https://project-panda.eu/

Research Innovation Action

GA # 824256

EUROPEAN COMMISSION
Horizon 2020 | GV-02-2018
Virtual product development and production of
all types of electrified vehicles and components

Deliverable No.	PANDA D1.5	
Deliverable Title	Organization of power interfaces for real testing (final)	
Deliverable Date	2022-05-31	
Deliverable Type	REPORT	
Dissemination level	Public (PU)	
Written By	Mladen DINIC (TY), Igor PINTARIC (TY), Sergio COSTA (TY), Calin HUSAR (SISW), Marius CIOCAN (SISW)	2022-05-12
Checked by	Daniela CHRENKO (UBFC) – WP leader	2022-05-17
Approved by	Philippe DELARUE (ULille) Calin HUSAR (SISW) Alain BOUSCAYROL (ULille) - Coordinator	2022-06-02 2022-05-20 2022-06-02
Status	Final	2022-06-02

Publishable Executive Summary

Leader: Mladen DINIC (TY), Participants: TY, UTCN, ULille, TUV

The objective of PANDA is to provide a disruptive and open-access model organization for an easy interconnection and exchange of models in the development process of EVs, with the goal to help reducing the time to market by 20%, thanks to advanced methods. Indeed, within the W-model process of product development, two testing stages occur, the virtual testing stage which needs offline simulation, and the real testing stage which needs real time simulation. Within the PANDA project, a unified model organization has to be developed for both stages.

Having that in mind, creating the best possible test environment, in both software and hardware sense, is of utmost importance for the final results and goals of the project to be achieved. This document describes the hardware side of the test setup, or, to be more precise – the elements that allow for the usage of the setup to the full extent, by connecting them in a proper way. In addition, it provides a detailed user guide (provided by SISW) on how to establish a cloud connection with the power interfaces to run Simcenter Amesim © co-simulation tests.

The objective of the document is to showcase the final test setup used for real-time tests in the PANDA project, and how to set up the interfacing elements for users interested in applying similar tests to those demonstrated in the project. Methods used to describe the power interfaces present a result of a decade-long experience of TY engineers in making such documents desirable by the customers, with the focus being on the ease of use/readability and graphical representations of the main features.

Contributions:

No	Who	Description	Date
1	TY	D1.5 report creation and outline	2022-05-09
2	TY / ULille / VUB / UTCN	Sections 1-5 & Appendices (from D1.4)	2020-07-06
3	SISW	Section 6	2022-03-30
4	TY	Conclusions	2022-05-12
6	TY	Internal review	2022-05-12

Contents

Figures

Tables

1 Introduction

The objective of PANDA is to provide a disruptive and open-access model organization for an easy interconnection and exchange of models in the development process of EVs, with the goal to help reducing the time to market by 20%, thanks to advanced methods [PANDA 2020]. Indeed, within the W-model process of product development, two testing stages occur, the virtual testing stage which needs offline simulation, and the real testing stage which needs real time simulation. Within the PANDA project, a unified model organisation has to be developed for both stages [PANDA D1.1]. This objective has been achieved using the Energetic Macroscopic Representation (EMR) formalism [Bouscayrol 2012] as model organization tool. All models are developed within the Simcenter Amesim package where an EMR-library has been specifically developed [Husar 2019] [PANDA D4.1]. A cloud has been used to simulate both virtual testing and real testing using HIL simulation method. In power-HIL testing, a real subsystem is interfaced with a real-time simulation through a dedicate interface system [Bouscayrol 2011]. This deliverable is focused on the power interfaces used for real testing in PANDA. It should be noted that an EMR library has already been developed in Typhoon HIL Control Center [Genic 2017].

As a part of the Task 1.3 (Power Interfaces, [M8-M20] – [M31-42]), and to perform the physical/real testing, dedicated testing interfaces have been developed between the hardware subsystems to test. The experience of project partners in real-time simulation of the subsystems using HIL philosophy, as well as some development details (that came from the knowledge models previously created), were used to obtain specific rules for the deduction of the testing interfaces. This report has been updated from the different feedback and reports on HIL testing where the Typhoon HIL interface has been used:

- Stand-alone Hardware-In-the Loop (HIL) testing of a battery for the reference [PANDA 5.1],
- Stand-alone HIL testing of an e-drive for the reference P-HEV [PANDA 5.2],
- Stand-alone HIL testing of an e-subsystem for the reference P-HEV [PANDA 5.3],
- Cloud-based HIL testing of a battery for the reference BEV and P-HEV [PANDA 2.2],
- Cloud-based HIL testing of an e-drive for the reference BEV and P-HEV [PANDA 3.3],

This document describes the conclusions and results reached after a set of requests received (by TY) from partners that desired test equipment on-site. Moreover, the technical descriptions in this document are the result of the aforementioned requests: the so-called Spec document, prepared by TY, with the inputs from VUB, ULille and UTCN, describes the Power Interfaces (teststands) that the partners received from TY. Their final design is adjusted for intended use solely based on the inputs from the partners.

2 PANDA Project hardware specification

The test environment is comprised of three hardware systems, one for the real (physical) testing of the electrical subsystems of the plug-in series parallel Hybrid Electrical Vehicle (P-HEV), one for the e-drive real testing and one for the battery real testing. For these purposes, power hardware in the loop (P-HIL) technology has been chosen and implemented. Using this technology provides with a possibility to test HEV drive stage and control algorithms in controllable and safe environment. With its high level of certainty, this approach provides a good representation of real electrical and mechanical behaviour of the HEV in everyday exploitation conditions.

2.1 HEV E-subsystem real testing

The studied HEV consists of two electrical drives and one internal combustion engine. The front drive subsystem consists of a hybrid system including an electrical drive and a combustion engine. The rear drive subsystem is a pure electrical drive (Figure 1).

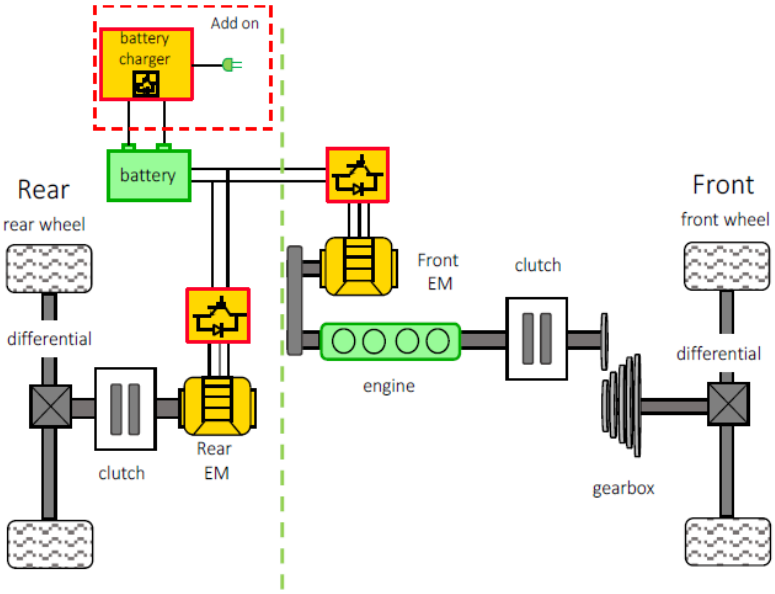


Figure 1: HEV structure

In the PANDA project, the behavior of HEV mechanical rotational subsystems (wheels, internal combustion engine, gearbox…) is emulated by electrical machines with their power converter and control. Typhoon HIL real-time simulation platform is used to control the emulation interface and emulated HEV control unit (ECU). The HEV electrical drive is controlled by ECU that is implemented on the dSPACE platform (Figure 2).

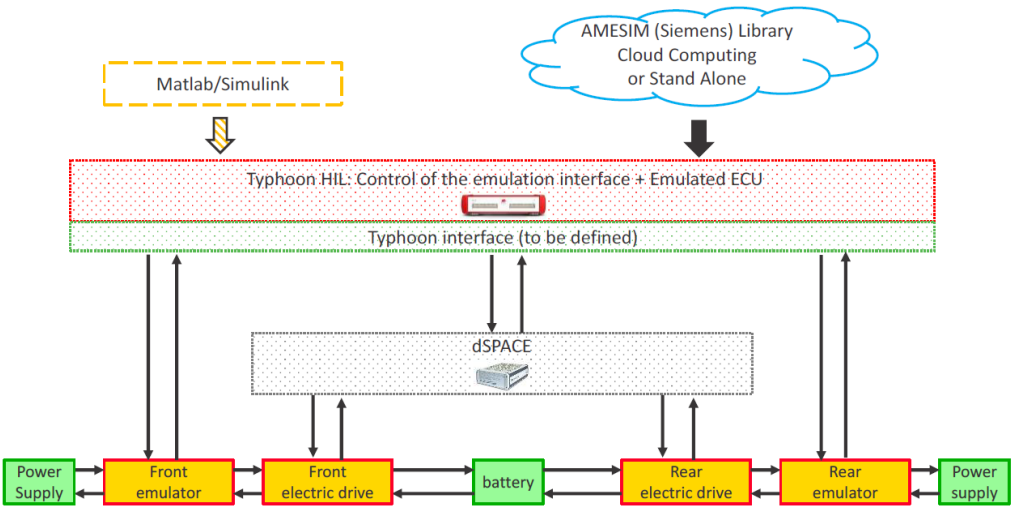


Figure 2: HEV PHIL Representation

The electrical drives (machines and inverters) are provided by Valeo (VEEM). These are three-phase inverters that demand 6 PWM signals each and provide 6 analog measurement signals (machine current and voltage measurements). There are also sensors for torque and speed of the machines. TY provides control signals for emulation interface (PWM-s) and reads the measurement feedback. dSPACE is used for controlling the electrical drives, and to implement the algorithms for the HEV control.

