Politecnico di Torino

Politecnico di Torino

DEPARTMENT OF CONTROL AND COMPUTER ENGINEERING (DAUIN)

MASTER DEGREE IN MECHATRONIC ENGINEERING

A.a. 2020/2021

Development of a HIL application for a BEV Powertrain Control Unit

Academic Supervisor:

Prof. Federico Millo

Company Supervisor:

Dr. Giulio Boccardo

Candidate:

Antonio Maria Dentamaro

Abstract

In manufacturing field, testing control algorithms can be time-consuming, expensive and potentially unsafe.

To remain competitive, to reduce time to market and to deliver high-quality and reliable control systems, test engineers have replaced traditional testing methods with Hardware-in-the-Loop (HIL) testing, in which the plant to be controlled is replaced by models running on a real time computer and connected to the physical controller by specific IO board.

A HIL based on a commercially available National Instruments NI platform has been developed to simulate a Permanent Magnet Synchronous Motor to test the response of an Electric Vehicle Control Unit (EVCU) under test. After connecting and emulating all the signals needed form the EVCU, the HIL application performances have been analysed by actuating a torque on the EVCU user interface and monitoring the EVCU behaviour, checking if the simulated inverter and electric motor are correctly responding to the commanded gate switch by computing the correct currents and torque.

As it will be shown in the thesis the HIL application works correctly even though some deviation between the actuated torque and the calculated torque, mainly due to a quite simple electric motor modelling approach.

As a future work this HIL application will be expanded including all the vehicle domain and additional control node (real or emulated), in order to reproduce as close as possible the controller operating condition as in the real vehicle.

Contents

List of Figures

1 Introduction

1.1 Motivations

In an era where the competition is the core of the society, the research of the fastest and the cheapest way to do something is the main objective of everyone.

In manufacturing field, testing control algorithms can be time-consuming, expensive and potentially unsafe, especially if something is tested by a real system. To remain competitive and deliver high-quality controller software, test engineers have replaced traditional testing methods with Hardware-in-the-Loop (HIL) testing. The latter allows engineers to verify the controller design without the complete system hardware, saving moneys and time, ensuring the highest possible safety.

By this kind of tool, the opportunity to simulate an entire system is given to the test engineers, how could create one or more simulation models which can be easily modified, in order to obtain the best result ever. Moreover, there is the chance to add some I/O interfaces, such as CAN and LIN, or processor, such as FPGA, to let the real-time simulated system communicate with external boards, which can be, in their turn, real or simulated systems. Reducing costs and time-consuming.

The safety is one of the main aspect to consider during a test. Most of the time, test engineers have to validate something completely new and, as a consequence, the reaction of that specific item could be unexpected and really dangerous.
As an example, think about a test of an autonomous driving car. Firstly, if something goes wrong, resulting in a crash, the vehicle could be permanently damaged, following in an increase of the costs of the project, since it could be needed to change the car with a new one; Secondly, the main consequence of a car crash could be an injury of someone if, for instance, the test is performed in a real street. On the other side, the latter can be substituted with a test track, but it will increase again the costs and it can't ensure that the entire entropy of the real world is reproduced during the test, resulting in a potential unsafe condition when the car will be commercialized.

The test of an ECU (electronic control unit) or, generally, a complex embedded board, could lead to some wrong computation or calibration, causing a permanent damage of the circuit of the real hardware. By the HIL, this kind of problem is completely avoided, since everything is simulated and in the worst case, if something isn't correctly set, the model won't correctly works and some software error could appear.

For all thees reasons, the HIL becomes the best way to test a real-time system, increasing the efficiency of the test and validation process, reducing costs and risks.

1.2 Hardware in the loop

1.2.1 How it works

HIL simulation is a technique for validating a control algorithm, running on a target controller under test, by creating a virtual real-time environment that represents the physical system to control. HIL helps to test the behaviour of a control algorithms without physical prototypes, saving time and moneys being not a real hardware.

Firstly, the developer have to create and simulate a virtual real-time implementation of physical components such as a plant, sensors and actuators on a real-time target computer. Then, the control algorithm is run on an embedded controller, instead, the plant or the environment model is run, in real time, on a target computer connected to the controller. The embedded controller interacts with the plant model simulation through various I/O channels.

In this way, if no mistakes have been done, wiring and configuring the simulator, the controller is tricked and it "thinks" to be connected to a real system.

The final step is to refine and to tune the software parameters of the control algorithm and gradually replace parts of the system environment with the real hardware components.

Only after all this steps, the test of a real system could be performed. For instance, if the hardware under test is the motor ECU of a vehicle, the latter could be tested on a roller bench.

By this approach, represented in figure 1, took from MathWorks website, in which the hardware under test is an embedded controller and the plant model is a representation of a physical system, the HIL simulation can eliminate costly iterations in hardware fabrication.

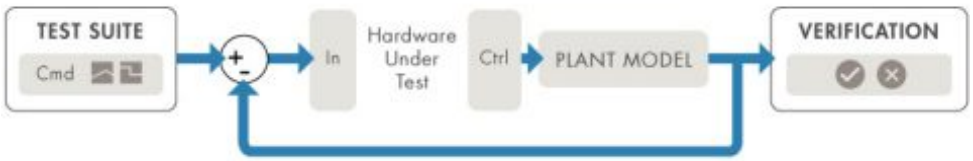


Figure 1: HIL simulation steps

A properly configured HIL simulation must include electrical emulation of sensors and actuators. They act as the interface between the plant simulation and the embedded system under test. The value of each emulated sensor is controlled by the plant simulation and it is read by the embedded system under test ($feedback$). Likewise, the embedded system under test implements its control algorithms by outputting actuator control signals, based on stimulus coming from the simulated plant. This signals are read by appropriate input board on the HIL simulator and changes in the control signals result in changes to variable values in the plant simulation.
As the example reported on the MathWorks website, a HIL simulation platform for the development of automotive anti-lock braking systems may have mathematical representations for each of the following subsystems in the plant simulation:

- Vehicle dynamics, such as suspension, wheels, tires, roll, pitch and yaw;
- Dynamics of the brake system's hydraulic components;
- Road characteristics.

As it can be seen in the figure 2, the hardware of the HIL is composed by modules. In many cases, any of the modules can be added or removed based upon the needs. For instance, taking into account how many digital or analog I/O channels are needed rather than a certain type of network, such as I2C, CAN, LIN, or there is the necessity of a lot of computational power, more than one real-time target computer could be added.
This kind of structure makes the HIL very flexible and easy to configure.

Figure 2: HIL

1.2.2 Use cases

Automotive systems

In context of automotive applications, which is the application field of this thesis project, vehicles are no longer self-standing mechanical entities driven by persons and Hardware-in-the-loop simulation systems provide such a virtual vehicle for systems validation and verification. Since in-vehicle driving tests for evaluating performance and diagnostic functionalities of, for example, an Engine Management Systems are often time-consuming, expensive and not reproducible, HIL simulators allow developers to validate new hardware and software automotive solutions, respecting quality requirements and time-to-market restrictions.

Nowadays, the electronic part is dominating on the mechanical part. More than 50 % of the cost of developing a vehicle comes from the design and validation of the electrical part.[1] The trend to introduce electrical components in a vehicle is grown exponentially in the last fifty years, as shown in figure 3.

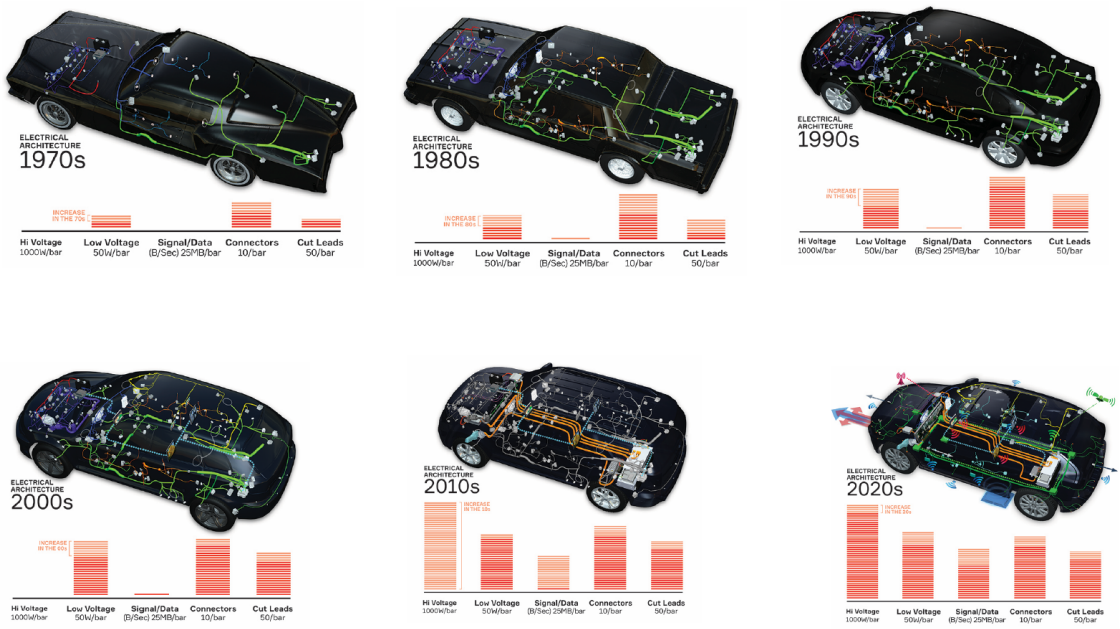


Figure 3: Trend of electrical components in a vehicle [1]

A modern vehicle is an interconnection of several intelligent entities cooperating to-

wards a common goal, such as lower emission, long range and higher safety. More than 100 ECUs embedded when all optional equipment is installed, an example of configuration is in figure 4 ; millions of software embedded lines; several in-vehicle networks to connect ECUs, which for different systems/services have different requirements and require different technologies.

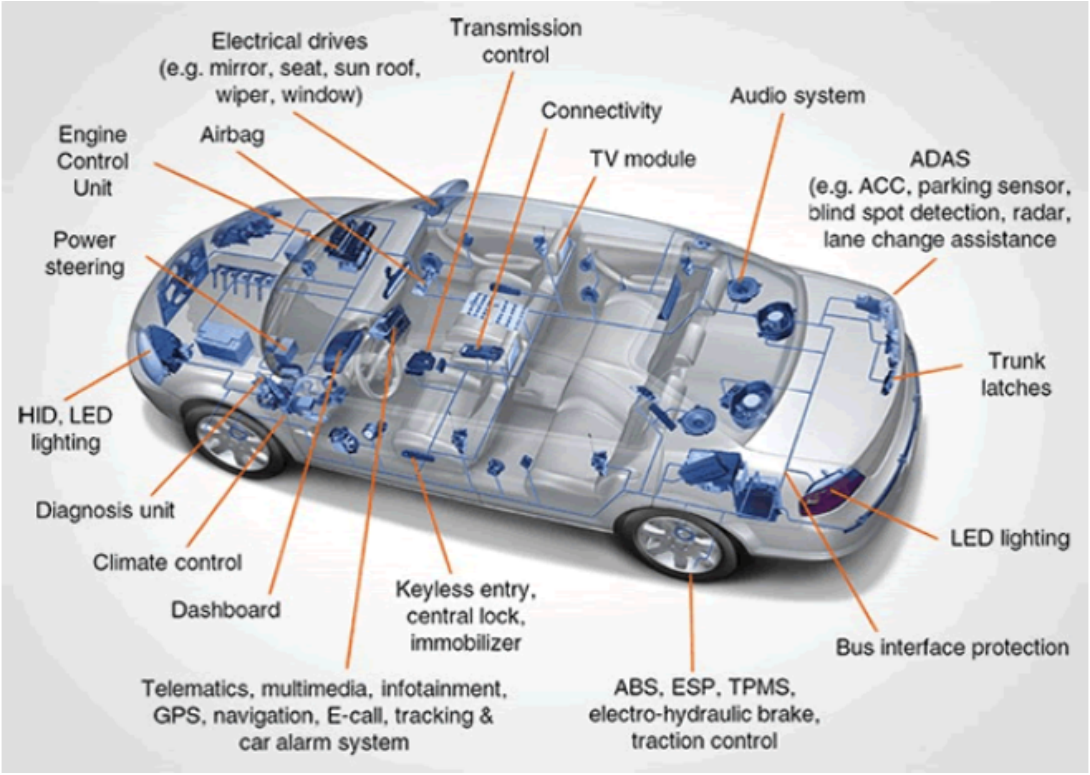


Figure 4: In-vehicle electronic equipment [1]

In a typical HIL Simulator, a dedicated real-time processor executes mathematical models which emulate engine dynamics. The use of an FPGA is mandatory to follow the small time constants of the power electronics, in order to test an ECU. In addition, an I/O unit allows the connection of vehicle sensors and actuators (which usually present high degree of non-linearity).

Then, the Electronic Control Unit under test is connected to the system and stimulated by a set of vehicle maneuvers executed by the simulator.

One of the main aspects is that, this kind of approach, offers a high degree of repeatability during testing phase.

In literature, several HIL specific applications are reported and HIL simulators were built according to some specific purpose. When testing a new ECU software release, for example, a first experiments can be performed in open loop and therefore several engine dynamic models are no longer required. The strategy is restricted to the analysis of ECU outputs when excited by controlled inputs. Then, when all the open-loop signals have been checked, the loop could be closed.

In this thesis, the first approach has actually been in open-loop, in order to check if all the signals coming from the simulator were in the pre-defined ranges and consistent with the expected signals. Then, the loop has been closed and the signals has been actuated from the Embedded System under test which, in its turn, receives signals fed back from the simulator.

Radar

Digital Radio Frequency Memory (DRFM) systems are typically used to create false targets to confuse the radar in the battlefield, but these same systems can simulate a target in the laboratory, in order to test the radar itself. This configuration allows for the testing and evaluation of the radar system, reducing the need for flight trials and field tests, reducing time and costs, and it can give an early indication to the susceptibility of the radar to electronic warfare techniques, reducing to the minimum the false targets.[2]

Robotics

Techniques for HIL simulation have been recently applied to the automatic generation of complex controllers for robots.

A robot uses its own real hardware to extract sensation and actuation data, then uses this data to infer a physical simulation (self-model) containing aspects such as its own morphology as well as characteristics of the environment.

Algorithms such as Back-to-Reality, as reported in [6] and Estimation Exploration, as reported in [7].

Power systems

In recent years, HIL for power systems has been used for verifying the stability, operation, and fault tolerance of large-scale electrical grids, especially because the necessary evolution of Green Power Production Technologies caused by the huge request of energy for civil users and factories.

Current-generation real-time processing platforms have the capability to model large-scale power systems in real-time. This includes systems with more than 10,000 buses with associated generators, loads, power-factor correction devices, and network interconnections, as in the Real-time transient stability simulation tool by OPAL-RT.

These kind of simulation platforms enable the evaluation and testing of large-scale power systems in a realistic emulated environment. Moreover, HIL for power systems has been used for investigating the integration of distributed resources, next-generation SCADA systems and power management units, and static synchronous compensator devices, as reported in [9].

Offshore systems

In offshore and marine engineering, control systems and mechanical structures are generally designed in parallel. Testing the control systems is only possible after integration. As a result, many errors are found that have to be solved during the commissioning, with the risks of personal injuries, damaging equipment and delays.

As a consequence, to reduce these errors, HIL simulation is gaining widespread attention, as reported in [10].

1.3 Development metrics

Assuming that the most effective method to develop an embedded system is the HIL simulation, some metrics during the development and test have to be considered. The main metrics to follow are: [2]

1. Cost;
2. Duration;
3. Safety;
4. Feasibility.

The first parameter should be a measure of the cost of all tools and effort that will take part to the project. The second parameter regards the duration of development and testing which will affects the time-to-market for a planned product and have to be as short as possible. Safety factor and feasibility are typically equated to a cost measure.
The following conditions are example cases where the use of HIL simulation is the best solution:

- The quality of testing enhanced;
- Demanding development schedules;
- Expencive plant;
- Early process human factor development.

