_pulse into an active-low reset signal for the FIFOs.

## Aurora parameters

The included Aurora IP comes with the correct configuration pre-applied for interfacing with OPAL-RT simulators. When creating a project from scratch, the Aurora IP must be configured in Vivado with the specific parameters listed below.

<b>Protocol</b>	Aurora 8B10B	<b>Interface</b>	Framing
-----------------	--------------	------------------	---------

<b>Line Width (Bytes)</b>	4	<b>Flow Control</b>	None
<b>Line Rate (Gbps)</b>	5	<b>Little Endian Support</b>	Yes
<b>Dataflow Mode</b>	Duplex	<b>CRC</b>	Yes

## Experimental validation

### Physical setup

The physical setup is straightforward:

1. Connect both devices to the network, so that they can be configured and monitored from the PC.
2. Connect the imperix controller to the OP4510 with an SFP cable. As provided, this example considers the port **SFP 0 (UP)** on the controller and **SFP CH00** (eHS) or **SFP CH02** (CPU) on the OP4510.
3. Turn on the two devices.

### Software-side setup

To experimentally validate the system:

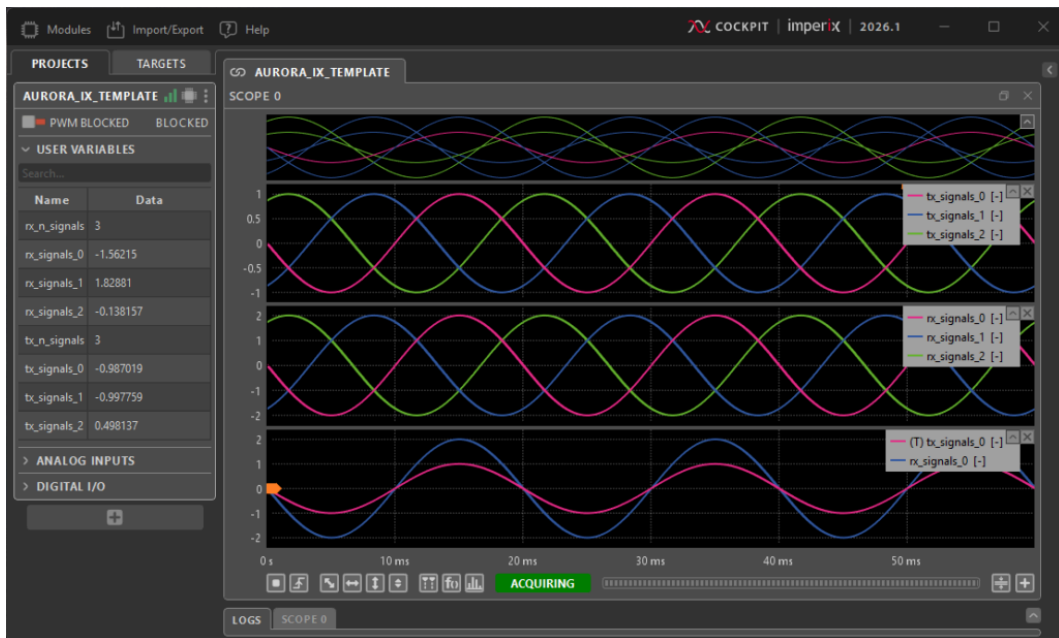
1. Download the three software parts available in the [downloads](#) section.
2. Import the project in RT-LAB. Build, load and execute it on the OP4510, as explained in the [Import in RT-LAB](#) section above.
3. [Generate the bitstream](#) for the imperix controller.
4. Load the bitstream on the imperix controller via Cockpit.
5. Build the [user application template](#) and launch it on the imperix controller via Cockpit.
6. Use Cockpit to monitor the exchanged signals.

The whole system should now be running.

### Real-time monitoring

Connect to the imperix controller with Cockpit. Add a [scope](#) in the project (from the **Modules tab** in the top bar) and drag-and-drop the variables of interest.

The exchanged signals can now be monitored in real-time in Cockpit. As expected, the amplitude of the transmitted signals is multiplied by two in the simulator.



The exchanged signals can also be monitored from the OP4510, as described in the [Import in RT-LAB](#) section. If the real-time signals stay at 0, ensure that the constant in the real-time console selects the right interface (1 for eHS, 0 for CPU).