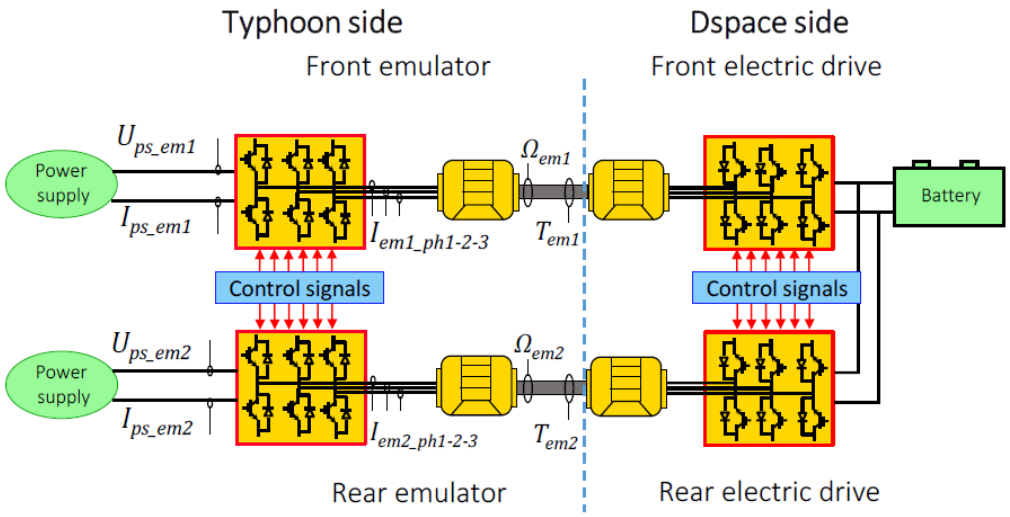


Figure 3: Emulation interface and electric drives

An electric drive has also been tested at UTCN, but the global testing scheme is the same as presented before (with only one drive to test). Therefore, no specific description for that real testing is provided. Indeed, a part of the presented interface will be used in that second case.

2.2 Battery real testing

The battery pack is one of the most important parts of electric vehicle, as it provides the energy for the e-drive. It should be capable to handle various stress conditions in terms of temperature, current, decreased voltage etc. It is essential to verify the battery capability to deal with such stress conditions.

The HEV Battery pack consists of 2 modules of 15 cells each, with a nominal terminal voltage of 48 V. During the process of charging/discharging of the battery pack, it is essential to monitor and regulate state of charge (SoC) and temperature of each cell/module. In multi-cell battery packages, where multiple cells are connected in parallel and series, there is a possibility of disbalanced charging/discharging of the cells, because of the variations in cell characteristics in terms of internal resistance and capacity. If not regulated, modules with multiple batteries tied in parallel can suffer from unequal current sharing between multiple cells. Also, in battery packs with multiple cells or modules tied in series, different voltage of modules can appear if there is no good regulation of module voltages.

Regions of increased temperature – the so-called hot spots – can lead to hazardous situations. When the temperature of the lithium battery cell increases, leakage current of the cell also increases. This can lead to phenomena of thermal runaway that is actually a positive feedback loop between cell current and temperature, which can lead to destruction of the cell and even start a fire.

Charging and discharging the lithium batteries at low temperatures can result in battery damage in the sense of lithium metallization on battery electrodes, which increases the battery parasitic resistance, and thus decreases its usable capacity [Shuai 2018].

Therefore, it is essential to monitor the temperature of the battery pack at multiple locations. While testing a HEV battery, temperatures of each cell and module are monitored. This means that there are 17 temperature measurements. K-type thermocouple temperature sensors are used.

A Cinergia two-quadrant DC source is used as a charger and load for the battery real testing. It is controlled by Typhoon HIL real-time simulator. Currents, voltages, and temperatures of the battery package are also monitored by Typhoon HIL real-time simulator. References and control signals for Cinergia amplifier are also provided by Typhoon HIL real time simulator shown in Figure 4.

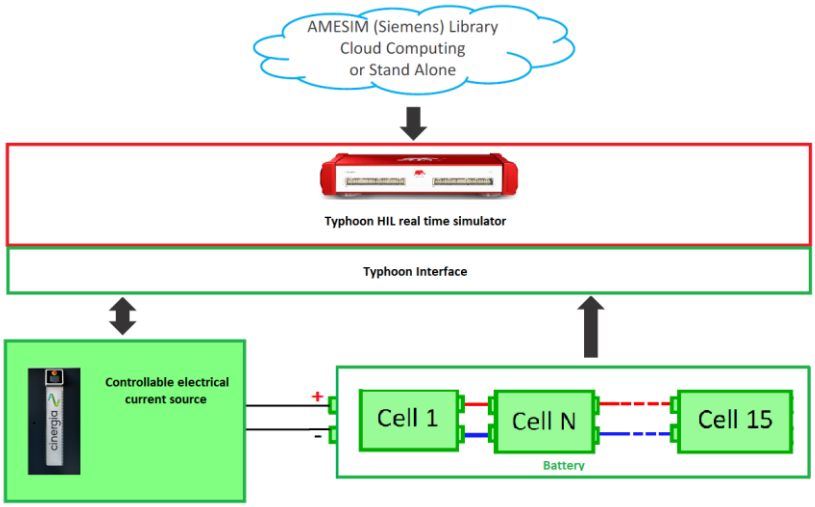


Figure 4: Battery real testing system

3 Typhoon HIL hardware equipment requirements

From previous chapters it can be seen that there are multiple equipment pieces from multiple vendors in the HEV emulation and battery testing system. To build the complete HIL system, it is necessary to connect Typhoon HIL real-time simulator (ECU), a dSPACE control system, machines and inverters from Valeo (VEEM), a Cinergia DC current source, battery pack, and also the Simcenter Amesim Cloud. For this purpose, Typhoon HIL has developed interfacing equipment, the so-called HIL Connects, and real-time simulators for automotive applications.

There are also demands from Universitatea Technica Cluj-Napoca (UTCN) to connect NI PXIe-7856R IO Module Card, and from Vrije Universiteit Brussels (VUB) to connect PEC SBT8050 Battery Life Test System to the Typhoon HIL Cabinet.

The Typhoon HIL System is custom designed, according to user specifications gathered from each partner:

1. University of Lille (ULille) – real testing of the e-subsystems of the P-HEV
2. Universitatea Technica Cluj-Napoca (UTCN) – real testing of an e-drive
3. Vrije Universiteit Brussels (VUB) – real testing of a battery

Detailed user specifications are given in the next section.

3.1 University of Lille (ULille)

For the purposes of tasks 5.3 (E-subsystem real testing) [PANDA D5.3] and 5.1 (battery real testing) [PANDA D5.1] of WP5, ULille has detailed the following requirements:

1. Test setup shall consist of:
    a. Typhoon HIL Cabinet, used for simulation, interfacing and control
    b. Rotational HEV power and drive emulators, controlled by HIL and dSPACE respectively
    c. Cinergia power amplifier, used as a current sink/source for battery testing
    d. Battery pack
    e. dSPACE control systems used as ECU
2. Typhoon HIL Cabinet shall provide connectors to interface with Power emulators, Cinergia, dSPACE boards
3. IO characteristics shall be as given by ULille
4. Proposed connector layout (Figure 5)

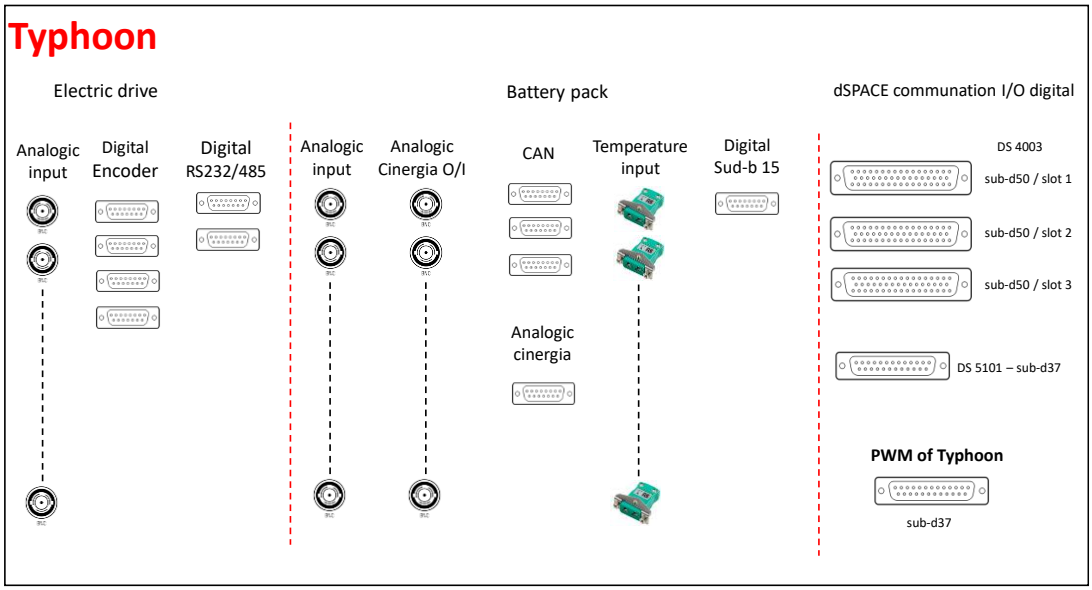


Figure 5: HIL Cabinet connectors, specified by ULille

3.2 Universitatea Technica Cluj-Napoca (UTCN)

UTCN would like to connect to a NI PXIe-7856R IO Module Card. TY assumes all the signals from Figure 6 should be connected to the HIL Cabinet.

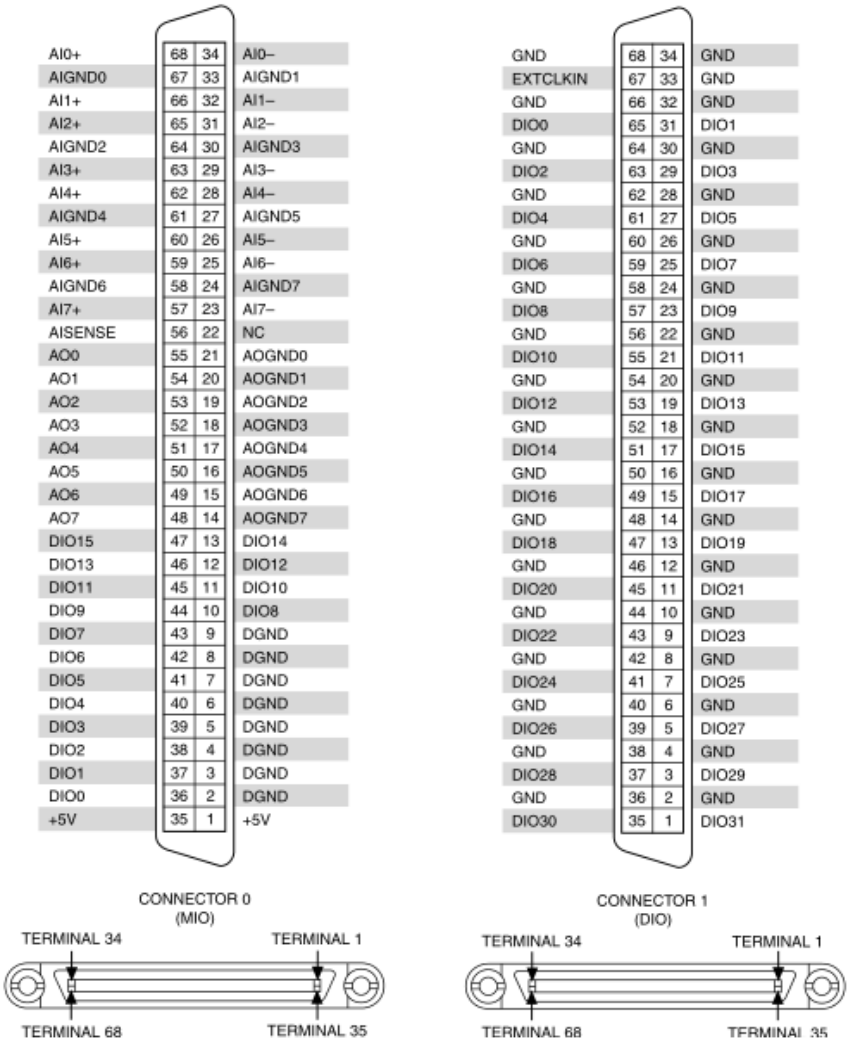


Figure 6: NI PXIe-7856R pinout[1]

The connector is not readily available on the market, but NI offers terminal breakout boards (NI SCB-68 and SCB-68A).