1.3.1 The quality of testing enhanced

The use of a simulated systems in a HIL enhances the quality of the test just increasing the scope of the testing. Without this approach, an embedded system would be tested against the real plant, however, most of the time the real plant itself imposes limitations in terms of the scope of the testing. For example, testing an engine control unit against

a real plant can lead to dangerous conditions for the test engineers, such as testing at or beyond the range of the certain ECU parameters rather than testing and verification of the system at failure conditions. In these scenarios, the HIL provides the efficient control and safe environment where test or application engineer can focus on the functionality of the controller, avoiding to injure themselves and break real components.

1.3.2 Demanding development schedules

The demanding development schedules in the automotive, aerospace and defense programs in the last years, make it impossible to wait for a prototype to test an embedded systems. For this reason, in many cases the HIL simulation will be used in parallel with the development of the plant. For example, by the time a new automobile engine prototype is made available for control system testing, the engine controller algorithms could be tested by a HIL simulation.

1.3.3 Expensive plant

In many cases, the plant is more expensive than expected and it will probably lead to exceed the budget. Then, the best economical way to develop and test an embedded system is to connect it to a HIL simulator rather than the real plant.

As an example, in jet engine manufacturers the development of Full Authority Digital Engine Controllers (FADEC) for aircraft jet engines has extremely costly plant. However, a HIL simulator designed to test a jet engine manufacturer's complete line of engines is about ten times cheaper than a single engine, which can cost millions of dollars.

1.3.4 Early process human factor development

HIL simulation is a key step in the process of developing human factors to ensure usability and system consistency using software ergonomics, human-factors research and design, collecting usability data from man-in-the-loop testing for components which provides a human interface.

As an example, developing an FPV Drone, the behaviour of a flight control is defined by control algorithms. Changes in algorithm parameters can translate into more or less flight response from a given flight control input, which could be the movement of the a joystick lever. Likewise, changes in the algorithm parameters can also translate into more or less force feedback for a given flight control input, leading the drone to be more or less reactive to the stimulus. The "correct" parameter values are a subjective measure. Therefore, it is important to get input from numerous man-in-the-loop tests to obtain optimal parameter values. In many cases, more than one map containing different control algorithm parameters could be present in a drone and could be chosen by the user depending on the scenario.
In this case HIL simulation is used to simulate human factors. The flight simulator includes plant simulations of aerodynamics, engine thrust, environmental conditions, flight control dynamics and more. Prototype flight controls are connected to the simulator and test pilots could evaluate flight performance given various algorithm parameters.

Moreover, if the aircraft is bigger and more expensive than a FPV drone, the alternative to HIL simulation for human factors and usability development is to place prototype flight controls in an aircraft prototypes and test for usability during flight test. This is not the best way to do testing, since:

- **Cost**: A flight test is extremely costly and therefore the goal is to minimize any development occurring with flight test;
- **Duration**: Developing flight controls with flight test will extend the development duration of an aircraft. As mentioned before, using HIL simulation, the flight controls may be developed well before a real aircraft is available;
- **Safety**: Using flight test for the development of critical components such as flight controls has a major safety implication. Errors could be present in the design of the prototype flight controls, thus the result could be a crash landing;
- **Feasibility**: It may not be possible to explore certain critical timings of human

actions with real users operating a plant, such as sequences of user actions with millisecond precision. Likewise more than one scenarios could be tested, as particular meteorological conditions which could compromise, in this case, the flight.

1.4 Thesis goals

The project has been developed in Powertech Engineering srl, a consulting firm in automotive field in Torino. This activity is a part of a wider programme in development of the company's capabilities in Hardware-In-The-Loop scope.

The final goal of the project is to implement and configure an Hardware in the Loop (HIL) system to simulate a vehicle model, in order to test a complex embedded system, which is, in this case, represented by an EVCU (Electric Vehicle Control Unit) board.
The latter is responsible for all the signals necessary to actuate a single Permanent Magnet Synchronous Motor mounted on an electric vehicle. The EVCU will operate as if it is in a real vehicle communicating with the other nodes through the available networks, such as CAN and LIN, which will be simulated thanks to the HIL.

However, the main scope of the thesis is to close the loop between the HIL simulator and the EVCU in dyno mode. In this mode the EVCU ignores CAN and others I/O signals and directly drives PMSM motor.

The first step will be the configuring of the OPAL-RT PMSM VDQ model, which is a ready-to-use Permanent Magnet Synchronous Motor model, which will be, in this case, modified to better represent the motor to simulate.
This procedure is described in the chapter 3.3.1.

The second step will be to assemble the wiring harness. The NI simulator will be connected to the EVCU through Analog and Digital I/O, passing through three 3.3V-5V logic adapters for a few signals. The EVCU is connected to the Host PC via xETK. The EVCU power is taken from a bench power supply.
This procedure is described in the chapter 3.2.

The next step will be the parameters configuration. All the motor parameters will be set in order to better represent the motor to simulate. All the gains, the offsets and the mapping of the signals will be adjusted.
The setting guide is in chapter 3.3.2 .

Then, two user interfaces will be configured in order to monitor the simulation and to

command the necessary signals, as shown in chapters 3.2.2 and 3.3.3.

After that the hardware and software configuration will be done, the experimentation can begin. It consists on testing the alignment of the motor parameters estimated from the EVCU and the parameters in output from the simulated model, checking, at the same time, if the EVCU behaves as in a real vehicle as expected.
The experiment and the troubleshooting are described in chapter 4.

Finally, an additional experiment has been done, as described in chapter 5, where a part of the vehicle model has been added in the simulator, with the addition of the CAN network.
The vehicle model includes the vehicle dynamic, the driveline and a virtual driver who will actuate an action on the accelerator and the brake pedal, which will be transformed in the desired torque.

2 Toolchain

In the this section, the hardware and the software utilized in this project to develop the HIL application are illustrated.

2.1 Hardware

In order to implement the toolchain needed to carry out the HIL used in this project, a particular set of hardware components have been chosen and they can be divided into two categories:

- the hardware used to implement the simulator;
- the EVCU and the hardware used to communicate with it.

To better understand how the components have been utilized, all the items used could be distinguished into two different subcategories, as shown in the following table. Each subcategory is described further on and the entire system is shown in figure 5 :

Simulator side	**EVCU side**
Chassis	ECU
I/O interfaces	Gate driver board
Logic converters	Arduino Mega
	Power relè

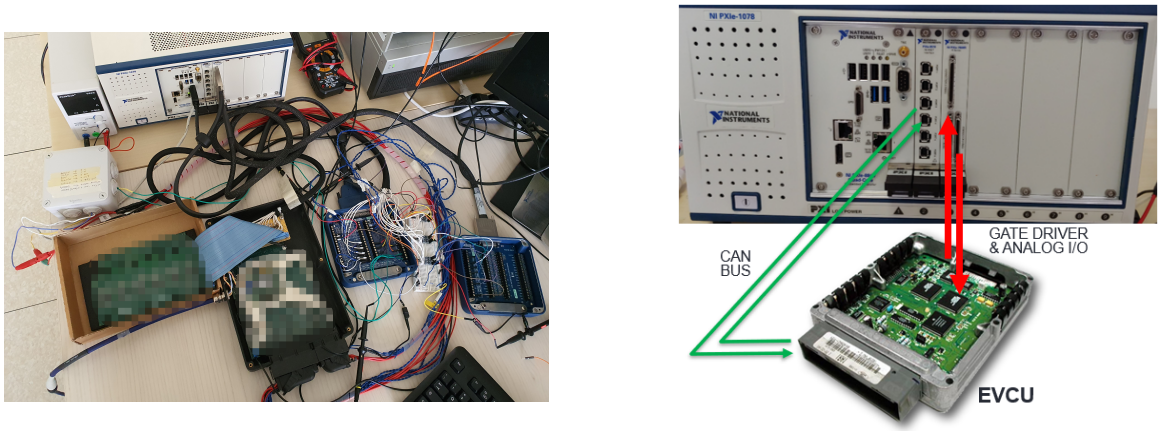


Figure 5: Whole system

2.1.1 EVCU side

ECU and GDB

On the EVCU side the hardware under test can be find. It consist in an Electric Control Unit (ECU) of a Battery Electric Vehicle (BEV) where a GDB is connected.

The ECU communicate with the simulator by analog and digital channels connected to the terminal blocks directly or passing through the logic converter modules. Furthermore, it is connected to the computer where the user interface is running by the xETK.

The xETK, in figure 6, as reported on ETAS website [11], implements a full functional Ethernet interface on board to facilitate direct connection to the PC. xETKs utilize the XCP-on-Ethernet protocol for data transfer. The benefits of this type of connection are:

- Direct connection between ECU and PC;
- Open standards-based architecture (Ethernet, XCP);
- Highest performance – supports rasters as fast as 10;
- Simultaneous access to a single ECU by up to four software applications;
- ETK compatible;

Figure 6: ETAS xETK

Setting a desired torque through the Host PC, the EVCU will generate six PWM signals to actuate the IGBT switches, simulated by the HIL, expecting a feedback from the

latter, through analog I/O pins, of the phase currents, the phase voltages, the DC_Link voltage and the sine and cosine signals of the resolver.
In addition, it is responsible to read the six PWMs signals of the IGBTs temperature and the excitation signal of the resolver.

Relays

In order to easily interact with the ECU, two relays have been added to remotely control the power supply to the ECU and to simulate the "KEY" state.
The relays are 3-24V clean contact relays, in this particular case, they have been piloted with 5V Logic level.
To implement this control unit, a strip board has been used, as shown in figure 7 , where the welds shown in figure 8 have been done according to the scheme described in chapter 3.2.1 .

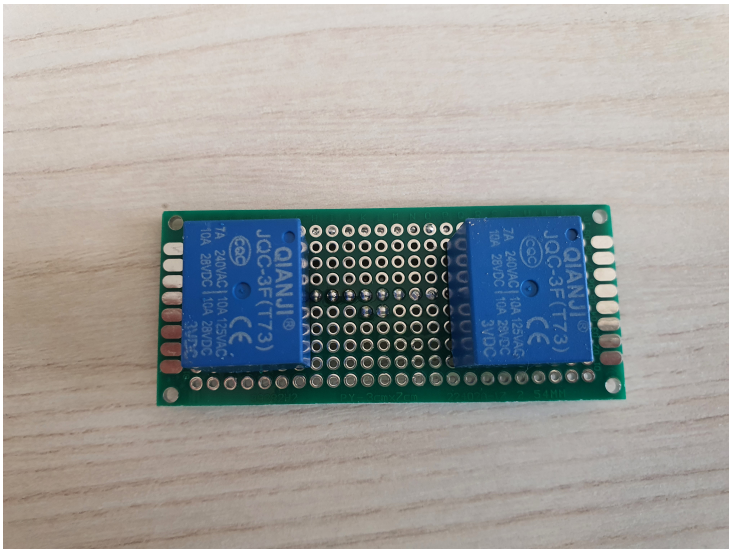


Figure 7: Relays board

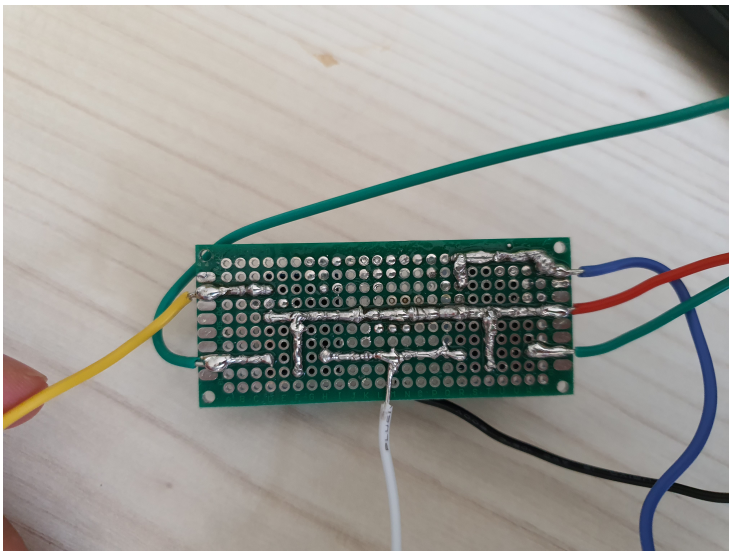


Figure 8: Welds on relays board

Arduino Mega

To pilot the relays an Arduino Mega has been selected. It has the task to read the user command from the simulator through the analog input pins and to write on its digital output pins the required value, HIGH or LOW. To be more specific, 3.3V from the simulator pins correspond to the HIGH output value.

In this way each relay could be remotely controlled from the user, almost instantaneously, through the computer running the user interface.

Arduino is necessary since the 3.3V coming from the simulator, up-scaled to 5V, is not powerful enough to actuate the relays.

An analog signal has been preferred since the different logic levels of the Arduino MEGA and the FPGA, as described before. In this way, any adapter is needed, since the 3.3V is directly read by the Arduino analog input pin.

The assembled part as been positioned in a box as shown in figure 9 and in figure 10.

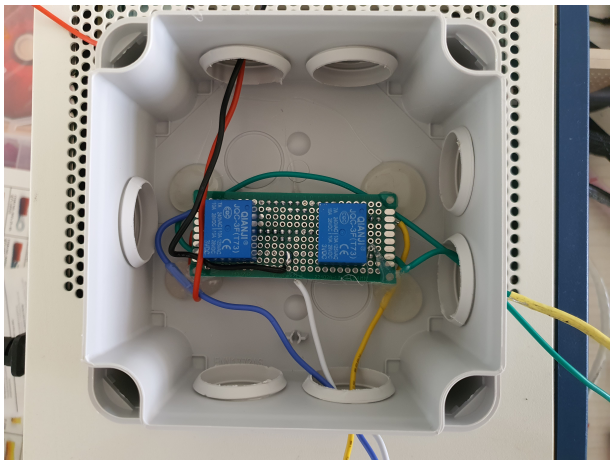


Figure 9: Relays board in the box

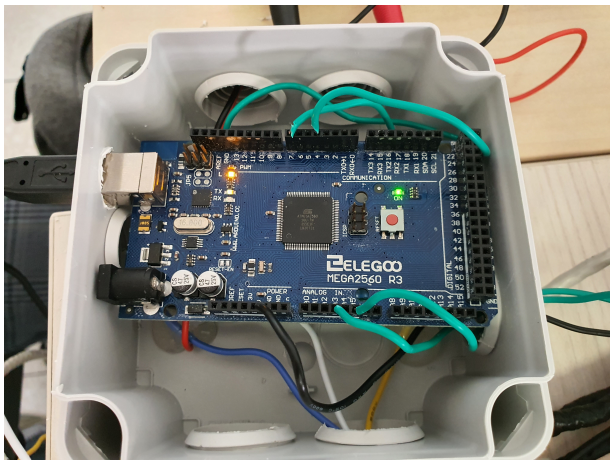


Figure 10: Assembled part in the box

2.1.2 Simulator side

Chassis

The core of the HIL is the **PXIe-8840** (fig. 11), which is an embedded controller for PXI systems, running a 2.6 GHz processor, developed by NI.
The module has two Ethernet 10/100/1000BASE-TX (Gigabit) ports, one of them has been used as mean of communication between the module and the computer running the user interface software. In addition, it has USB ports, a serial port, other I/O interfaces and an HDD where the simulation model is stored.

The previous module is mounted on a chassis, the **PXIe-1078** (fig. 12), characterized by a backplane composed by 9 PXI Express slots, with a maximum speed of 1.75 GB/s.

Figure 11: PXIe-8840

Figure 12: PXIe-1078

On the chassis is connected the FPGA, where the motor model, used in this thesis project, has been uploaded. The module is the PXIe-7868R (fig. 13), which allows you to direct control the I/O signals.

It has 18 analog output channels directly connected to the FPGA Kintex-7 325T, that guarantee a very precise timing as needed in the HIL appliocations, since the latter require a very high sample frequency of the order of 1 MS/s to simulate an electromagnetic model, as in this case. Thus, this speed performance is guaranteed only utilizing an FPGA.

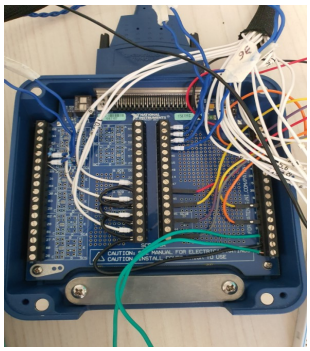
Finally, it includes an ADC converter for each digital channel.

For the project scope 10 analog channels and 14 digital channels have been used.

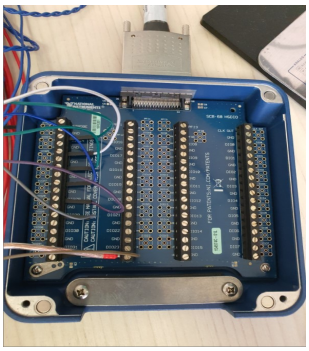
Figure 13: PXIe-7868R

I/O interfaces

To implement the interface communication between the FPGA and the EVCU, two separate terminal blocks have been used. The first one ("terminal block 1", fig. 14a) connected to the "*connector*0", dedicated to analog channels and digital output channels, and the other one ("terminal block 2, fig. 14b) connected to the "*connector*1", dedicated to the digital input channels.



(a) Terminal block 1



(b) Terminal block 2

Figure 14: Terminal blocks

Logic converters

Unfortunately, for what regard the digital channels, the simulator works with a 3.3V logic instead of the EVCU which works with a 5V logic.
For this reason, three logic adapter boards have been adopted, as seen in figure 15 .

They are composed by 4 channels each and each channel can be used as a double way converter. The reference 5V and 3.3V logic has been taken respectively from the 5V output pin of the terminal block 1 and the 3.3V output of a digital output pin of the FPGA, always fixed to the HIGH logical level, 3.3V in this case.

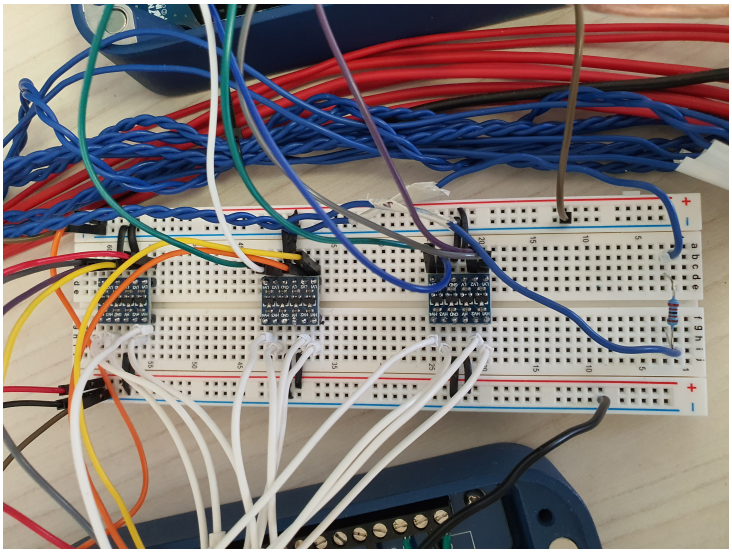


Figure 15: Logic level adapters

Unluckily, this approach leads to a little delay on PWM signals produced by the ECU. As shown in figure 16 , the calculated delay with an input PWM signal of 5V 4KHz is $\approx 3.6\mu s$, with an output voltage of $\approx 3.3V$.

The yellow line represents the output signal, instead the magenta line represents the input.

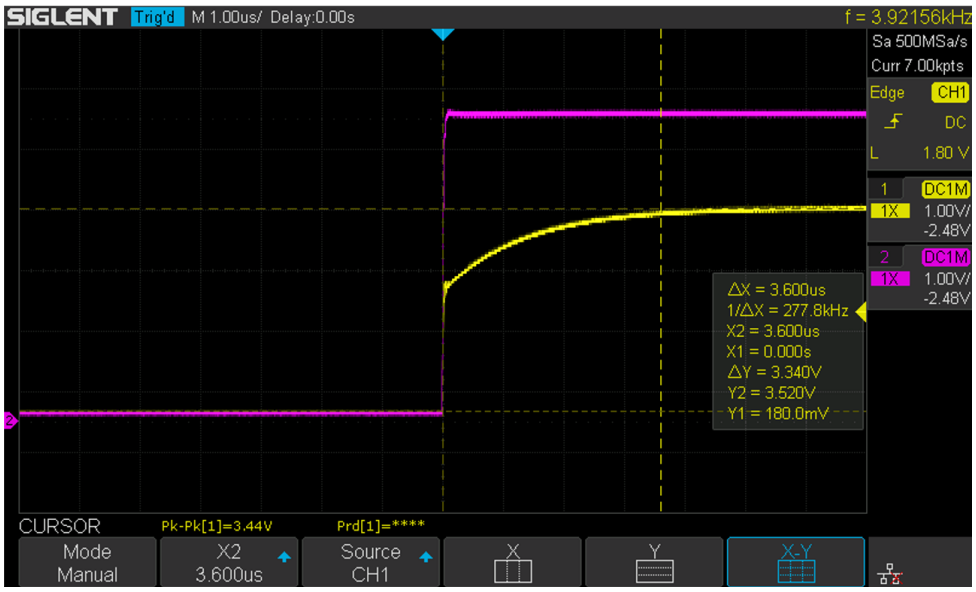


Figure 16: PWM converted signal

Even if this approach doesn't lead to particular problems, in order to have the best performance possible, a simple voltage divider could be used for each channel which goes from the EVCU to the HIL. The device is represented in figure 17.

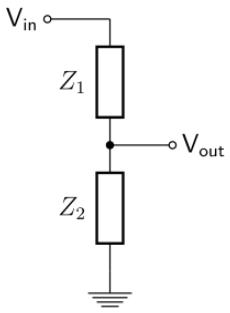


Figure 17: Voltage Divider

In this component, the V_{out} is a fraction of V_{in} depending on Z_1 and Z_2, following the law:

$$V_{out} = \frac{Z_2}{Z_1 + Z_2} * V_{in}$$

Where in this specific case $Z_1 = 2.2k\Omega$ and $Z_2 = 4.7k\Omega$

2.2 Software

Even the software could be distinguished into two categories. The software utilized in the simulator side where Veristand plays the main rule and the software utilized in the EVCU side as INCA.

2.2.1 Simulator side

Veristand

As specified on NI website [4], Veristand is an application software, developped by National Instrument, which helps developer engineers to configure I/O channels, data logging, stimulus generation, and host communication for NI real-time hardware.

It is also possible to import simulation models and control algorithms, to respond to events by configurable alarms and to enable test automation with software plug-ins. The monitor and the interaction with the application data, alarm states and system execution metrics is possible using a run-time editable user interface. It is also allowed the integration of third-party software, such as LabVIEW, ANSI C/C++, Python, and ASAM XIL to add custom functionality to VeriStand as needs.

Veristand helps test engineers to reduce the time needed to test the developing products with a wide range of functionality including configurable data acquisition, simulation model integration, test sequencing and logging.

In this case, Veristand is the core of the project, since it allows you to configure the FPGA and interact with the generated acquired signal in real-time, thanks to the run-time user interface. The configuration will be described in chapter 3.

In order to use Veristand to implement the thesis project, some software are required to be installed on the PXIe-8840. All the necessary software are visible in the figure 18 .

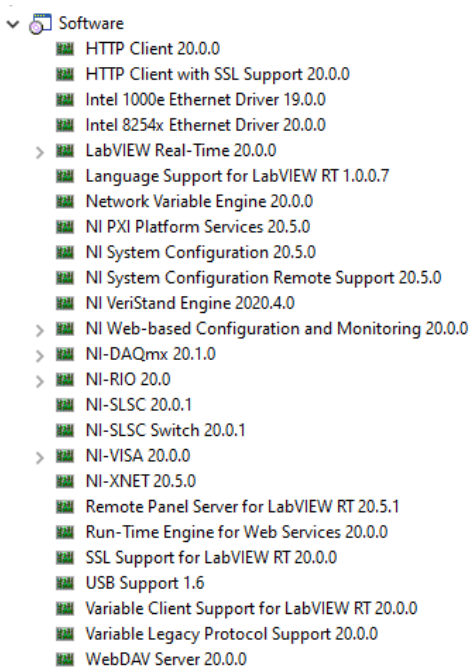


Figure 18: Needed software on real-time machine

OPAL-RT Add-On for Veristand

The OPAL-RT Power Electronics Add-On [5] for NI Veristand is a powerful FPGA-based simulation tool for Hardware-in-the-Loop (HIL) testing of controllers for power electronics applications, as described on OPAL-RT website.

It combines the performance of many floating-point FPGA solvers developed by OPAL-RT Technologies with an integrated workflow in Veristand to create a scalable and flexible HIL test bench.

The Power Electronics Add-On contains many features, including:

- the electric Hardware Solver (eHS) developed by OPAL-RT, which is a floating-point solver that allows the developer engineer to simulate an electric circuit on FPGA with no need to manually write the mathematical equations. A few example components to simulate could be converters, inverters and drives;

- FPGA-based Electric Machine Model solvers including the configurations of permanent magnet synchronous motor (PMSM) and induction motor (IM);

- a really small time-steps simulation of sub-microsecond;
- a user interfaces to map the electrical model inputs and outputs to high speed Input/Output, creating an interface between the simulated model and a real Device Under Test;
- for what regards open and/or closed loop testing a build-in signal generation engines (Sinewave, PWM, Sinusoidal PWM) in the FPGA design is present;
- very high compatibility with the NI VeriStand platform.

In order to use the OPAL-RT tool, some additional software are required, such as:

- Simscape Electrical Specialized Power Systems Simulink Blockset, for MATLAB 2011b-2019b;
- MATLAB Runtime R2016b (9.1);
- Java Runtime Environment (JRE) 8+ (1.8.0.151);
- Veristand 2018, 2019 or 2020.

2.2.2 EVCU side

INCA

INCA tools are used for ECU development and test as well as for validation and calibration of electronically controlled systems in the vehicle, on the test bench, or in a virtual environment on the PC, as in this case.

The tools offer a wide variety of functions including precalibration of function models on the PC, ECU flash programming, measurement data analysis, calibration data management, and automated optimization of ECU parameters. The generated calibration and measurement data can be processed and evaluated continuously.

In this case INCA has been used to flash the ECU with the firmware developed by the ECU producer and as user interface to directly command the ECU and read the

parameters to be compared with the ones of the simulator, to prove that everything is working correctly.

3 System configuration

3.1 Description

As mentioned above, the main goal of the thesis project is to close the loop between the NI based simulator and the EVCU in dyno mode.
In this mode the EVCU ignores CAN and others I/O signals and directly drives PMSM motor.
On the other side, the EVCU parameters are defined by Host PC via xETK.

Generally speaking, the PWM gate driver signals are read form the PXIe-7868R I/O and used to compute electric motor model operating at constant speed.
The PXIe-7868R is also responsible to fed back the Currents, the Voltages and the Resolver signals to the EVCU through the I/O interface.

After connecting and emulating all the signals needed from the EVCU, the HIL application performances have been analysed by actuating a torque through the EVCU user interface and monitoring the EVCU behaviour, checking if the simulated inverter and electric motor are correctly responding to the commanded gate switch, computing the correct currents and torque, as described in chapter 4.2.

To let all this possible the configurable FPGA OPAL-RT models will be used and properly modified.

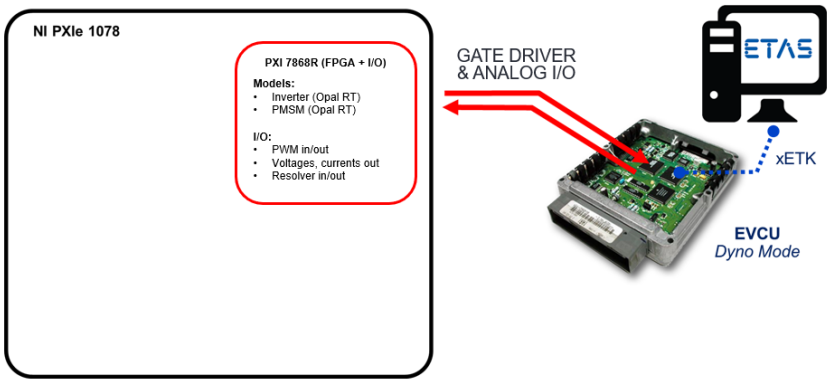


Figure 19: Close Loop Configuration

3.2 HW configuration

3.2.1 Physical connections

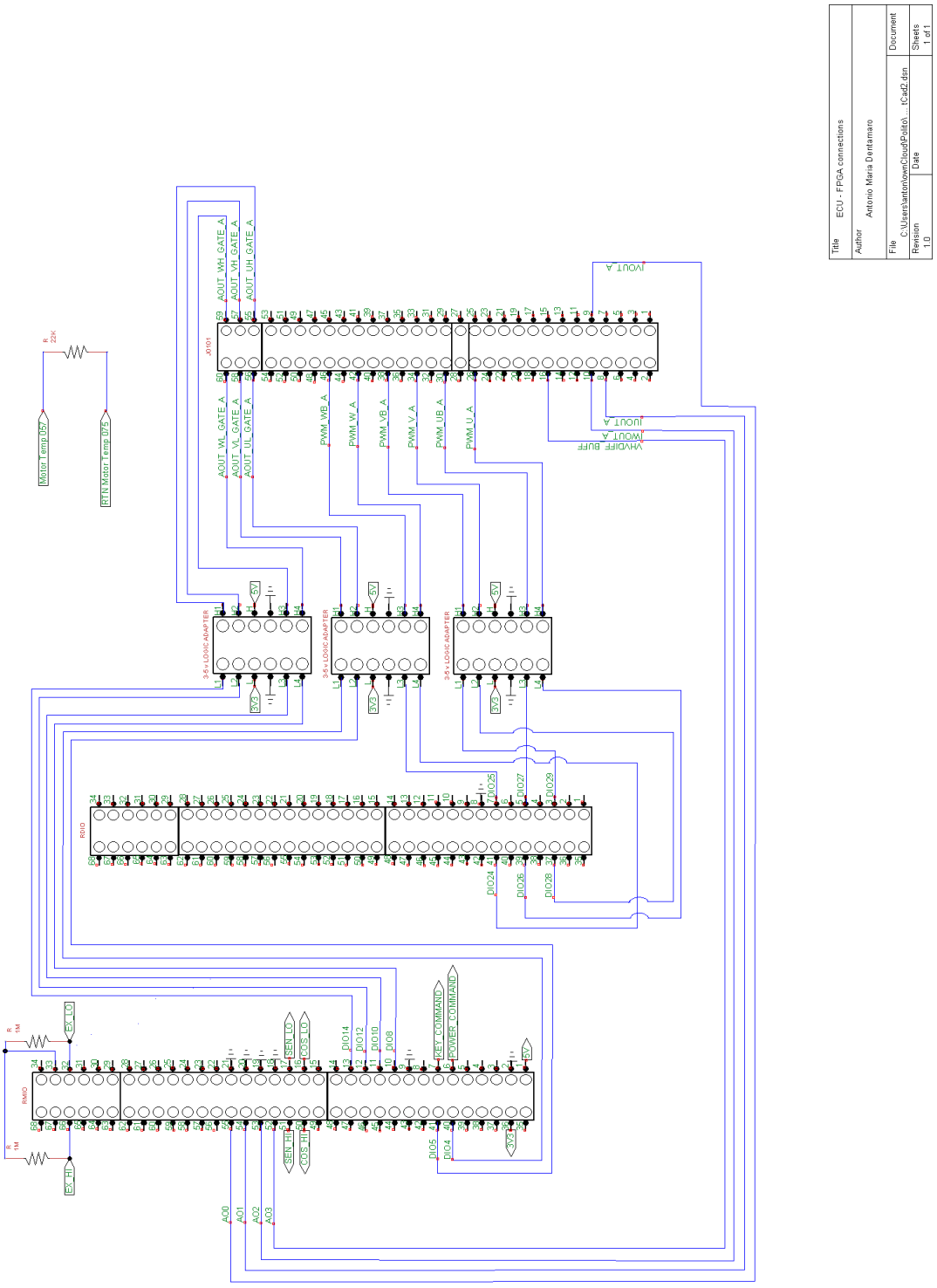


Figure 20: Scheme of physical connections of EVCU and FPGA

In the figure 20 is represented the electrical scheme of the links between the EVCU and the FPGA. The scheme is easier represented in figure 21.