TY assumed that UTCN had these terminal breakouts available and provided BNC and DSUB connectors on the HIL Cabinet.

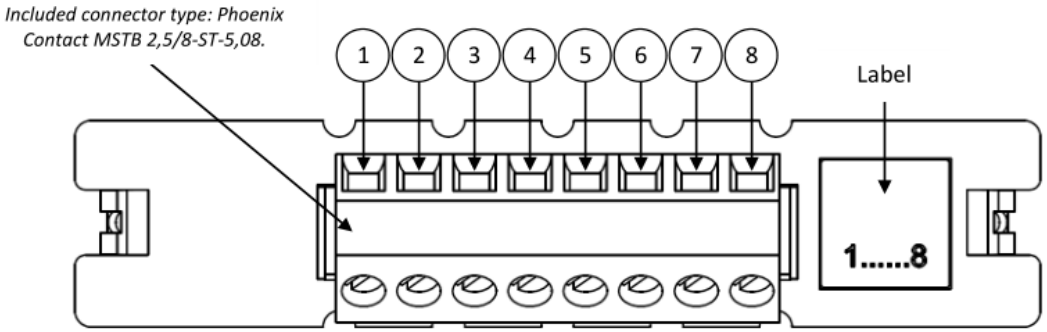
TY also provided cables which break out BNC and DSUB into wires suitable for terminal connection to NI SCB-68 and SCB-68A.

[1] Source: NI PXIe-7856R User Manual

3.3 Vrije Universiteit Brussels (VUB)

VUB would like to connect a PEC SBT8050 Battery Life Test System to the Typhoon HIL Cabinet. The TY Cabinet should have a connector specific to this device.

The device should be controlled via the connector shown in Figure 7.



The mating connector <100452210 / Phoenix Contact / 1826348 - SMSTB 2,5/8-ST-5,08> is included.

No.	Description	Range, …
1	GND	0 V
2	IN1-	**Do not connect anything**
3	IN1+	
4	GND	0 V
5	OUT-	-15 V (for external device: max. 65 mA)
6	IN0-	±10 V
7	IN0+	
8	OUT+	+15 V (for external device: max. 65 mA)

Figure 7: PEC SBT8050 analog input connector[2]

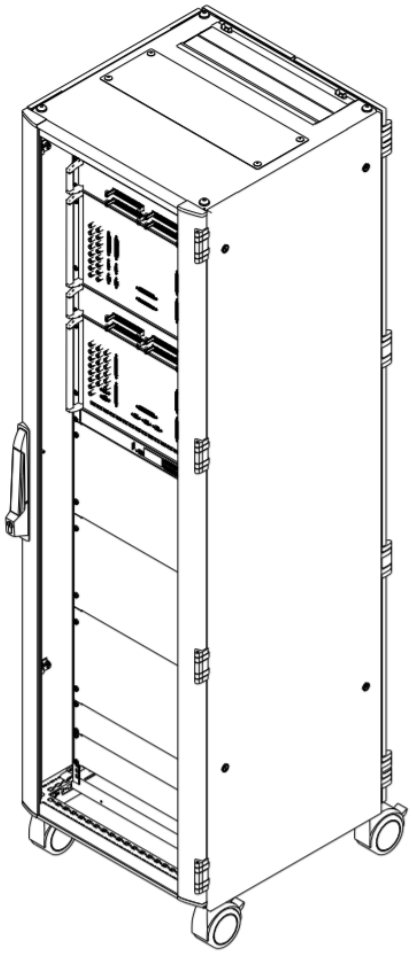
The connector to be used is a terminal block type connector. To meet this requirement, TY has provided a cable which breaks out one BNC into wires suitable for terminal block connection.

[2] Source: PEC Technical Reference Manual Auxiliary IO System

4 System components

4.1 Cabinet

The components are housed in a 19-inch rack cabinet (Figure 8). Connectors for communications and IO signals are located on front side in order to be easily accessible.



Manufacturer and part number:	RITTAL DK 5529.151
Installation height for components:	42 U
Basic material:	Sheet steel
Door material:	Glass
Dimensions:	Width: 600 mm Height: 2000 mm Height (with rollers): 2142 mm Depth: 600 mm
IP rating:	IP55
Color:	Custom defined color and branding
Additional equipment:	Climate control unit Rollers

Figure 8: Typhoon Cabinet

4.2 Prototype HIL simulator

The prototype HIL Simulator is based on the existing HIL simulators from TY, optimized for automotive applications.

The PANDA Typhoon cabinet contains two HIL simulators, which work in parallel mode as a single device (Figure 9).

Figure 9: Typhoon HIL simulator

4.2.1 HIL Connect

HIL Connect is a 19", 6U rackmount unit, designed to convert standard HIL IO levels to customer specification.

PANDA HIL Cabinets contain two HIL Connects, custom designed per requirements of partners in **Typhoon HIL hardware equipment requirements** (the Spec document), one per each physical HIL Simulator.

The connector layout is designed to meet the defined layout per ULille requirement in the best possible way, in order to keep the logical and functional properties in terms of equipment that will be connected to the HIL System.

As stated in 4.2, two HIL simulators are paralleled, and the whole system (2 x HIL simulator + 2 x HIL Connect) behaves as a single device (Figure 9).

4.2.2 HIL Connect 1

HIL Connect 1 is intended for the connection of HEV emulated power stages for e-subsystem real testing and dSPACE DS2103 D/A board. It contains the following connectors:

1. BNC for voltage, current, speed and torque measurements of power emulators
2. DSUB for machine encoders and torque meters; for PWM signals for power emulators
3. BNC for DS2103 D/A board connection (16 channels)
4. DSUB for spare analog and digital IO

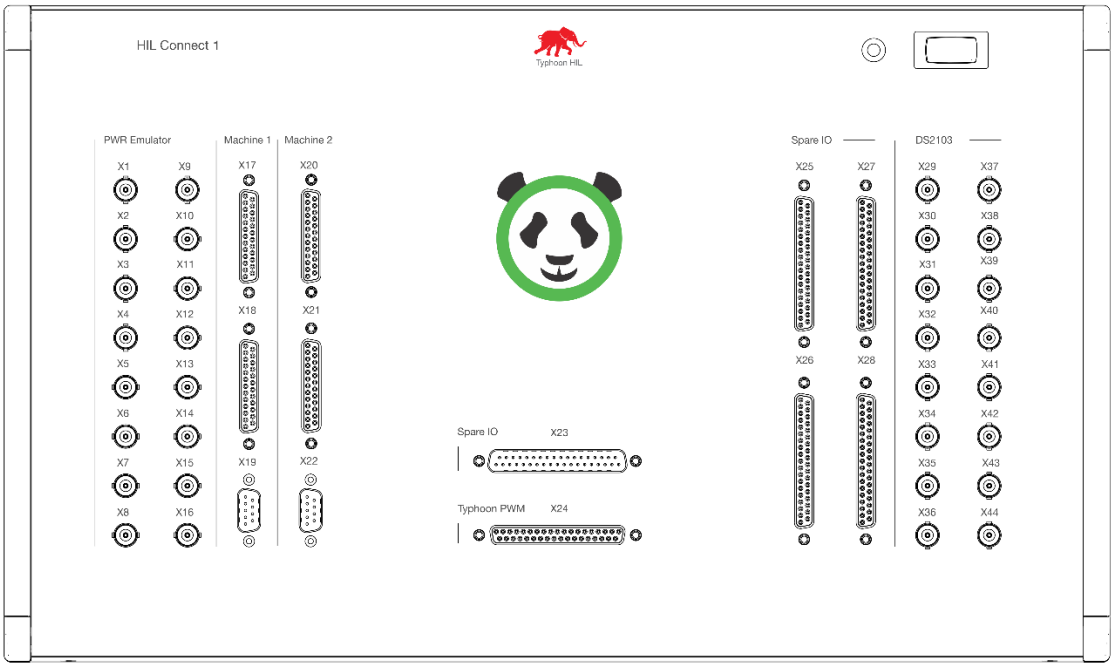


Figure 10: HIL Connect 1 front layout

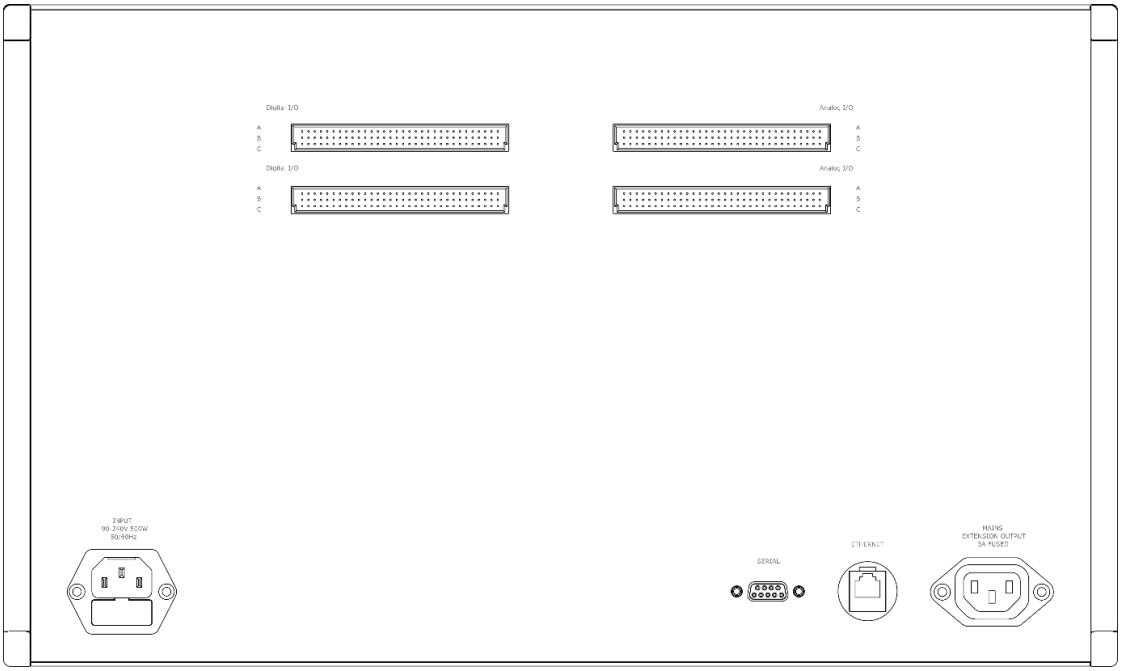


Figure 11: HIL Connect 1 rear layout

4.2.3 HIL Connect 2

HIL Connect 2 is intended for the connection of Cinergia amplifier and thermocouples for Battery real testing, dSPACE modules, NI PXIe-7856R IO Module Card, and PEC SBT8050 Battery Life Test System. It contains the following connectors:

1. BNC for Cinergia AIO
2. DSUB for Cinergia AIO and DIO
3. DSUB for CAN BUS
4. DSUB for dSPACE DS4003 (NI PXIe-7856R) DIO (3 DIO DSUB Connectors - 16 x DI and 16 x DO each)
5. DSUB for dSPACE DS5101 PWM (12 PWM inputs, and 12 PWM outputs)
6. BNC for dSPACE A/D module DS2002/2003 (16 Channels)
7. BNC for spare AI (NI PXIe-7856R)
8. DSUB for spare AO (NI PXIe-7856R, PEC SBT8050)

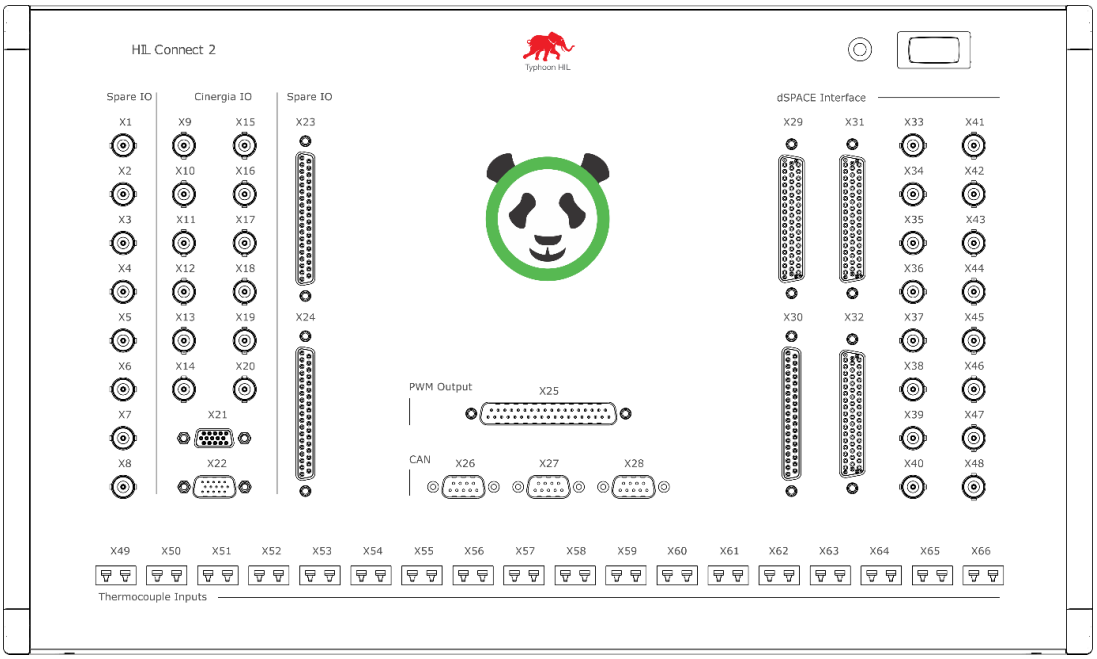


Figure 12: HIL Connect 2 front layout

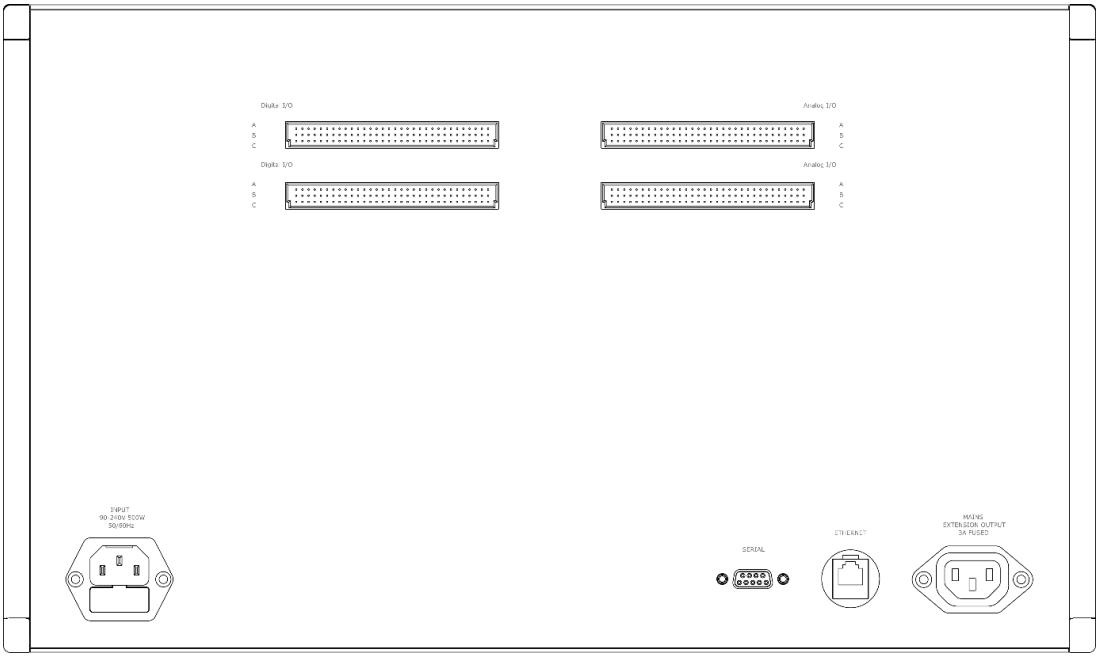


Figure 13: HIL Connect 2 rear layout

5 HIL System inputs and outputs

Based on the requirements and specifications from the project partners, Typhoon HIL has designed an interface system with the following characteristics (Table 1):

Table 1: IO per cabinet

Type	Range	Qty	Bandwidth	Accuracy	Connected device	HIL Connect	Comment
Analog input	±10 V	16	0-200 kHz	0.1% + 0.5 mV	Power emulators	1	1 MΩ input impedance
Analog input	±10 V	16	0-200 kHz	0.1% + 0.5 mV	DS2103	1	1 MΩ input impedance
Analog input	2 V + 500 mV	18	-	0.1% + 0.1 mV	Thermocouples K type	2	-100 °C ... 900 °C Resolution 0.03 °C
Analog input	±10 V	6	0-200 kHz	0.1% + 0.5 mV	Cinergia	2	100 kΩ input impedance
Analog output	±10 V	6	0-200 kHz	0.1% + 0.5 mV	Cinergia	2	20 mA max current
Analog output	±10 V	16	0-200 kHz	0.1% + 0.5 mV	DS2002/2003	2	20 mA max current
Digital Input	5 V	24	-	-	Encoders	1	4 x incremental encoders (KHO5S or similar) Differential inputs (RS422)
Digital input	5 V TTL/CMOS	48	-	-	DS4003 / NI PXIe-7856R	2	24 V maximum input voltage
Digital input	5 V TTL	12	-	-	DS5101	2	Buffered input
Digital input	24 V	3	-	-	Cinergia	2	0..0.8 V logical low 2..30 V logical high
Digital Output	5 V TTL	12	-	-	Power emulators PWM	1	20 mA max output current
Digital Output	5 V CMOS	48	-	-	DS4003 / NI PXIe-7856R	2	400 Ω output impedance
Digital Output	24 V	4	-	-	Cinergia	2	Push-Pull
Modbus	-	1	-	-	-	-	-
RS232/485	-	2	-	-	-	1	Serial to Ethernet
CAN	-	3	-	-	-	2	Can BUS for 3 sensors

Spare IO is included without additional cost, and can be used for future expansions, or for UTCN and VUB needs (Table 2).

Table 2: Spare IO per cabinet

Type	Range	Qty	Bandwidth	Accuracy	Comment
Analog output	±10 V	100	0-200 kHz	0.1% + 0.5 mV	2 mA max. load (PEC SBT8050)
Analog input	±10 V	8	0-200 kHz	0.1% + 0.5 mV	1 MΩ input impedance (NI PXIe-7856R)
Digital input	5 V CMOS	30	-	-	(NI PXIe-7856R)
Digital output	5 V CMOS	48	-	-	(NI PXIe-7856R)

6 Cloud Simcenter Amesim– Typhoon HIL co-simulation user guide

In order to perform cloud-based co-simulation tests using the described testbeds, there are some additional prerequisites which need to be detailed first. The co-simulation setup needs:

- A computer which runs Simcenter Amesim, which is able to accept SSH connections
- A computer which is used to manage the Typhoon real-time platform
- The real-time Typhoon test platform
- A hardware component which interfaces with the real-time device

6.1 Simcenter Amesim Cloud Computer

While a Cloud computer can have a strict definition of a virtual machine which runs in a server cluster, for all intents and purposes, it can be a computer which can accept connections from another one through the internet. Internally, tests were ran using an Amazon Web Services (AWS) Elastic Cloud Compute (EC2) machine, but they can be replicated with a personal computer or using another on-demand computing provider. The choice should depend on the location of the available options; in general, the closer it is to the Typhoon HIL device, the better. While in theory both Windows and Linux should work, only a Windows configuration has been validated. From this point on, the steps will assume that Windows is being used, on both the Simcenter Amesim and intermediary tunnel computers.