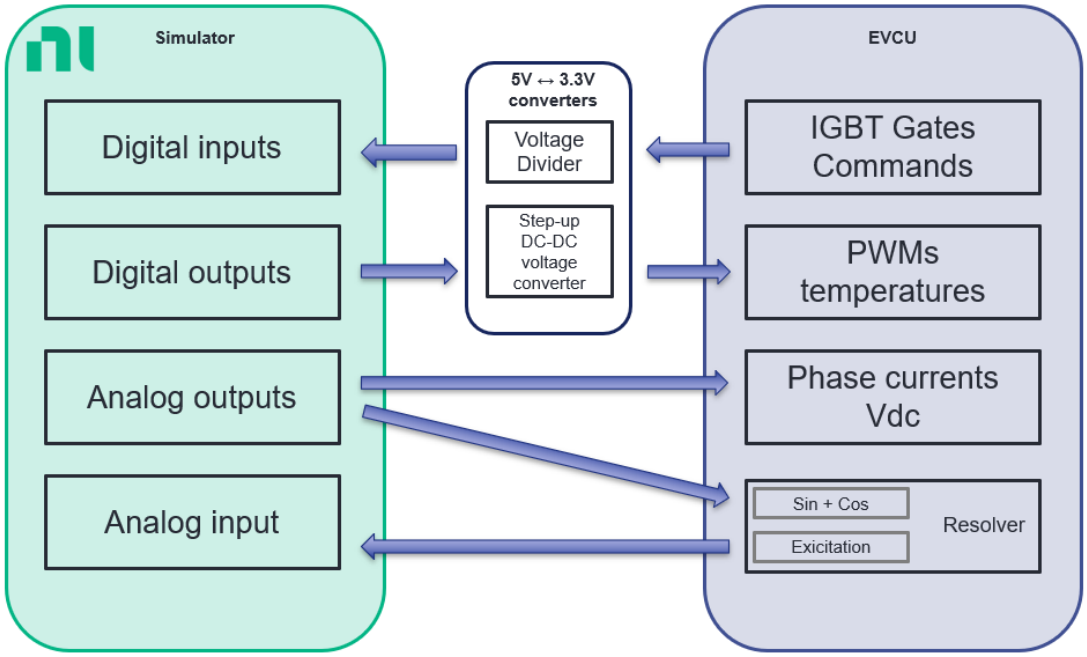


Figure 21: Scheme of physical connections of EVCU and FPGA

To be more accurate, the connectors RMIO and RDIO correspond to the terminal blocks of the FPGA. They are respectively the terminal blocks containing the analog I/O and the digital output ports and the terminal blocks containing the digital input ports. On the other side, the 3-5V logic adapters are represented near the connector J0101 of the EVCU.

On the J0101, from the pin 55 to the pin 60, there are the PWM input signals of the Gates temperatures. These pins have to work in 0-5V with frequency of 4kHz.
Then, the pins number 26,30,34,38,42,46 are the output PWM commands of the IGBTs, which are connected to the RDIO after logic conversion. They will actuate the six switches of the inverter.
Finally, the pins 8,9,10,16 are the feedback from the simulator to the EVCU of the phase currents and the DC-Link voltage. This feedback is necessary for the EVCU in order to properly set the duty cycle of the IGBTs that will actuate the inverter switches.

Instead, the motor temperature is read from a simple NTC pin, connected directly to

a $22K\Omega$ resistor. The choice of this resistor value has been taken in order to maintain the motor temperature more or less at 44 °C.

On the RMIO, the pins 32,33,66 are responsible to read the resolver excitation signal from the ECU. These pins are connected considering the signal as a floating differential signal, according to the following scheme (fig 22) took from the NI PXIe-7868 data sheet.

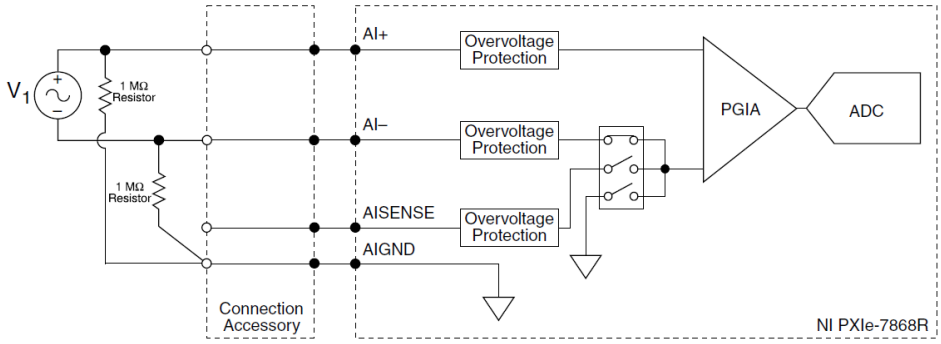


Figure 22: Resolver Excitation signal connection scheme

The sine and cosine signals of the resolver are instead outputted respectively on the pins 51,17 and 50,16, respecting the polarities.
All these signals are waveform in the range from -12V to 12V.

In addition, even on the RMIO, there are the analog output pins from 52 to 55 of the phase currents and DC-link voltage as feedback coming from the simulated model to the EVCU.
The currents have an offset of 2.5V and they are in the range 0-5V as the voltage.
The digital output pins from 10 to 13 and 40,41 are the 4kHz PWM signals responsible to simulate the IGBTs temperatures for the EVCU.
Finally, there are the two digital command of the two relays in figure 23, they are digitally set from the user interface and they directly go to the two Arduino Mega Analog Input pins.

In figure 23, there is the scheme of the relay board connected to the Arduino Mega and the power supply.

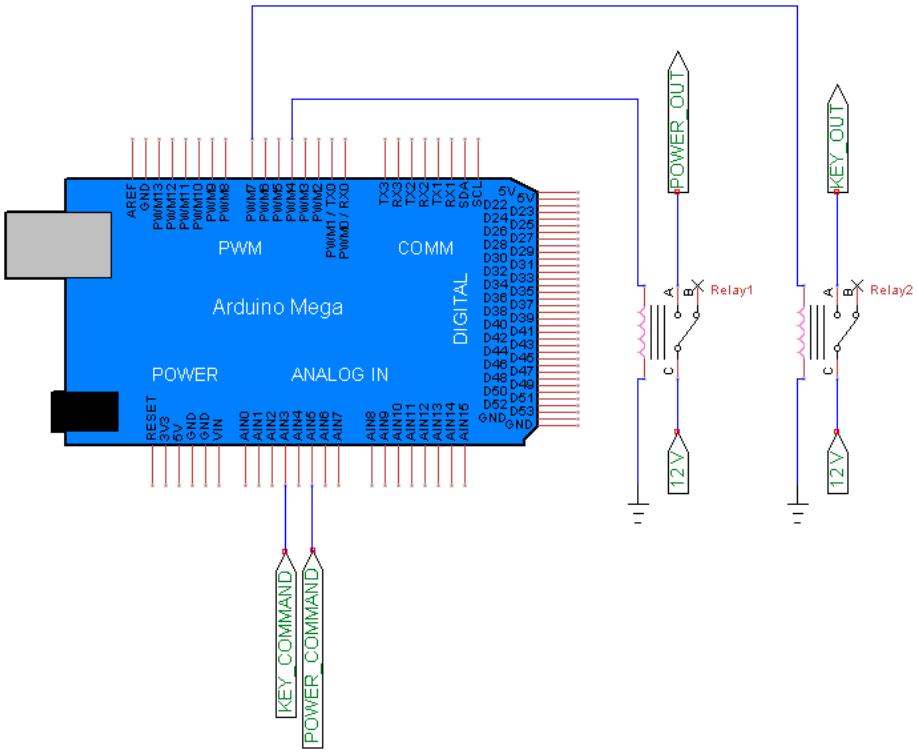


Figure 23: Scheme of physical connection of Ardunino Mega and relay board

The Arduino Mega system works as follows:

1. The user set the two virtual buttons of the "power" and the "key" to ON/OFF from Veristand;

2. The command of the "power" and the "key" are read from the Arduino Mega through the analog input pins respectively the AIN5 and the AIN3. This approach has been chosen in order to bypass the logic level incompatibility issue between the 5V logic Arduino Mega and the 3.3V logic of the FPGA;

3. the relays are piloted by the digital pins 4 and 7, according to the simple logic described in the following table, where "power_in" and "key_in" are the analog inputs and the "power_out" and "key_out" are the digital output pins;

4. the logic behavior is represented in the following table. For example, if the two command of the relay are in "on" state they close the contacts and they give 12V

power to the EVCU and to the KEY. Note that, when only the KEY is on, as in the second case, nothing happen, since the POWER is off.

power_in	key_in	power_out	key_out
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

Notice that for what regard the analog input pins, shown in the previews table, the value "1" corresponds to a read voltage greater than 3V, instead the "0" corresponds to the opposite values.

3.2.2 Veristand mapping

On Veristand, the mapping is mandatory to let the simulator properly works.

In figure 24 the mapping is shown. In detail:

- A user channel has been created to set the source V_DC_Link, which corresponds to the inverter voltage (Vdc);
- Vdc is connected to the Analog Output AO03;
- the Digital Input pins from DI24 to DI29 are connected to the six switches of the inverter;
- the Sine and Cosine output of the resolver are routed to the Analog Output AO04 and AO05;
- the phase currents are outputted from the Analog Output pins AO00, AO01 and AO02.

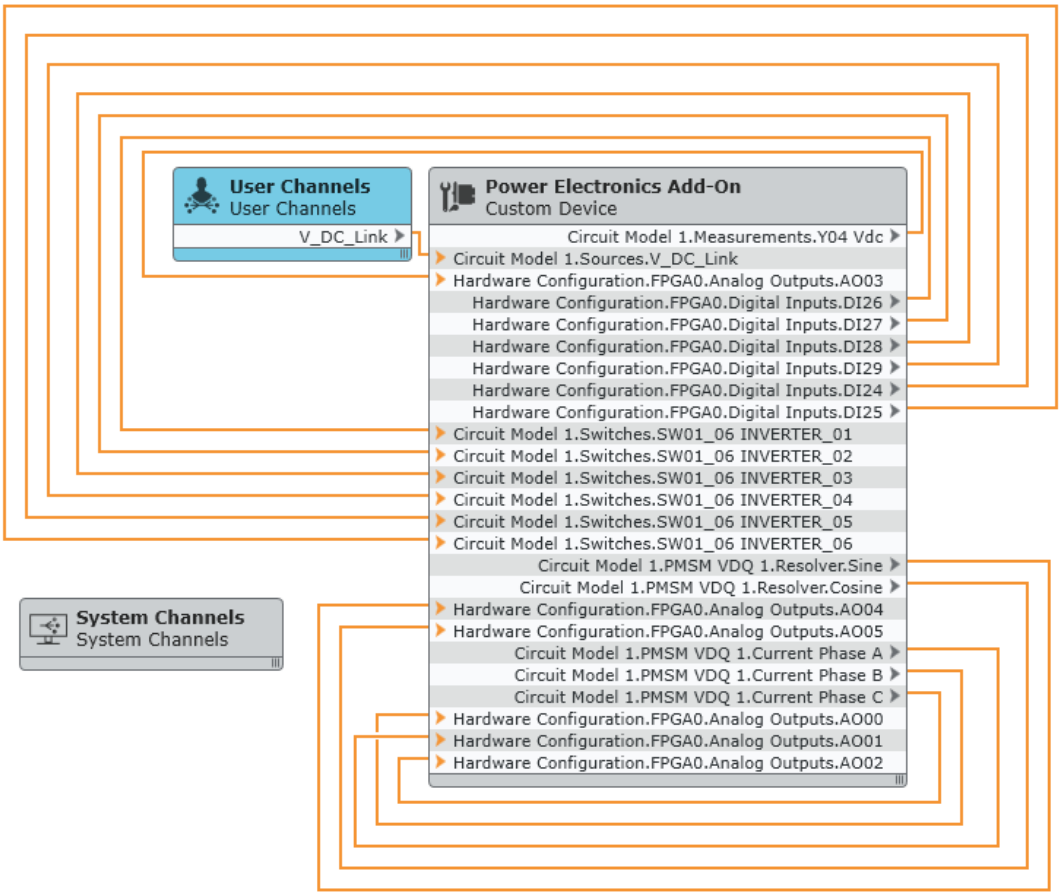


Figure 24: Veristand mapping

3.3 SW configuration

3.3.1 Model

For the purpose of this thesis, a PMSM motor has been configured through Veristand thanks to the OPAL-RT add-on.

Considering the OPAL-RT PMSM VDQ model, it results easy to configure the model just entering the real parameters, such as the number of pole pares, inductance values, phases resistances and so on. Then, it is just required to map all the I/O of the model as in figure 24.

In the thesis case, the Electromagnetic model has been parameterized to the vehicle under test specs using constant values for inductance, phase resistance and permanent magnet flux linkage.

The model operates in speed-mode imposing revolution speed from Veristand user interface.

The workflow of the model is represented in fig 25 , where it results clear that the model is subdivided into four sub-models:

- circuit Model;
- electromagnetic model;
- mechanical model;
- resolver model.

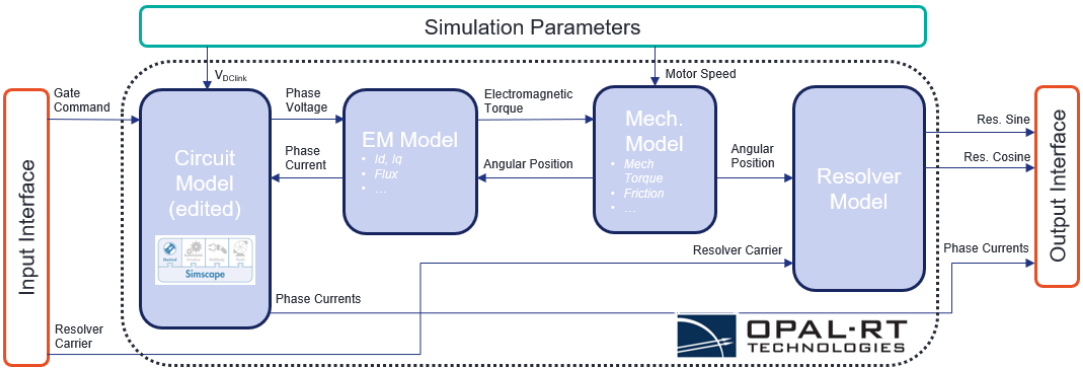


Figure 25: Model workflow

In order to get the required type of motor model, the circuit model in SimScape was edited to use a constant DC-link voltage source, starting from the original model in figure 26.

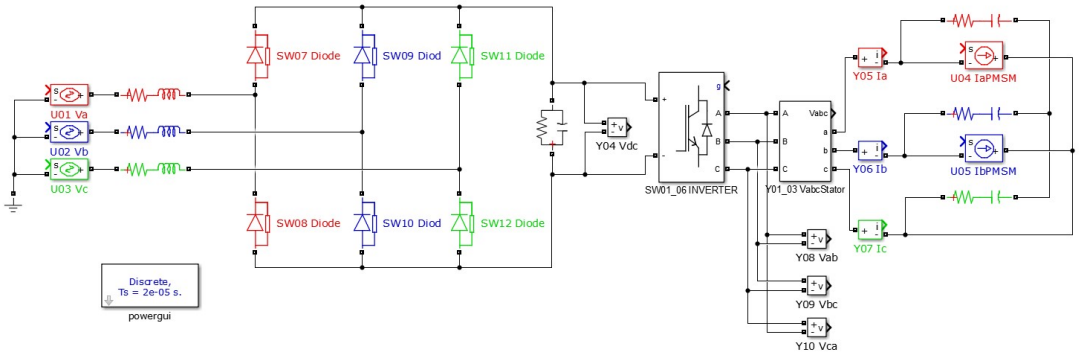


Figure 26: Original OPAL-RT model

The original model was composed by:

- grid-tied;
- three-phase diode rectifier for charging the DC Link;
- two-Level, three-phase universal bridge;
- the PMSM machine model.

In this configuration the bridge is electrically connected to the PMSM machine model through the project configuration.
A three phase signal is generated using the built-in FPGA signal generators to simulate the three phase voltages Va, Vb, Vc AC sources in the electrical circuit.
The universal bridge gates "SW01_06 INVERTER" are controlled using the signals coming from the on-board SPWM generators.
The PMSM calculated currents are fed back to the universal bridge through the current sources "U04 IaPMSM" and "U05 IbPMSM".

However, the model has been modified to better simulate the required system.
In particular, the grid-tied and the three-phase diode rectifier have been substituted by a DC power source called V_DC_Link, as shown in figure 27. In this way the DC_Link is directly set by the user, simulating the vehicle battery.

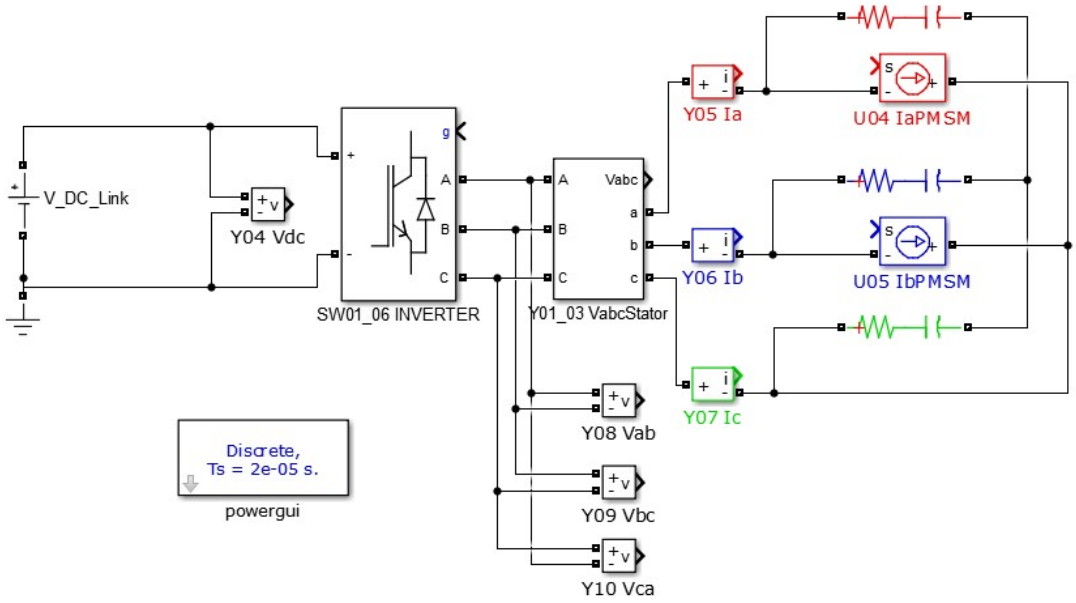


Figure 27: Modified OPAL-RT model

3.3.2 Simulator settings

To let everything work properly, obtained the desired motor model, as described in the previous subsection, the following steps have been performed. They are necessary to set the correct parameters of the simulated motor model.

1. Chose the Circuit Model file path, of the previous subsection, and set the Timestep of this circuit model to 2.5E-7 s;

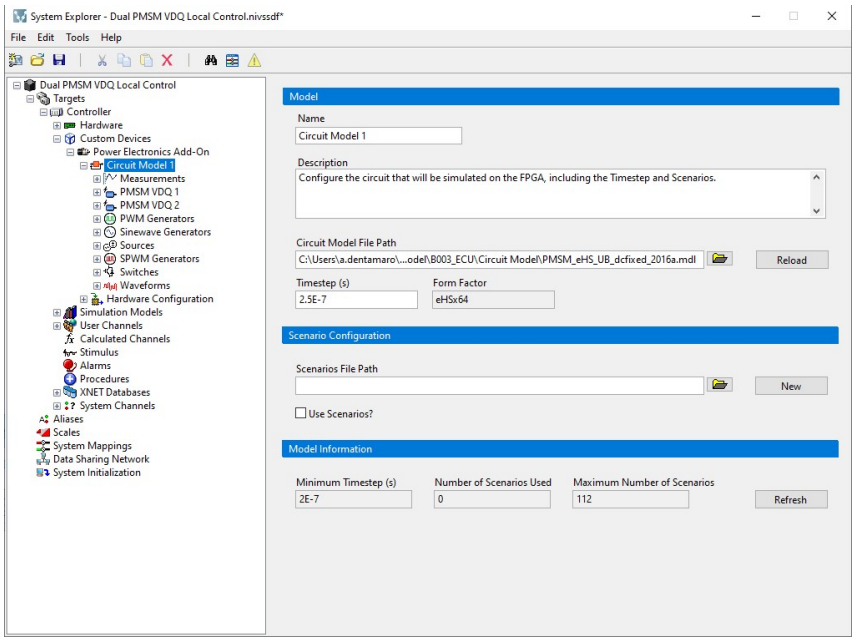


Figure 28: Circuit Model file path

2. In the "PMSM VDQ 1" section, there are the PMSM motor parameters.

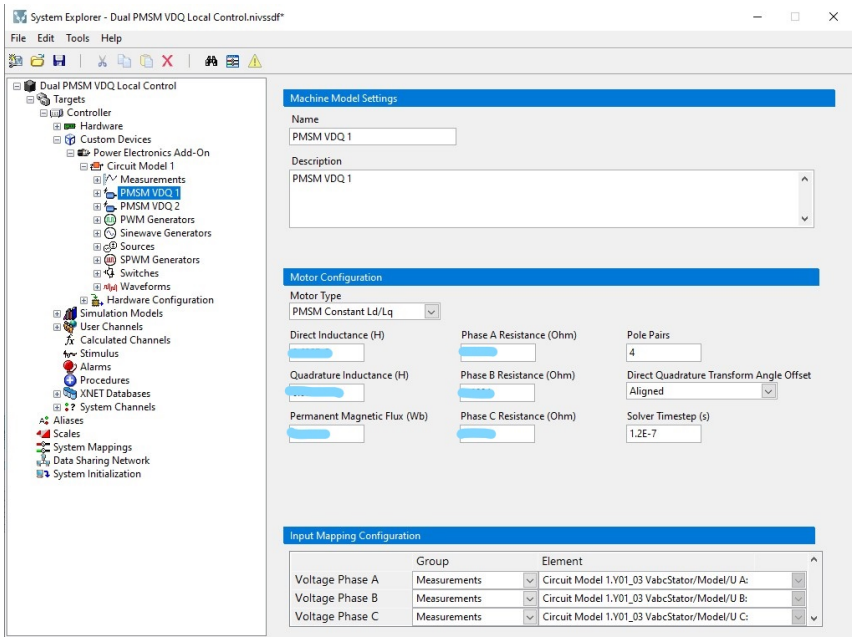


Figure 29: Motor configuration

- In "Motor Configuration", from the drop down menu, "PMSM Constant Ld/Lq" ha been chosen as Motor Type;
- The Direct Inductance, the Quadrature Inductance, the Permanent Magnetic

Flux and the three phase resistances are confidential parameters since they have been set referring to the manufacturer specs;

- The Pole Pairs has been set to 4;
- The Direct Quadrature Transform Angle Offset has been set to Aligned;
- The Solver Timestep has been set to 1.2E-7 s.

3. In the "Resolver" section, the resolver has been configured.

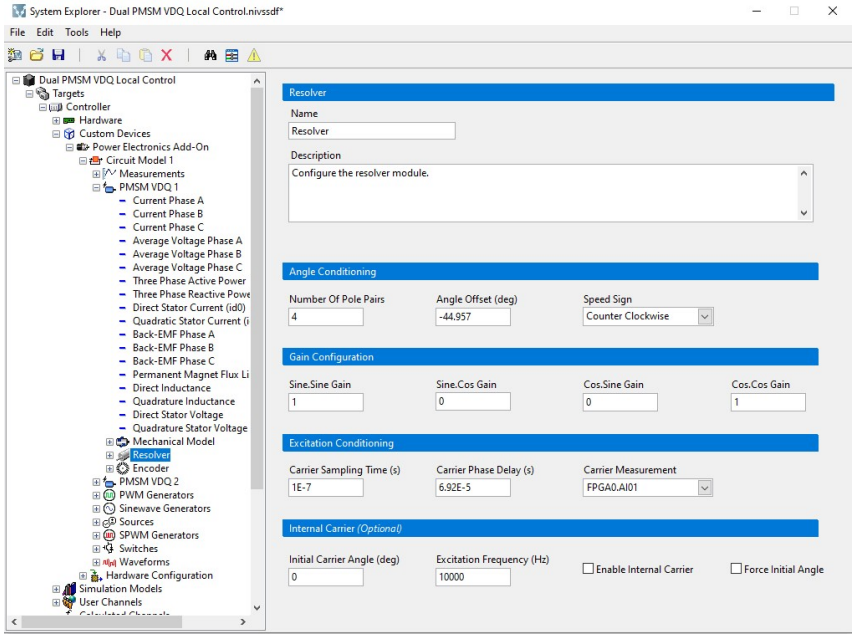


Figure 30: Resolver configuration

- Number of Pole Pairs has been set to 4;
- After some experimental tests, an Angle Offset equal to -44.957° has been introduced to align the angular position of the simulated model with the angular position estimated from the EVCU;
- The Speed Sign has been set to Counter Clockwise;
- The carrier sampling Time has been set to 1E-7 s;
- A Carrier Phase Delay equal to 6.92E-5 s has been introduced after experimental test to synchronize the EVCU and the simulator as described in the Troubleshooting chapter;

- The Carrier Measurement is don on the Analog Input pin 1 (FPGA0.AI01).

4. In the "Sources" section, the phase currents and the V_DC_Link source have been set as in the figure 31 .

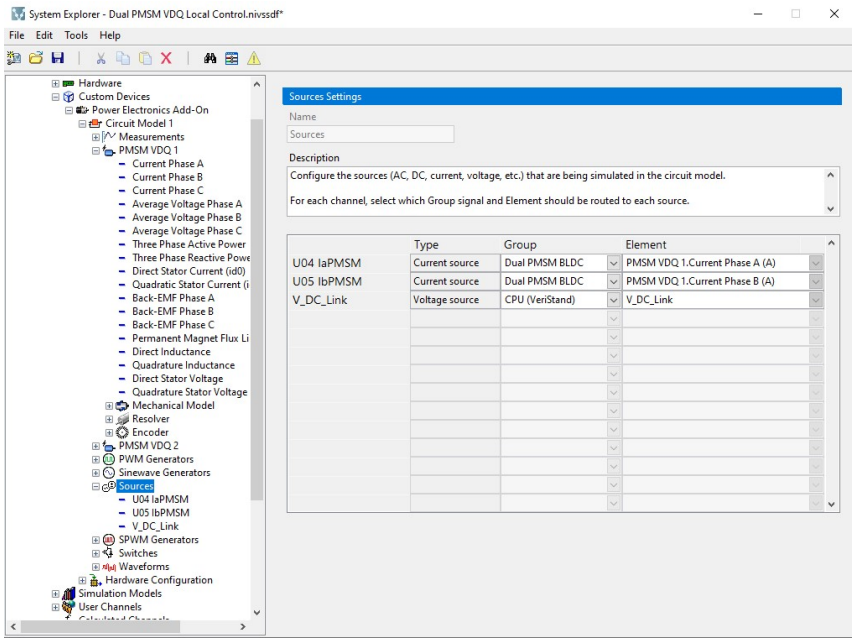


Figure 31: Sources configuration

5. In the "PWM Generator" section, the PWM signals have been configured in order to set the IGBTs temperature to $\approx 16°C$.

   In detail, the frequency of the PWM Generator 0 has been set to 4 KHz and the Duty Cicle to 47%. Then, the Active and Inactive Dead Time have been set to 0s.

6. In the "Switches" section, the switches/gates simulated in the circuit model have been configured. Each IGBT has been mapped to the corresponding Digital Input signal coming from the EVCU, through the Digital Input pins of the FPGA.

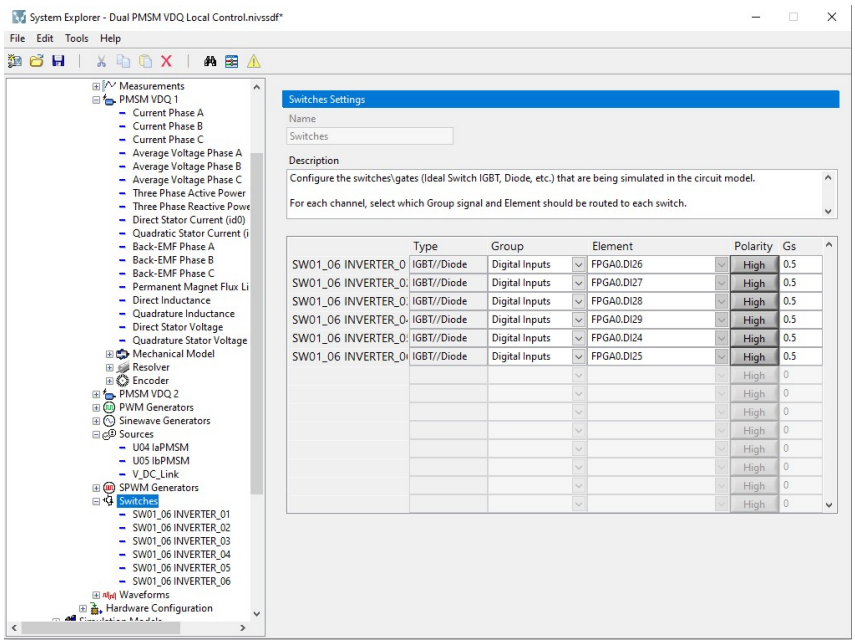


Figure 32: Switches configuration

The eHS Solver, as described on the OPAL-RT website [5], uses Modified Nodal Analysis to generate a conductance matrix that, when solved, returns the voltage at each node of the circuit and the current in each branch. This matrix is generated independently from the switches states, thus, it does not need to be recomputed when a switch is opened or closed during the simulation. This behaviour could be achieved representing each switch component as an impedance thanks to the implementation of the Pejovic method. In particular a close switch is represented as an inductor and an open switch is represented as a capacitor.

Thanks to the Backwards Euler discretization, the inductors and the capacitors could be further simplified, obtaining the equivalent circuits represented by a current source with a shunt resistance as shown in figure 33.

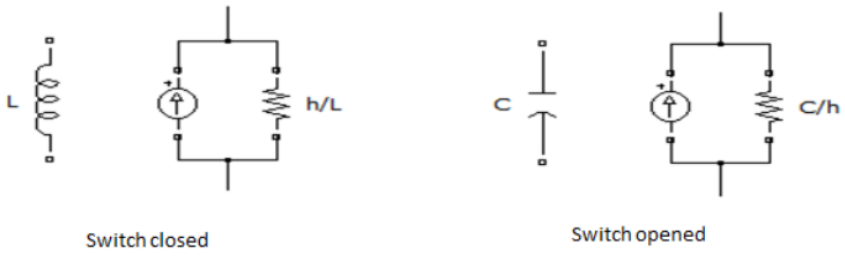


Figure 33: Switches representation

The equation $Gs = h/L = C/h = 0.5$, which define the Switch Conductance (Gs), must remain true in order to let the conductance matrix to remain constant throughout switching events.