Once the choice (AWS, Azure, local computer; configuring an on-demand virtual machine is different to each provider, with guides available on the Internet) is made, some preparation is needed to perform the Internet communication. First, Simcenter Amesim must be installed on the computer. The simulation will run on it, so it should be able to run simulations reasonably fast (hard requirements are not specified, but something from the last few years should be adequate).

Note: Everything that is related to the SSH connection can be skipped the Simcenter Amesim machine and the real-time device are connected through a VPN. If the Typhoon HIL device is visible to the computer, then a SSH tunnel is not necessary. Every connection method has its own pros and cons.

Once this is done, the computer should be made open to SSH connections. This is done so that it can communicate with the Typhoon device through a secure link, which is relatively easy is configure. Linux should be able to do this straight away, while Windows must have the OpenSSH server activated. After a SSH server is active, the network must be configured so that the computer can receive and accept connections on the SSH port. For cloud-hosted machines, this should be done from the control panel which should have a section regarding inbound rules (AWS has the **Security** tab for this).

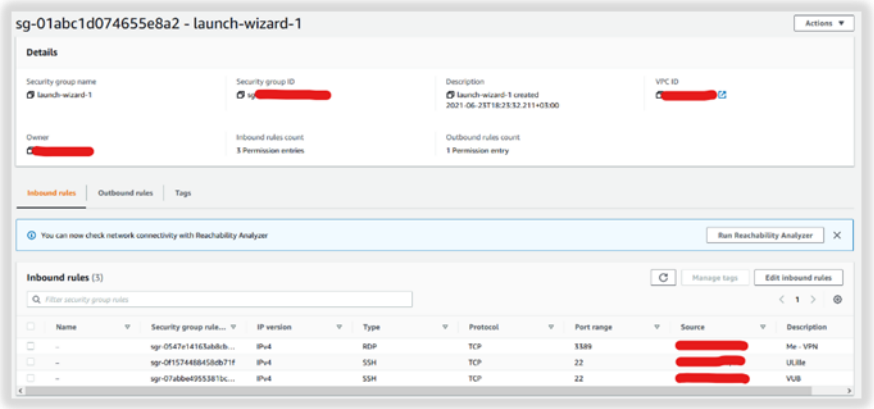


Figure 14: SSH configuration

Generally, they require the IP address of the computer which will connect via SSH and the port that it'll be using. The public IP address of a computer can be seen using a webservice (such as MyIP) and the port is (by default) **22** [MyIP 2022].

When using a local computer, it is required to port forward the port **22** to the local network address of the PC. Achieving this is different to each router manufacturer, but guides exist for most brands and if not, the general steps are the same. **Take note that this may not be an option present in the interface of some routers.** In broad strokes, this tells the router where to redirect the traffic on that specific port inside its network. Should the Internet provider use dynamic addresses for its customers, researching whether they offer the possibility to register an address in their dynamic DNS for future ease-of-use is recommended.
After this is done, the proxy Typhoon computer should be able to connect via SSH to this one.

In addition to the SSH server, Python version 3.8.X should also be installed on the computer (this version is mandatory as the Simcenter Amesim communication library was compiled for it), along with the Typhoon API module (see installing Python modules) [PyPi 2022], [Python 2022].

6.2 Typhoon HIL control computer setup

When using SSH tunnels, an additional computer should be used to facilitate the connection between Simcenter Amesim and the real-time device. For this to work, the computer and the Typhoon HIL ECU should be connected on the same network.

The SSH connection should have two tunnels set up, one for the Modbus (set up as a remote tunnel) and one for the remote connection (if controlling the Simcenter Amesim computer from this one is needed.) For example, Figure 15 shows a configuration that was used between two Windows systems (PuTTY is just a graphic interface, the same settings can be applied for any other SSH client.)
In this window are specified the address of the computer to which the client will connect to (for example, the virtual AWS machine, whose address can be found by visiting the same website mentioned earlier) and the port (by default, it should be 22.)

After this, the username of an account present on the computer should be specified. In Amazon's case, the default is **Administrator**. If not specified, the name **root** will be used which might not exist.

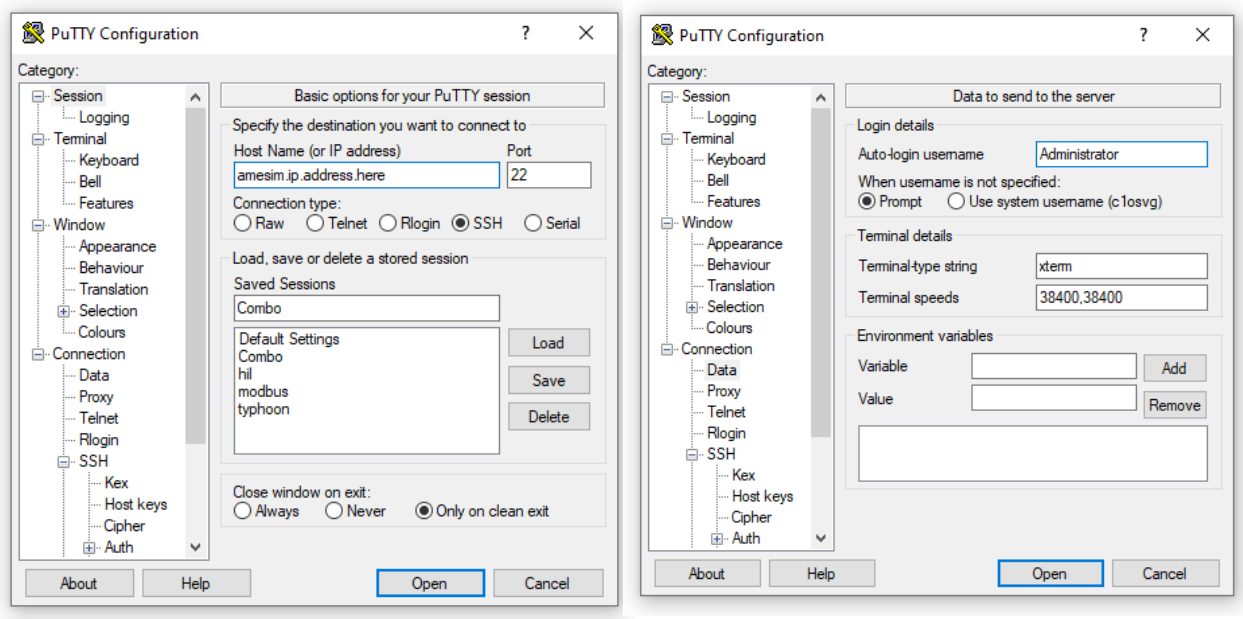


Figure 15: PuTTY Session (left) and Connection Data (right) configuration settings

Finally, two tunnels should be specified as in Figure 16:

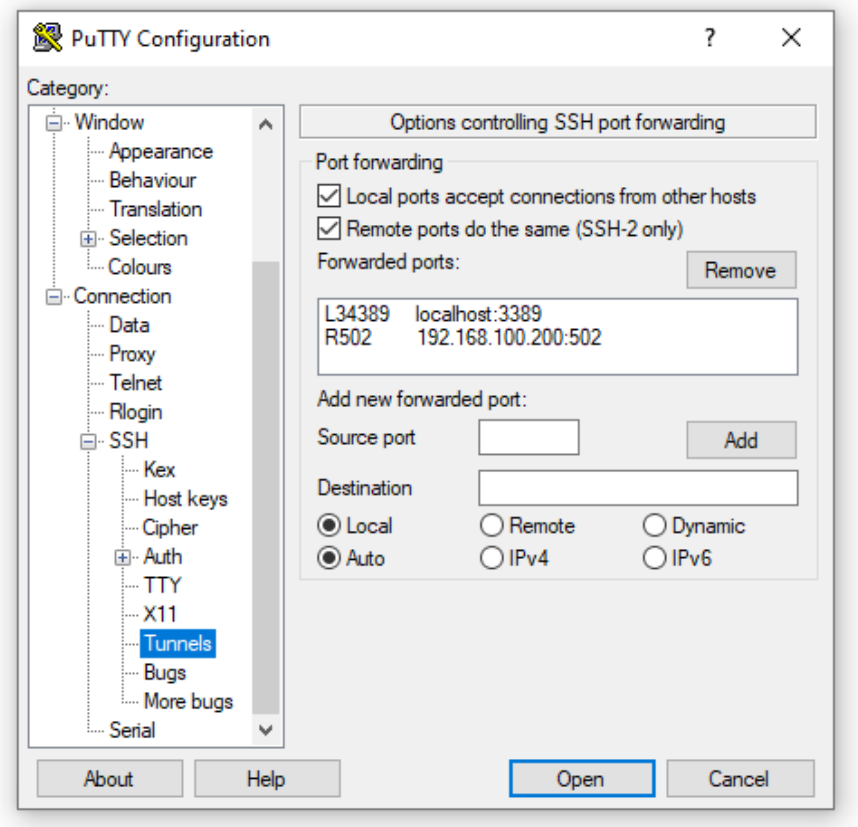


Figure 16: Tunnel configuration

- In case the target computer is running Windows, a local tunnel can be defined on any unused source port (in this picture, 34389) and have the destination as **localhost:3389**. With it, it's possible to connect to the Simcenter Amesim computer through **Remote Desktop Connection**, using **localhost:34389** (or the specified port.)
- For the Modbus to work through SSH, a link to the HIL device must be defined. This is a remote tunnel which means that it is configured from the point-of-view of the Simcenter Amesim computer. The source port can be any unused port (in this picture, 502) and the destination should be the address of the Typhoon device with the port used by it for the Modbus (more often than not, 502.) In order to choose a correct address for the device, checking the current network address can be done by accessing a command prompt. Opening a terminal and typing **ipconfig** (Windows) or **ifconfig** (Linux). InFigure 17, the computer is on the 10.146.248.X network. This means that the HIL device should have the same three starting parts, and a different final one, which can be any number from 1 to 254. Of course, the address that is picked must not be used by another device in the network.

```
C:\Users\c1osvg>ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection* 11:

   Connection-specific DNS Suffix  . : net.plm.eds.com
   IPv4 Address. . . . . . . . . . . : 10.146.248.128
   Subnet Mask . . . . . . . . . . . : 255.255.255.255
   Default Gateway . . . . . . . . . : 0.0.0.0
```

Figure 17: Identification of system configuration

After the SSH connection is configured, the Typhoon intermediary PC should be able to connect to the Simcenter Amesim computer (with a password if necessary.) While testing, the connection has the habit of crashing, so it should be checked before a simulation with a simple command like **dir** or **ls**.