7. In the "Analog Output" section, following the relation $AO = Gain * AO + Offset$, all the analog signals to be generated by the FPGA have been configured.

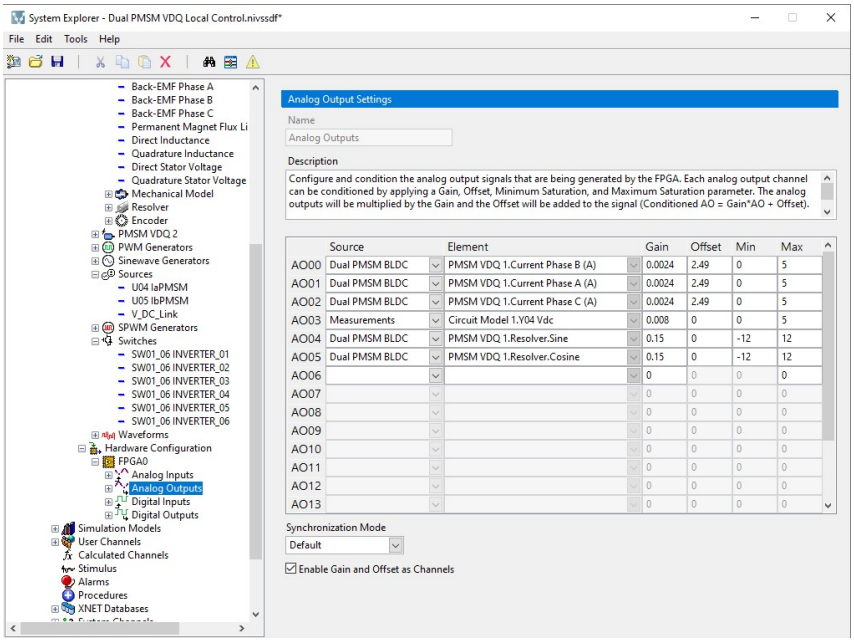


Figure 34: Analog Output configuration

- AO00, AO01 and AO02 are the phase currents that have been limited in 0 to 5 V range. They follow the relation $AO = 0.0024 * AO + 2.49$;
- AO03 corresponds to the Vdc, limited in 0 to 5 V range, following the relation $AO = 0.008 * AO + 0$;

- AO04 and AO05 are respectively the Resolver Sine and Cosine, oscillating in -12 to 12 V range. They follow the relation $AO = 0.15 * AO + 0$.

8. In the "Digital Output" section, the digital output signals generated by the FPGA have been configured.

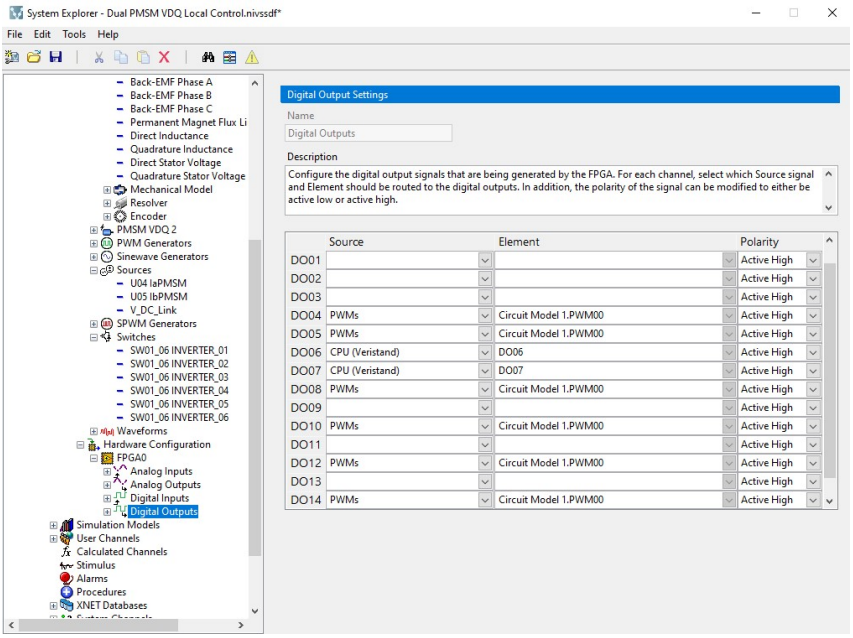


Figure 35: Digital Output configuration

- DO00 has been used as 3.3V reference for the 3.3-5V Logic Adapters. For this reason its polarity is Active Low;
- DO04, DO05, DO08, DO10, DO12 and DO14 are mapped to the PWM signals which control the IGBTs temperatures. Thus, they are Active High.
- DO05 and DO06 are the pins used to actuate the Power and the Key Relays.

3.3.3 Simulator and EVCU user interfaces

For the purpose of controlling the actuation of speed, relays commands, torque and monitoring the simulation at run-time, a well configured user interfaces is necessary.

The simulator user interface in figure 36 is described below:

- On the left side, there are all the simulated signals of the model;

- In the centre, there are the user-defined speed command, the speedometer showing the actual motor speed, the two buttons to actuate the Power and the Key relays and the read frequencies and duty cycles of the gates switches;
- On the right side, there are the plots of the simulated phase currents, phase voltages, direct and quadratic stator currents and the total torque.

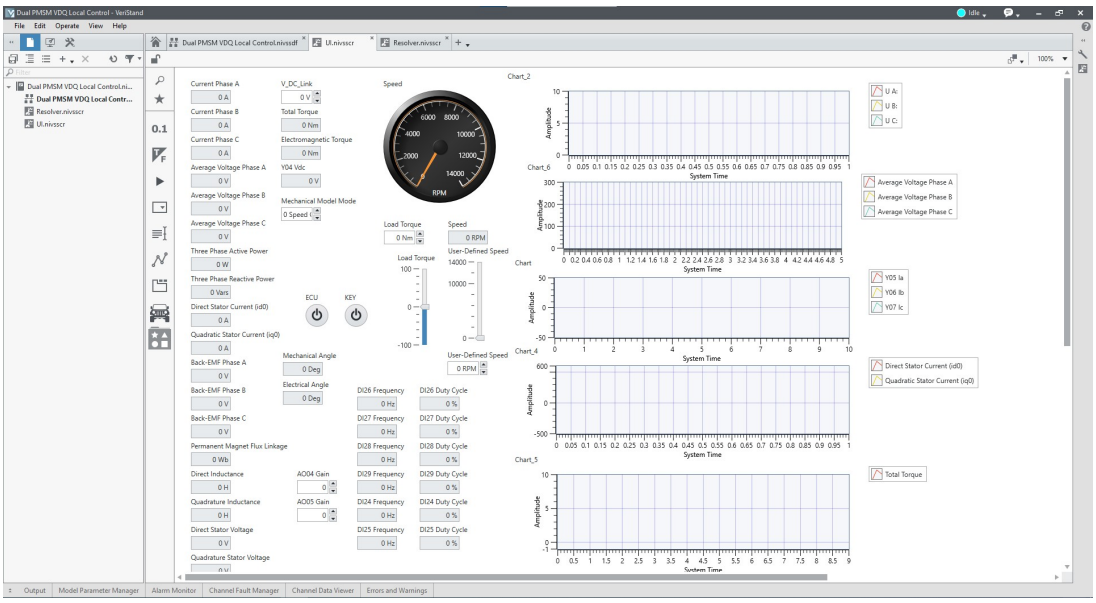


Figure 36: Veristand User Interface

On the EVCU side, no images can be shown in the thesis for confidentiality reason. However, it is composed by all the needed tool in a calibration task.

Both the user interfaces are easily modifiable even at run-time, according to the necessity.

4 Observations and results

In this chapter, the troubleshooting of a few issues, come across the development of the project, will be described. Then, the final test performs will be illustrated and discussed.

4.1 Troubleshooting

During the development of the thesis project, some troubleshooting on a few issues have been done.

Logic adaptation

With no doubt the main issue that has been solved, as described in the previous chapter, is the logic adaptation between the simulator and the EVCU.

The simulator works with a 3.3V logic than the EVCU works with a 5V logic. For this reason all the digital output signals, coming from the EVCU to the simulator, pass through a voltage divider. The other way around, the digital signals, coming from the simulator to the EVCU, pass through a step-up DC-DC voltage adapter.

Resolver

An other mandatory adaptation regards the resolver. It has been necessary to add an angle offset of -44.957°.

This particular value has been chosen based on experimental results. The experiment has been the comparison of the angular position coming from the mechanical model of the motor simulated on the HIL and the angular position estimated from the EVCU based on the sine and cosine signals coming from the simulator.
This two values have been tuned, through this offset, until they turned out to be more or less the same. In this case the angle offset of -44.957° is the best value to choose.

Temperatures

To let the EVCU work properly, some tuning have been done for the IGBTs temperatures and the motor temperature.

To be more specific, the motor temperature need a NTC to be set. In this case a resistor with a resistance value of $22K\Omega$ has been connected between two dedicated pins of the EVCU, as shown in figure 20. Thanks to this resistance value the motor temperature has been stabilized to ≈ 44.75°C.

On the other side, the IGBTs temperatures has been tuned setting the duty-cycle of the dedicated PWM signals, coming from the simulator, to 47%. In this way, a temperature of $\approx 15.5°$ has been reached for any of the IGBTs.

These two temperatures have not a particular meaning for the scope of this thesis project. They have been chosen just to let the EVCU work without problems.

General errors

During the experiment, some electrical fault on the EVCU have been got. These faults are mainly referred to the resolver and the Vdc signals coming from the simulator.

To avoid some electrical damage, all the signals have been checked one by one using an external oscilloscope. By this check, it has come out that all the signals respect the specifics given by the manufacturer.
For this reason, the faults have been momentarily bypassed. Meanwhile, the faults are under investigation by the manufacturer's engineers.

4.2 Validation

The following observations have been done based on the acquisitions got following the steps below. Than, the Torque, the Direct and the Quadrature currents have been compared in a few graphs. Note that for confidentiality reason, all the parameters had been normalized with the maximum value.

1. The motor speed will be set between 0 to the maximum speed, denoted as V_{max}. Likewise, a torque will be actuated, by the EVCU side, starting from the maximum torque, denoted as T_{max}, continuing with 85%, 57% and 28% of T_{max};

2. The EVCU will generate six signals to actuate the IGBT switches simulated by the HIL. Likewise, it will generate the resolver excitation signal, constantly sent by the EVCU to the simulator by a properly configured analog I/O;

3. The HIL simulator will generate the relative phase currents and phase voltages, that will be fed back to the EVCU, by properly set I/O interfaces, in addition with the DC_Link voltage, the sine and cosine signals of the resolver and other necessary signals, such as six PWMs signals of the IGBTs temperature;

The following acquisitions in figure 38, 39, 40 and 41 have been obtained setting a target torque on the EVCU and imposing the desired speed on the Simulator side.

In particular, the motor speed has been set from 0 to MAX Speed with a step-size of 7% of V_{max}. Meanwhile, the torque has been set from INCA to T_{max}, 85%, 57% and 28% of T_{max}.

The acquisition points has been represented in figure 37 .

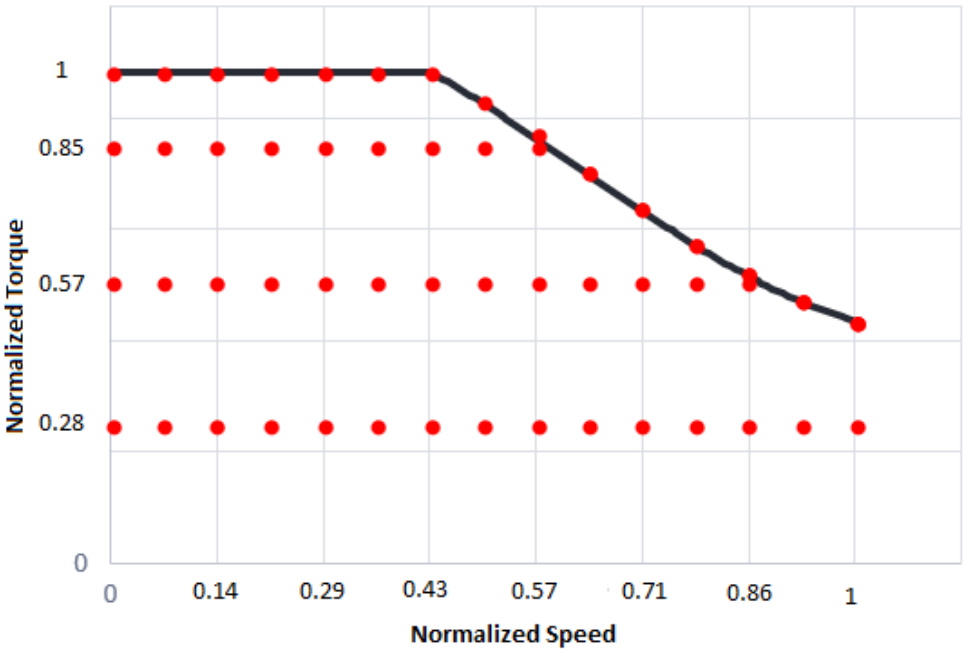


Figure 37: Acquisition points

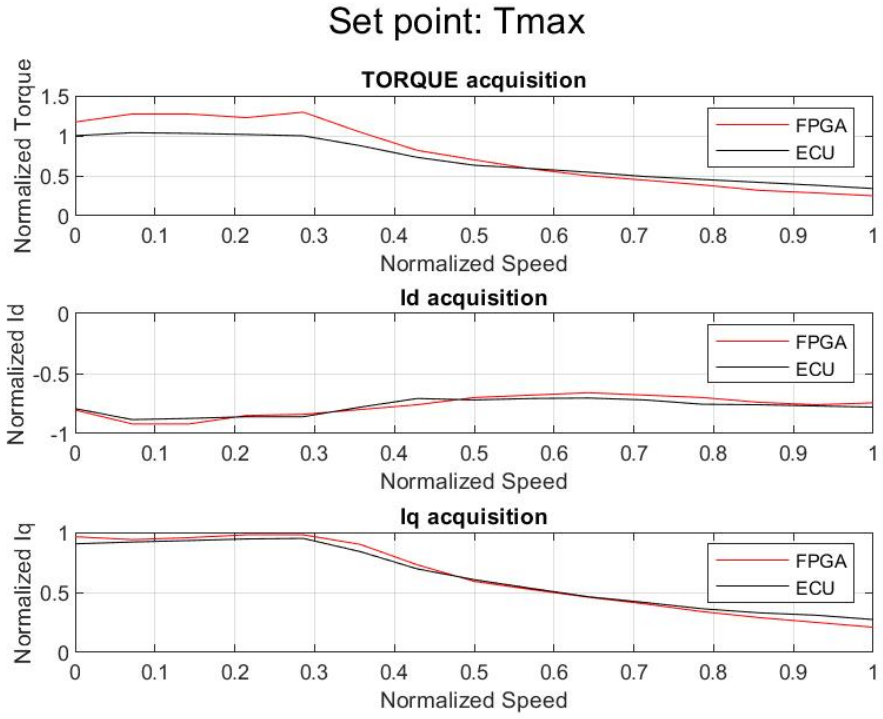


Figure 38: Torque, Id, Iq with the Torque set-point of T_{max}

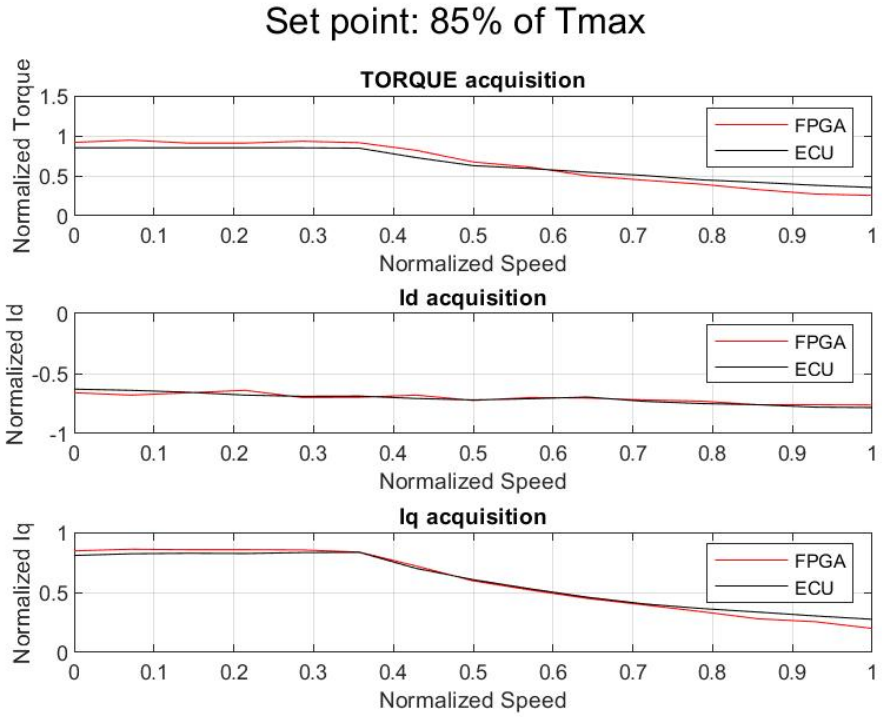


Figure 39: Torque, Id, Iq with the Torque set-point of 85% of T_{max}

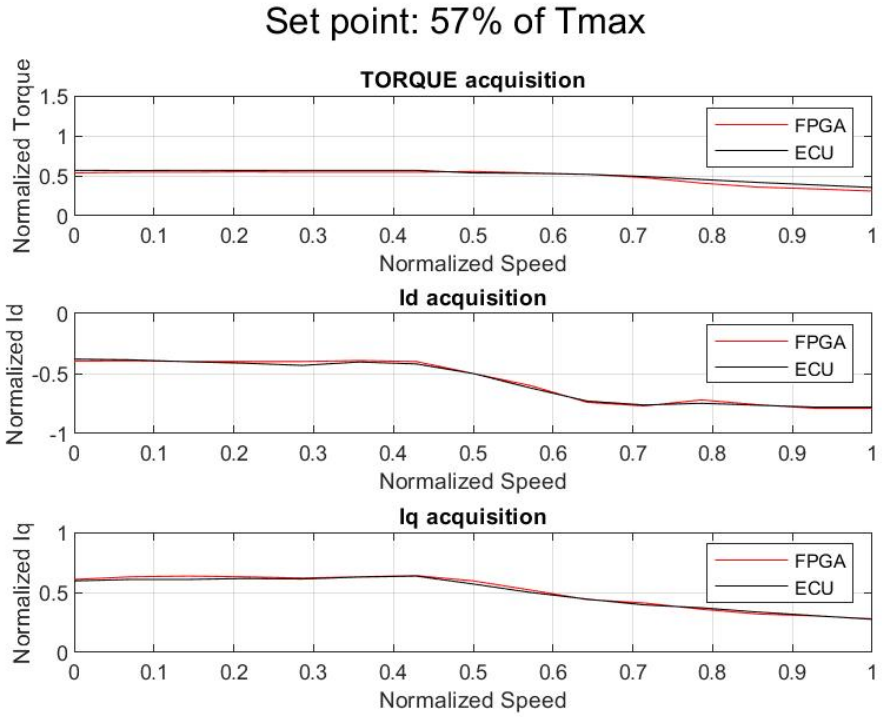


Figure 40: Torque, Id, Iq with the Torque set-point of 57% of T_{max}

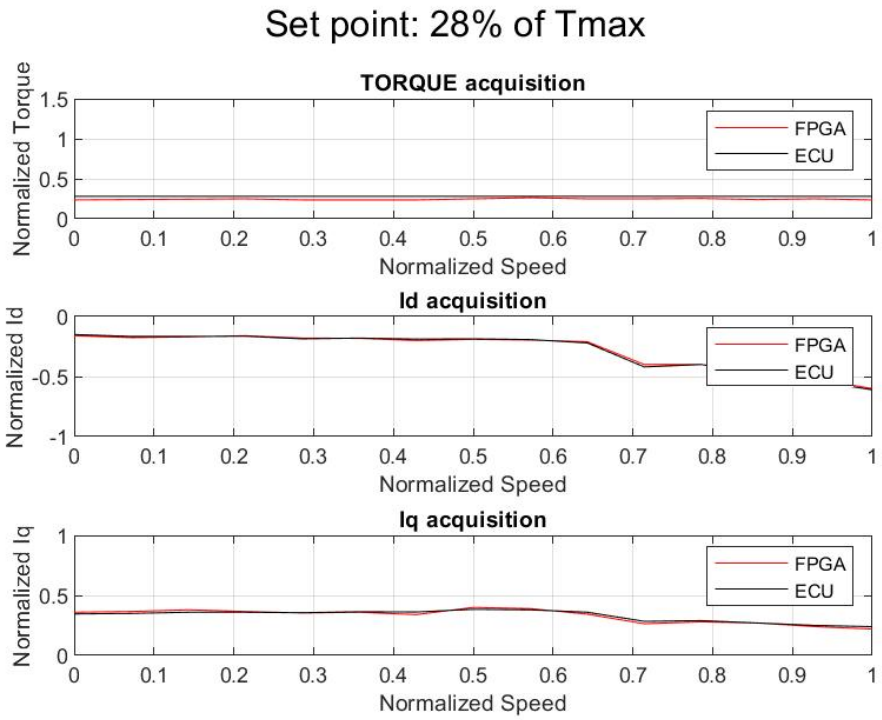


Figure 41: Torque, Id, Iq with the Torque set-point of 28% of T_{max}

In this way, the EVCU estimated torque, direct and quadrature currents has been compared with the model outputs on different operating points over the motor map.

The agreement between EVCU and simulated model is more than acceptable considering the quite simple modelling approach (constant Ld and Lq). As a future work, the adoption of a table-based model could be evaluated in order to improve simulation fidelity.

The agreement could be easier observed in the following scatter plots of the Torque, Id and Iq represented in figure 42, 43, and 44.

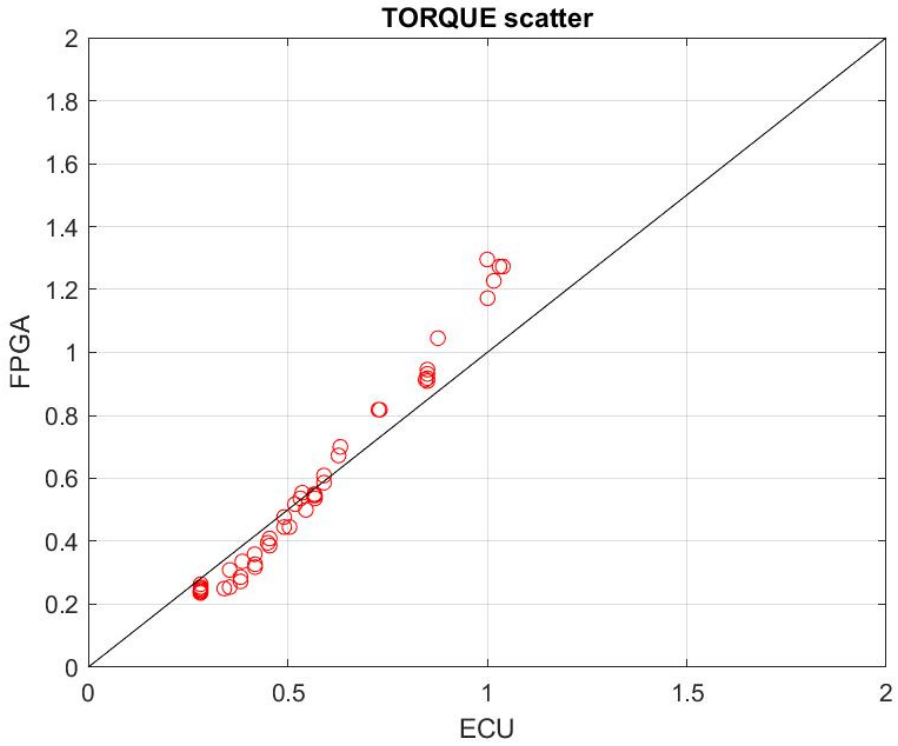


Figure 42: Torque scatter plot

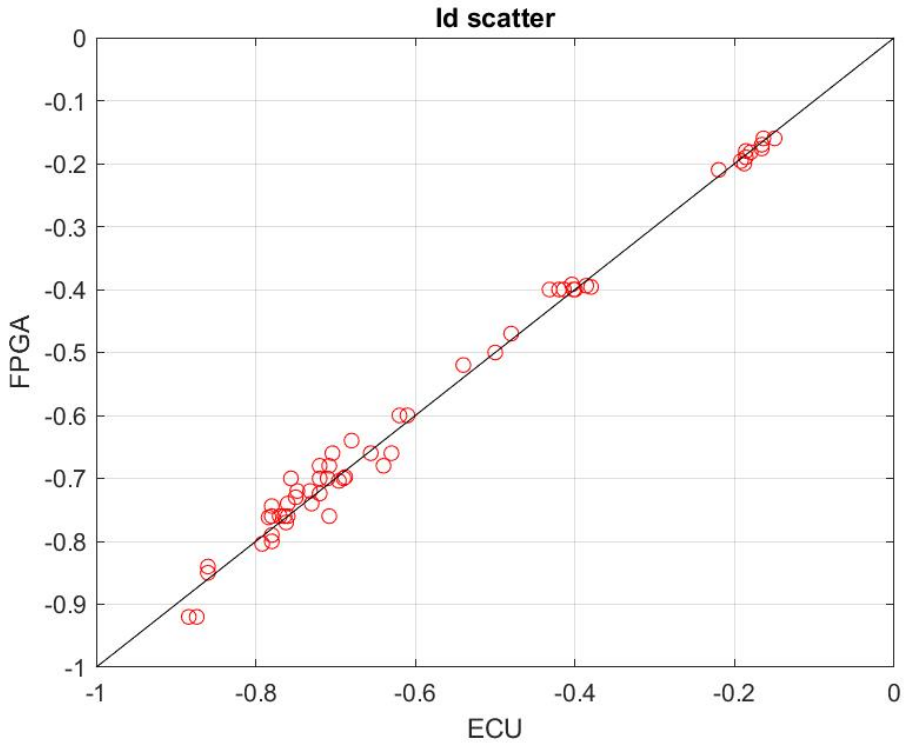


Figure 43: Id scatter plot

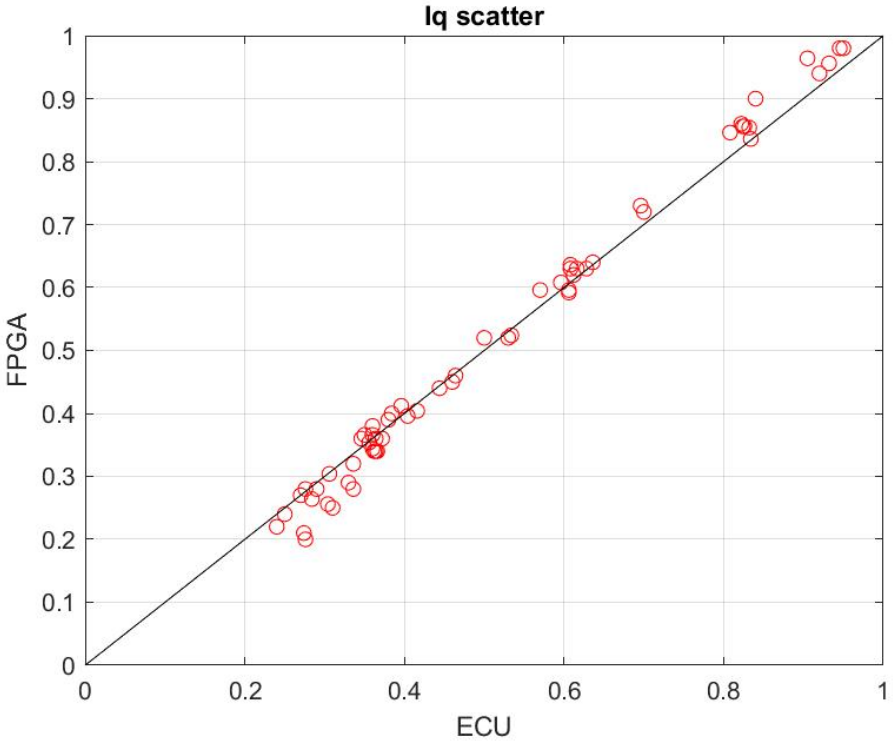


Figure 44: Iq scatter plot

In the above figure, it is easy to observe that the red circles, representing the acquired data, are really close to the bisector of the quadrant. This means that the simulated data of the HIL simulator are almost equal to the estimated data of the EVCU.

Thus, the simulator parameters have been correctly set as close as possible to the real system parameters.

5 Full vehicle model integration

In addition to the thesis project a part of the vehicle model has been added to the PMSM motor model.

It consists in three sub-models which represent the vehicle dynamics, the driveline and a virtual driver who actuate a torque simulating an action on the accelerator and the brake pedals.

Moreover, in order to actuate the torque by the driver simulated model directly from the HIL, the CAN network has been added to the system, through which a particular frame has been sent to the ECU from the simulator containing a payload carrying the desired torque command.

5.1 Necessary hardware

As mentioned before, there was the necessity to add a board to manage the CAN network. For this purpose the PXIe-8510 from NI, shown in figure 45, has been connected to the NI chassis.

Figure 45: PXIe-8510

The board is a CAN / LIN interface which can be configured through NI-XNET. This board is perfect to be use in real-time applications, as in this case, in which it has to exchange many CAN frames as fast as possible.

Thanks to the integrated CPU, the board is able to directly exchange CAN frames between the user application and the network bypassing the CPU of the HIL. This leads to a very low latency and a better management of the HIL's processor, allowing the latter to elaborate more complex processes.

5.2 Setup

The vehicle model is reserved and can't be shown in the thesis.

However, a simple scheme which represents it is illustrated in figure 46 .

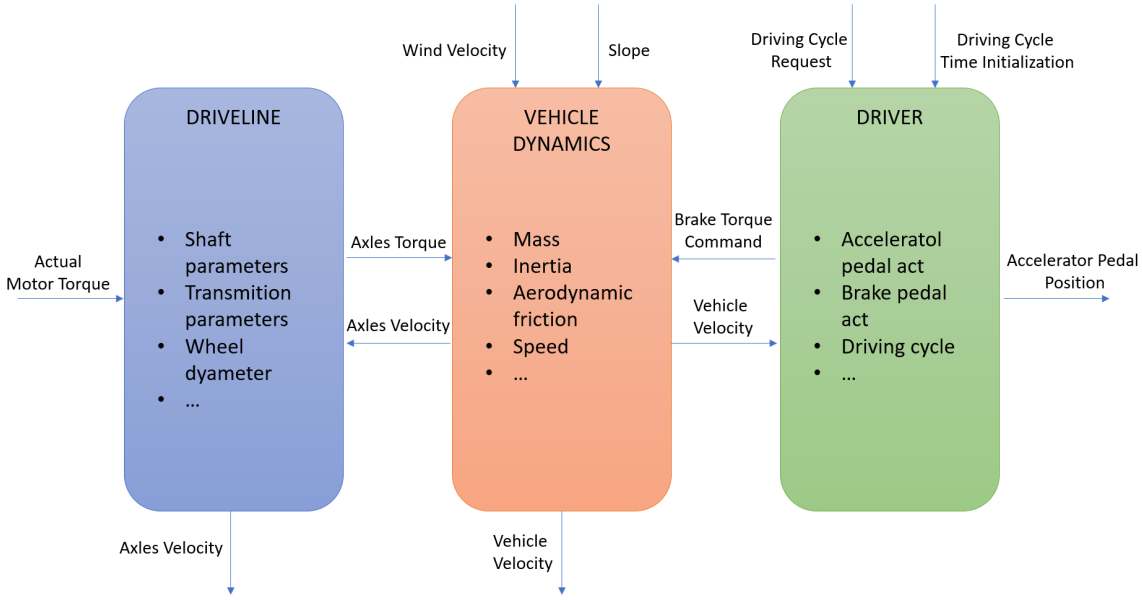


Figure 46: Vehicle Model Scheme

The driveline contains all the transmission parameters useful to compute the axle velocity and the necessary torque to move the wheels.