6.3 Preparing the Typhoon HIL test platform for co-simulation

To complete this step, a hardware-in-the-loop testbed, such as those described in Sections 2 through 5 of this report, must be set up and operational. If the testbed is working locally, it is ready for co-simulation.

In order to perform a co-simulation through the Internet, the Typhoon HIL ECU additionally needs to have an active Internet connection and has to be exposed to the Simcenter Amesim computer. Alternatively, the device needs to be on the same network as the computer which may connect through SSH to the Simcenter Amesim one. Either way, the real-time machine should have the Ethernet cable plugged in the port labeled 1, used for Modbus communication.

6.4 Cloud co-simulation steps

Once all the requirements are met, the process for performing a cloud co-simulation can start. Before running a co-simulation, two models must be prepared which will be loaded on both platforms (Simcenter Amesim and Typhoon.) If starting with a Simcenter Amesim EMR model, it can be converted to Typhoon using **PANDA Model Converter** (see its manual for details.) When the same model is present in both software, the part which be running on the other platform can be deleted. For example, here is a BEV model which will be tested in a battery HIL simulation (Figure 18).

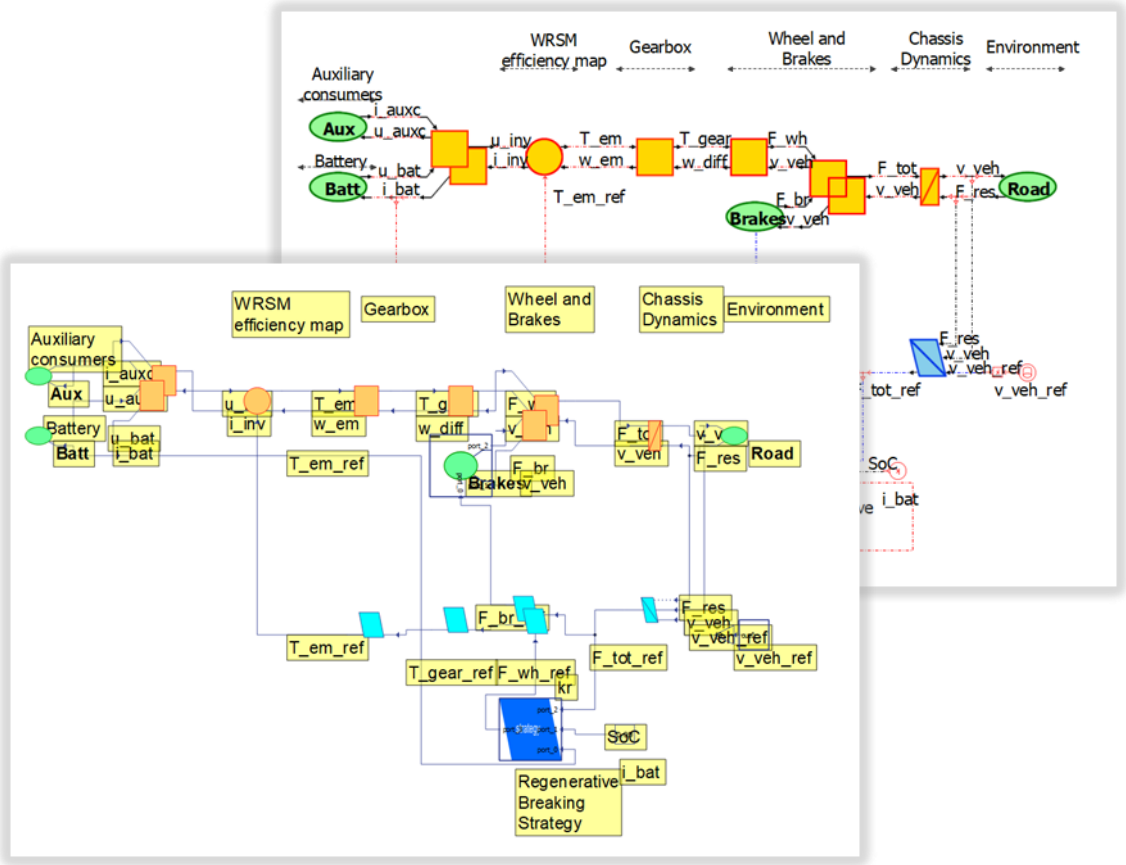


Figure 18: Co-simulation example in Typhoon HIL Control Center (front) and Simcenter Amesim (back)

In this case, the mechanical part is left inside Simcenter Amesim and the electrical components inside Typhoon, leaving the connections between the two parts open.

Then, communication components need to be added in each model, being careful to match the order of the signals (Figure 19).

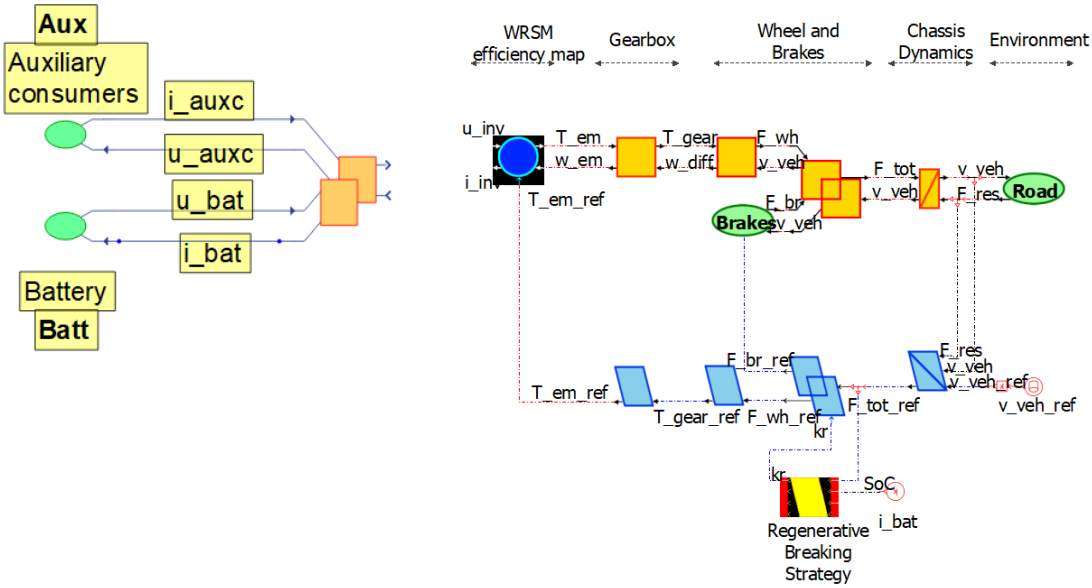


Figure 19: Communication between co-simulation parts

- In Simcenter Amesim a **shared memory co-simulation** block is added. It exchanges data with the Python script which uses talks through Modbus with Typhoon. In addition to it, the time sync (to run the simulation in "real-time") and the run stats (to check overrun) blocks must also be added. The parameters of the SHM should be **master** for the mode and the sample time should be larger than the latency between the Simcenter Amesim computer and the Typhoon device (Figure 20).

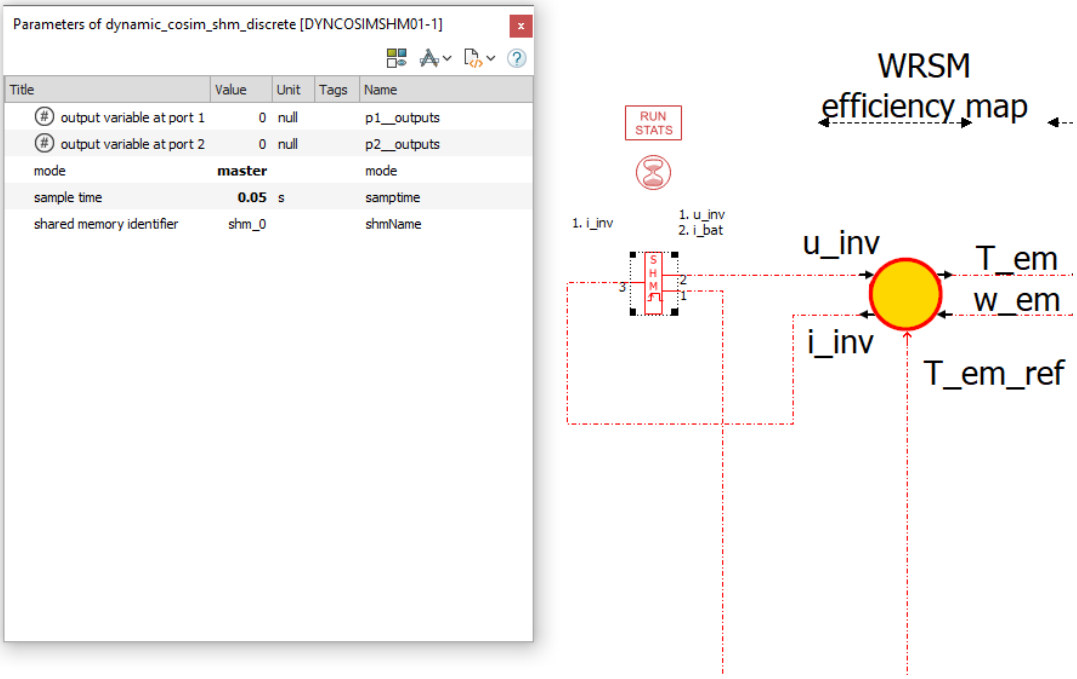


Figure 20: Example of data exchange

- In Typhoon a **ModBus Device** is added which will act as a server to which the Python script will connect to. The signals should be in the same order as they are in Simcenter Amesim (from top to bottom.) In case of multiple signals, a **Bus Join** or a **Bus Split** should be present at the holding input or the holding output, respectively. The other ports should be plugged with constants or terminations. In the parameters of the Modbus block, the configuration variable name should match the one in the initialization script (here it is **modbus_config** but anything else can be used, as long as it matches) and the execution rate should be the same as the rest of the model (Figure 21).

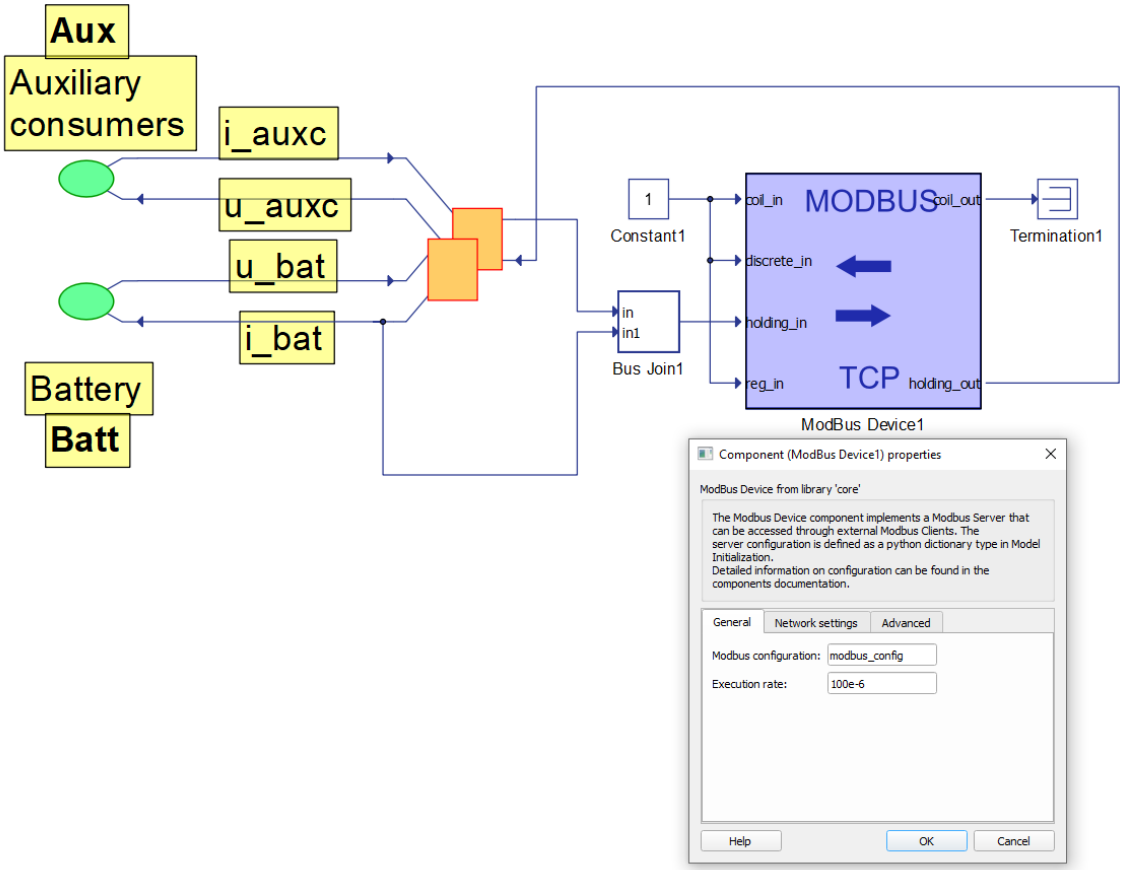


Figure 21: Typhoon Modbus communication example

- In the initialization script, an object variable should be placed which describes the options of the Modbus block. In this model, the code is shown in Figure 22:

```
modbus_config = {
    'ip_addr': '192.168.100.200',
    'port': 502,
    'netmask': '255.255.255.0',
    'slave_id': 1,
    'coil_registers': [],
    'discrete_registers': [],
    'input_registers': [],
    'holding_registers': [
        {'addr': 0, 'length': 4,  'data_type': 'float', 'word_order': 'BE',
             'io_type': 'in'},
        {'addr': 4, 'length': 4,  'data_type': 'float', 'word_order': 'BE',
             'io_type': 'in'},
        {'addr': 8, 'length': 4,  'data_type': 'float', 'word_order': 'BE',
             'io_type': 'out'},
    ]
}
```

Figure 22: Modbus configuration example

- **ip_addr** should match the remote tunnel address which was configured in PuTTY
- **port**, the same as above (the port in the destination part, not the source port)
- **holding_registers** has a few rules that must be followed:
  - the number of blocks inside the list should match the number of signals exchanged between Simcenter Amesim and Typhoon (i.e number of inputs plus number of outputs)
  - **addr** should increase by 4 for each consecutive line
  - **io_type** should be **in** or **out** depending whether the signal enters or exits the Modbus block. The inputs must come before the outputs (this is due to the script, as it expects the inputs to be first in line)!

Now, the models are configured for a Cloud co-simulation. Next comes the communication script which runs on the Simcenter Amesim computer. As discussed in the preparation part of the document, Python 3.8 must be installed on it. In addition to the script, the files **AmeCommunication3.py** and **binding_amecommunication.pyd** must be present next to it. Without these, the script cannot run. The highlighted lines are the ones which need to be modified when running a new model (see Figure 23). This is what they should contain, line by line:

- **n_inputs_from_hil** – from the perspective of the Simcenter Amesim model, this number should match the number of outputs on the shared memory block (the ones on the right-hand side)
- **n_outputs_to_hil** – from the perspective of the Simcenter Amesim model, this number should match the number of inputs on the shared memory block (the ones on the left-hand side)
- the port should be set to whatever it was set in PuTTY as the source port for the remote tunnel
- *optional* – if using a VPN or the Typhoon machine is exposed, then the host must also be changed to the address of the real-time device (**localhost** is used only for SSH tunnels)

With the script ready, everything is set and the simulation can be run. If your setup uses SSH tunnels, the intermediary computer should connect to the Cloud machine, start the Typhoon simulation, start the Simcenter Amesim simulation and launch the script with the command **py cosim.py** (or **python**, depending on the installation; **cosim.py** is assumed to be the name of the file containing the script; one should be careful to start the command prompt in the location of the script or navigate to it). If everything is ok, the two devices should exchange data (the flow of data can be seen in the prompt running the script).

```
import typhoon.api.modbus as modbus
from typhoon.api.modbus.exceptions import ModbusError
from typhoon.api.modbus.util import uint16_list_to_double, double_to_uint16_list
import AmeCommunication3 as amecom

def chunks(lst, n):
    """Yield successive n-sized chunks from lst."""
    for i in range(0, len(lst), n):
        yield lst[i:i + n]

n_inputs_from_hil = 2
n_outputs_to_hil = 1
modbus_client = modbus.TCPModbusClient()
try:
  modbus_client.set_host("localhost")
  modbus_client.set_port(502)
  modbus_client.set_endianness(modbus.TCPModbusClient.BIG_ENDIAN)
  modbus_client.set_auto_open(True)
  modbus_client.set_debug(True)
  modbus_client.set_timeout(10)
except ModbusError as ex:
  print(f'Modbus initialization failed: "{ex}"')

shm = amecom.AmeSharedmem()
shm.init(False, 'shm_0', 2 + n_inputs_from_hil, 2 + n_outputs_to_hil)
ret = shm.exchange([0.0] * (2 + n_outputs_to_hil))

while (True):
  try:
    hil_input_as_int16 = modbus_client.read_holding_registers(0, 4 * n_inputs_from_hil)
    hil_input_as_double = [uint16_list_to_double(chunk) for chunk in \
 list(chunks(hil_input_as_int16, 4))]
    print('HiL receive:',hil_input_as_double)
    send = ret[:2] + hil_input_as_double
    print(send)
    ret = shm.exchange(send)
    hil_output = [double_to_uint16_list(num) for num in ret[2:]]
    hil_output = [item for sublist in hil_output for item in sublist]
    modbus_client.write_multiple_registers(4 * n_inputs_from_hil, hil_output)
    print('HiL send:', ret[2:])
  except Exception as ex:
    print(ex)
    break

shm.close()
modbus_client.close()
```

Figure 23: Example communication script, run on the Simcenter Amesim computer

6.5 Troubleshooting

6.5.1 The simulation was started in Simcenter Amesim but it cannot be stopped

When running a simulation with a generic co-simulation block, Simcenter Amesim will not stop a simulation on-demand if the communication block has not connected with its pair. To forcibly end a simulation, start **Task Manager**, switch to the **Details** tab (Figure 24), and end the tasks named **STSDSIMDaemon** and **STSSIMManager** and then click **Stop** in Simcenter Amesim again.

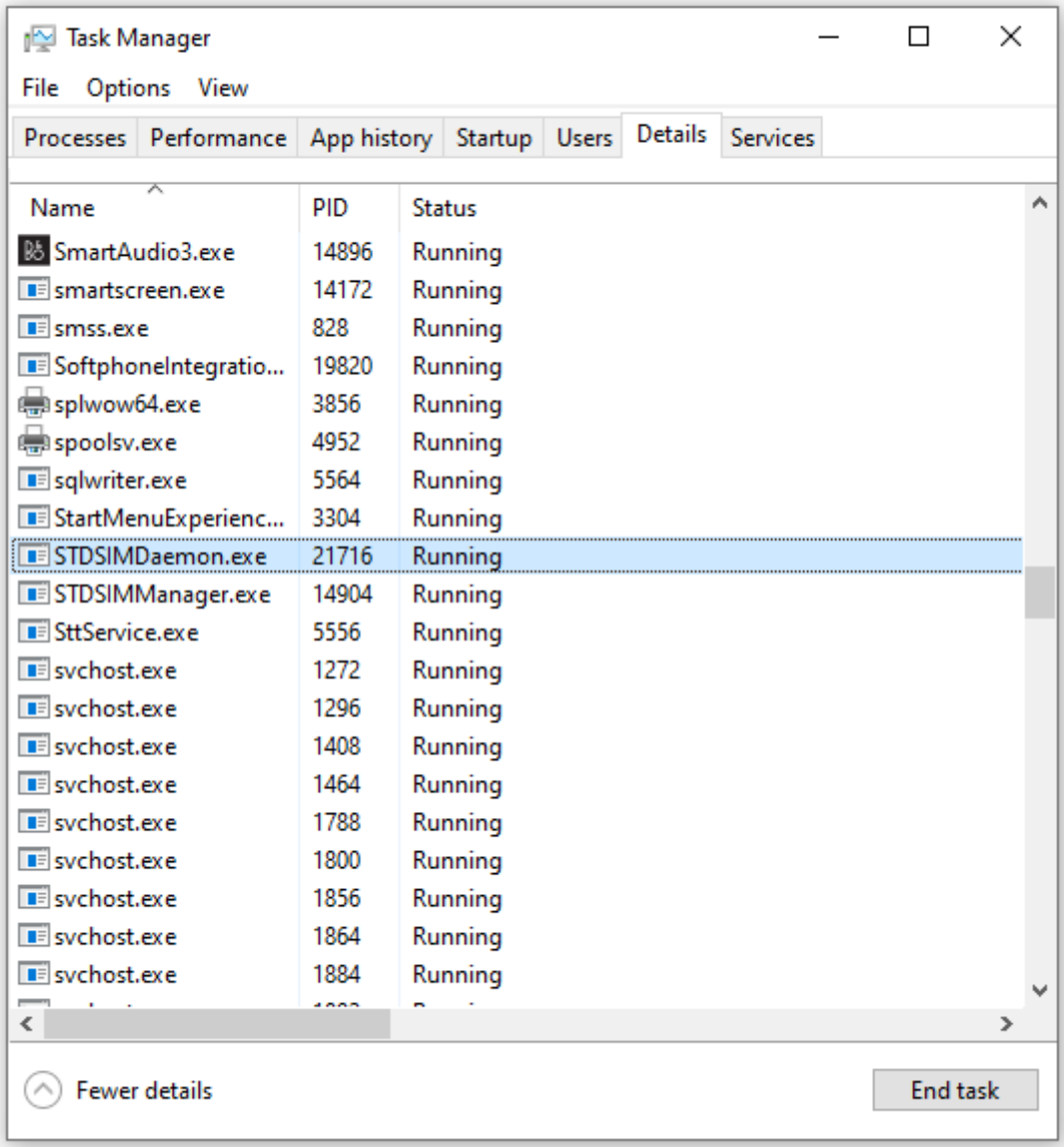


Figure 24: Troubleshooting using Task Manager

6.5.2 The simulation stops after a few steps

This might happen when the dynamics of the model cannot handle the exchange frequency between the two parts. The only way to fix this is to decrease the exchange step inside the shared memory block. However, this might lead into the next issue.

6.5.3 The simulation takes longer than it should

If the exchange frequency is set up high enough that an exchange exceeds the latency between the two sides of the model, then the Simcenter Amesim part will run in “slow-motion”, where a second of a simulation might be computed in two seconds of real time.

6.5.4 The script cannot connect to the Typhoon device when using SSH tunnels

This issue can have multiple causes, these being the most probable:

- *The address in the tunnel settings does not match the address in the Modbus configuration*. In addition to this, the address must be free. This can be tested by pinging the address and if it is free, the connection should fail or time out. After loading and starting the simulation on Typhoon (of a model which contains the Modbus) then pinging the same address should yield replies.
- *The tunnel is not remote*. The setup requires a tunnel from the Cloud computer to the HIL device so the starting point must be the Simcenter Amesim computer. When connecting through SSH from a computer inside the local network, the **Remote** option must be ticked for the Modbus tunnel.
- *The SSH connection has dropped*. After a simulation forcibly ends, more often than not the SSH connection drops. It can be checked by typing anything in the SSH terminal. If nothing appears, the connection must be established again.

7 Conclusion

This report features the different power interfaces required for performing battery and e-drive tests using the specific equipment featured in the PANDA project. A common framework has been defined using two real-time HIL simulators working in parallel with dedicated HIL Connects. The HIL connects have been adapted for PANDA partners to perform the defined real testing with a connection with the Simcenter Amesim cloud, but also the local materials of each partner. Lastly, a detailed user guide of how to set up a cloud connection with the testbed to perform co-simulation tests is provided.

This report includes the feedback of the partners from the different HIL testing. On the Typhoon HIL interface, there were no change from the initial design: each Typhoon HIL ECU has ensured the requested task. The main adaptation is related to the process to couple the could-based simulation with Simcenter Amesim and the local real-time simulator (Typhoon HIL ECU).

8 Deviations from Annex 1

Due to the COVID-19 crisis, there has been some delay in the building and the delivery of the battery system (Bluways) and the real-time simulators (TY). These delays have been included in the project extentions that was approved by INEA. The delays have not affected the final results. There is no other deviation according to the amended GANTT chart.

9 References

[Bouscayrol 2011] A. Bouscayrol, "Hardware-In-the-Loop simulation", Industrial Electronics Handbook, second edition, tome “Control and mechatronics”, Chapter 33, CRC Press, Taylor & Francis group, Chicago, March 2011, pp. 33-1/33-15, ISBN 978-1-4398-0287-8.

[Bouscayrol 2012] A. Bouscayrol, J. P. Hautier, B. Lemaire-Semail, "Graphic Formalisms for the Control of Multi-Physical Energetic Systems", Systemic Design Methodologies for Electrical Energy, tome 1, Analysis, Synthesis and Management, Chapter 3, ISTE Willey editions, October 2012, ISBN: 9781848213883.

[Genic 2017] A. Genic, C. Mayet, M. Almeida, A. Bouscayrol, N. Stojkov, “EMR-based Signal-HIL Testing of an Electric Vehicle Control”, *IEEE-VPPC’17*, Belfort (France), December 2017 (common paper Typhoon HIL and L2EP).

[Husar 2019] C. Husar, M. Grovu, C. Irimia, A. Desreveaux, A. Bouscayrol, M. Ponchant, P. Magnin, “Comparison of Energetic Macroscopic Representation and structural representation on EV simulation under Simcenter Amesim”, *IEEE-VPPC’19*, Hanoi (Vietnam), October 2019 (Siemens Software and L2EP/Univ. Lille within the framework of the H2020 PANDA project).

[MyIP 2022] “Check your IP address | MyIP.com”. Available: https://www.myip.com/. Accessed May 2022.

[PANDA 2020] A. Bouscayrol, A. Lepoure, C. Irimia, C. Husar, J. Jaguemont, A. Lièvre, C. Martis, D. Zuber, V. Blandow, F. Gao, W. Van Dorp, G. Sirbu, J. Lecoutere, “Power Advanced N-level Digital Architecture for models of electrified vehicles and their components”, *Transport Research Arena 2020*, Helsinki (Finland), April 2020 (within the framework of the PANDA H2020 European Project, GA #824256).

[PANDA D1.1] D. Chrenko, A. Bouscayrol, B. Lemaire-Semail, C. Irimia, “State-of-the-Art on vehicle simulation and testing”, PANDA H2020 GA# 824256, D1.1 Deliverable, public report, June 2019, [Online] available: https://project-panda.eu/ (accessed May 2022).

[PANDA D2.2] S. Costa, T. Kalogiannis, A. Bouscayrol, R. German, C. Husar, M. Ciocan, “Cloud-computing real testing of batteries”, PANDA H2020 GA# 824256, D2.2 Deliverable, confidential report, April 2022, [Online] Published executive summary available https://project-panda.eu/ Accessed April 2022.

[PANDA D3.3] S. Costa, C. Husar, M. Ruba, F. Tournez, M. Ciocan, C. Martis, A. Bouscayrol, “Cloud-computing real testing of e-drives”, PANDA H2020 GA# 824256, 3.3 Deliverable, confidential report, April 2022, [Online] Published executive summary available https://project-panda.eu/ Accessed April 2022.

[PANDA D4.1] C. Husar, “Simulation platform and library”, PANDA H2020 GA# 824256, D4.1 Deliverable, confidential report, February 2020, [Online] Published executive summary available: https://project-panda.eu/ Accessed C. Husar October 2020.

[PANDA D5.1] R. German, A. Desreveaux, F. Tournez, A. Bouscayrol, A. Genic, C. Husar, “Real test of the battery of the HEV”, PANDA H2020 GA# 824256, D5.1 Deliverable, confidential report, June 2021, [Online] Published executive summary available: https://project-panda.eu/ (accessed December 2021).

[PANDA D5.2] M. Ahmed Sanchez Torres, A. Lievre, W. Lhomme, F. Tournez, A. Bouscayrol, “Real test of the e-drive of the P-HEV”, PANDA H2020 GA# 824256, D5.2 Deliverable, confidential report, February 2022, [Online] Published executive summary available: https://project-panda.eu/ (February 2022).

[PANDA D5.3] W. Lhomme, F. Tournez, S. Roquet, “Real test of the e-subsystem of the P-HEV”, PANDA H2020 GA# 824256, D5.3 Deliverable, confidential report, February 2022, [Online] Published executive summary available: https://project-panda.eu/ (February 2022).

[PyPi 2022] “Typhoon-HIL-API 1.18.0”. Available: https://pypi.org/project/Typhoon-HIL-API/ Accessed January 2020.

[Python 2022] “Installing Python Modules – Python 3.8.13 documentation”. Available: https://docs.python.org/3.8/installing/index.html. Accessed May 2022.

[Shuai 2018] Shuai Ma, Modi Jiang, Peng Tao, Chengyi Song, Jianbo Wu, Jun Wang, Tao Deng, Wen Shang, “Temperature effect and thermal impact in lithium-ion batteries: A review”, Progress in Natural Science: Materials International, Volume 28, Issue 6, 2018, Pages 653-666, ISSN 1002-0071, https://doi.org/10.1016/j.pnsc.2018.11.002.

PANDA partners

The author(s) would like to thank the partners in the project for their valuable comments on previous drafts and for performing the review.

Table 3: Project Partners

#	Type	Partner	Partner Full Name
1	UNIV	ULille	Université de Lille
2	IND	SISW	Siemens Industry Software SRL
3	UNIV	VUB	Vrije Universiteit Brussels
4	IND	VEEM	VALEO Equipement Electriques Moteur SAS
5	UNIV	UTCN	Universitatea Tehnica Cluj Napoca
6	SME	TY	Tajfun HIL (Typhoon HIL)
7			
8	UNIV	UBFC	Université Bourgogne Franche-Comté
9	SME	UNR	Uniresearch BV
10	IND	RTR	Renault Technologie Roumanie
11	SME	Bluways	BlueWays International bva
12	IND	TUV-BT	TUV SUED Battery Gmh.

This project has received funding from the European Union's Horizon2020 research and innovation programme under Grant Agreement no. 824256.

Appendix A – Abbreviations / Nomenclature

Table 4: List of Abbreviations / Nomenclature

Symbol / Shortname	
DC	Direct current
EMR	Energetic Macroscopic Representation
HEV	Hybrid electrical vehicle
HIL	Hardware in the loop
IO	Input-output
P-HIL	Power hardware in the loop
PWM	Pulse width modulator