To let it works it need the actual Motor torque coming directly from the simulator, computed from the six IGBT switches duty-cycles actuated from the EVCU.

The vehicle dynamics block contains all the vehicle parameters, such as the vehicle

mass, vehicle speed, vehicle inertia and aerodynamic friction.
Receiving as input the actual torque and, given the vehicle and the road resistance, it computes the vehicle translation balance. Then, it output the vehicle velocity which is used in the driver model as reference to determine the motor rotational velocity.

The driver model actuates the accelerator pedal position, in a scale of 0% to 100%, to follow the reference velocity of the selected driving cycle.
Then, the pedal position signal is converted in a torque request by a linear law to be sent by the CAN network to the EVCU.

The vehicle model, which is composed of the three sub-models just described, has been integrated in Veristand to work with the motor model described in the previous chapters.

In this case the desired torque comes from the simulated driver and goes to the EVCU through the CAN network; the generated torque is in input to the vehicle model; the other parameters have been set as constants.

5.3 Validation

Set all up, the experiment consists on passing the number id of the driving cycle to simulate to the vehicle model.
Thus, the simulated driver actuate the accelerator pedal which is converted to the desired torque as described before. The latter is sent to the EVCU through the CAN network, which actuates the IGBT switches necessary to the simulator to actuate a torque.

Note that all the Torque, Id and Iq plots have been normalized for confidentiality reason.

NEDC Driving Cycle

As first case, the standard NEDC driving cycle in figure 47 has been chosen, selecting the corresponding number in the cycle request in the simulator user interface.

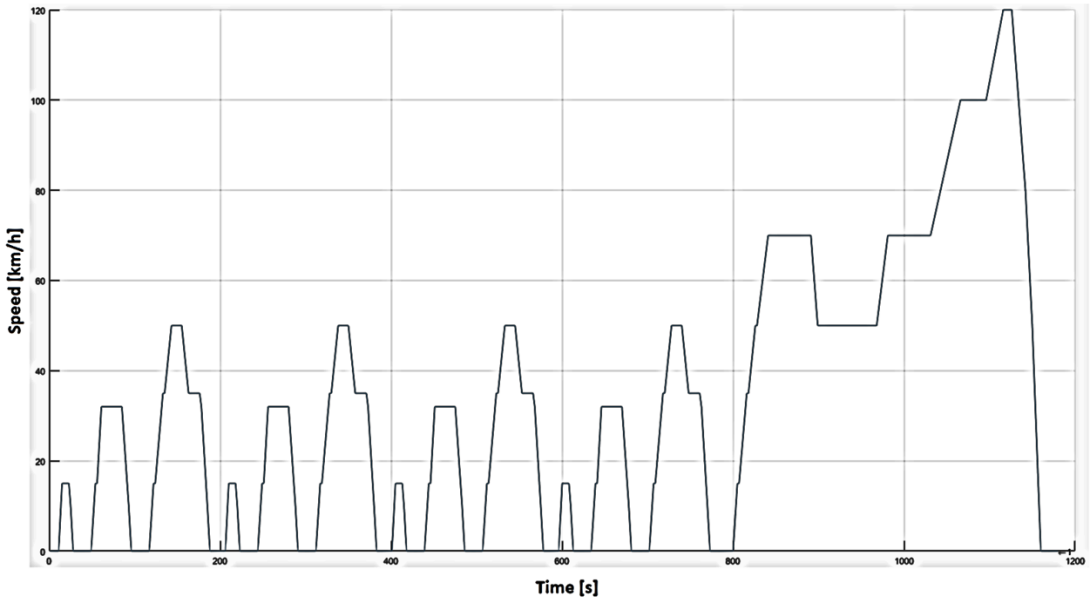


Figure 47: NEDC Driving Cycle

In the figure is represented on the **X** axes the driving cycle time and on the **Y** axes the desired vehicle velocity.

The simulator has followed the cycle, as shown in figure 48, and the torque has been correctly actuated, as shown in figure 49.

An additional consideration could be done for the Direct and Quadrature Currents generated by the HIL simulator and estimated by the EVCU. As shown in the figures 55 and 56, the are very similar to each other, confirming that the system is working correctly.

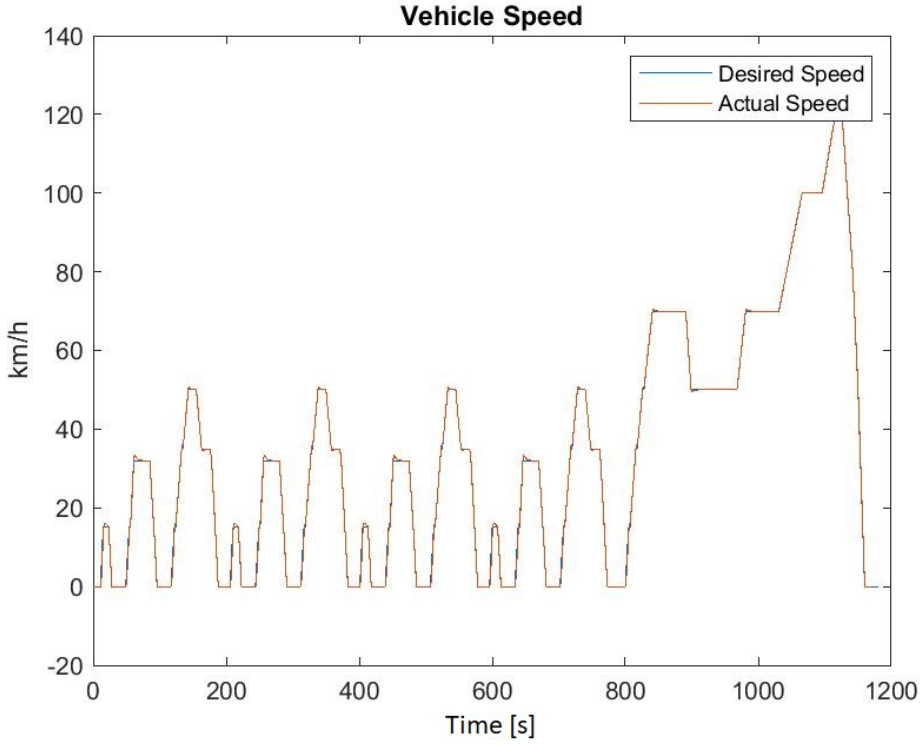


Figure 48: Vehicle Speed

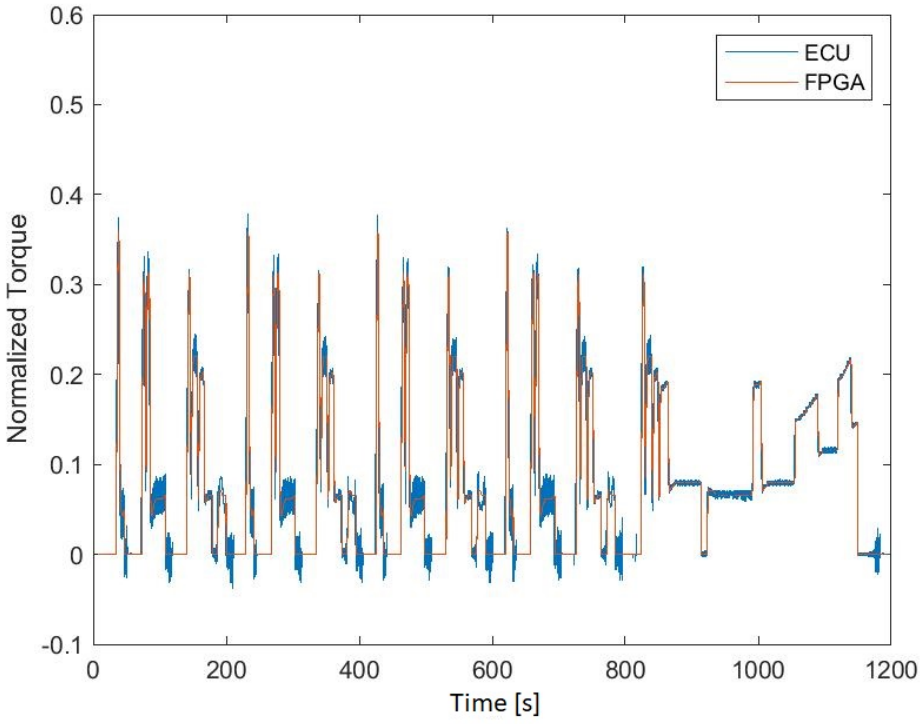


Figure 49: Torque

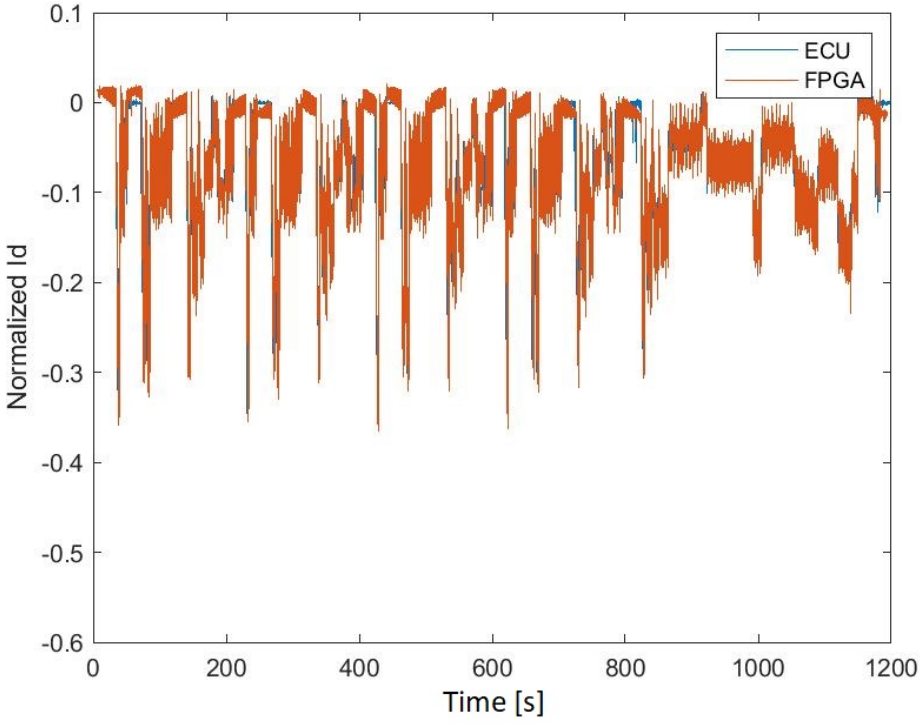


Figure 50: Direct Current

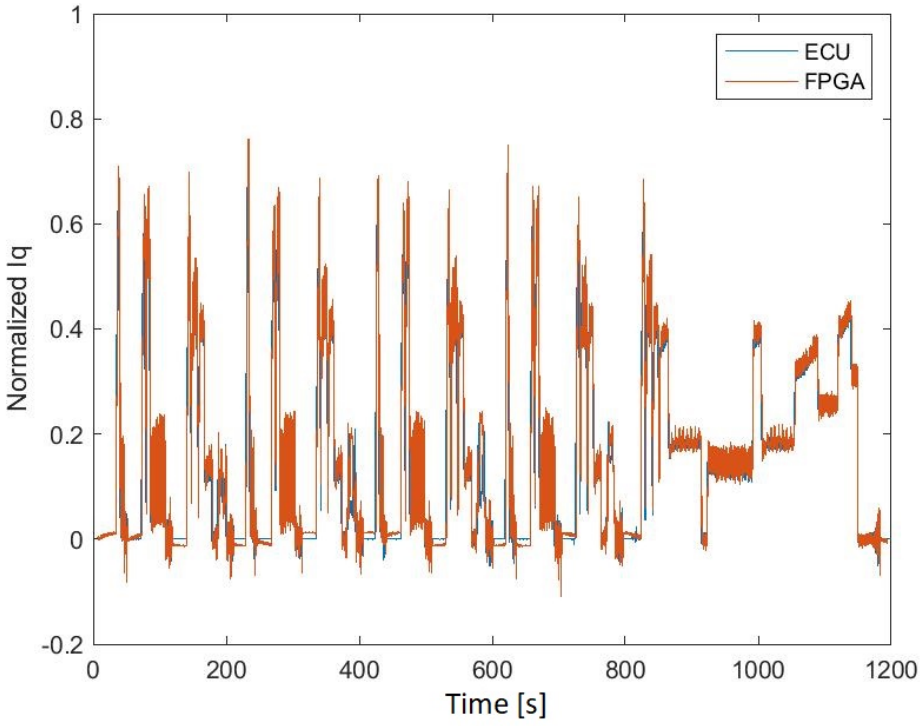


Figure 51: Quadrature Current

WLTC Driving Cycle

As second case, the standard WLTC driving cycle in figure 52 has been chosen, selecting the corresponding number in the cycle request in the simulator user interface.

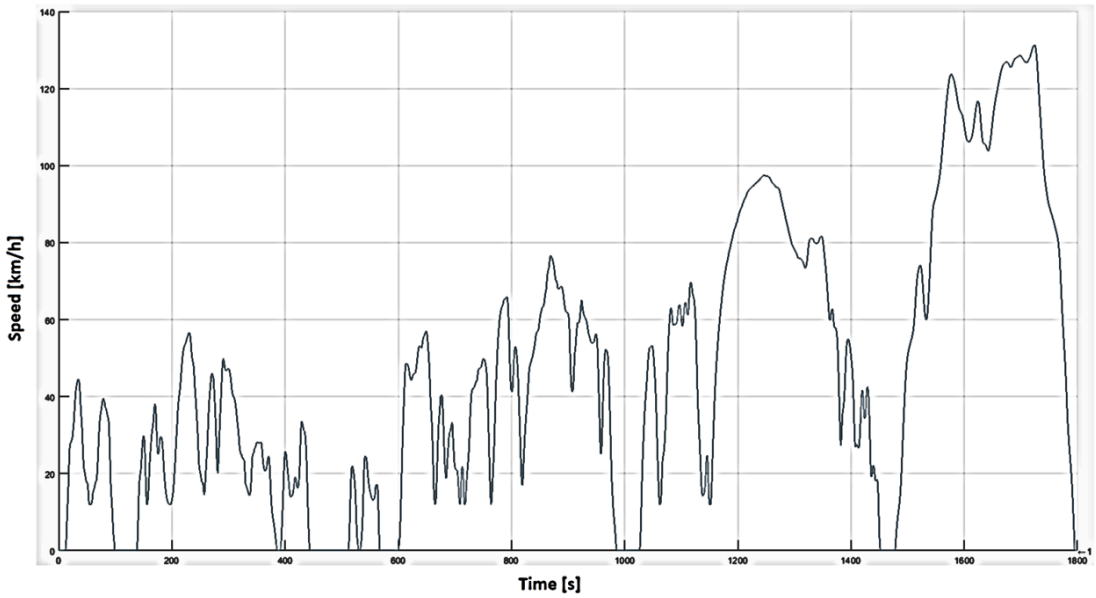


Figure 52: WLTC Driving Cycle

As the previous example, in the figure is represented on the **X** axes the driving cycle time and on the **Y** axes the desired vehicle velocity.

The simulator has followed the cycle, even in this case, as shown in figure 53, and the torque has been correctly actuated, as shown in figure 54.

Again, an additional consideration could be done for the Direct and Quadrature Currents generated by the HIL simulator and estimated by the EVCU. As shown in the figures 55 and 56, the are very similar to each other, confirming, again, that the system is properly working.

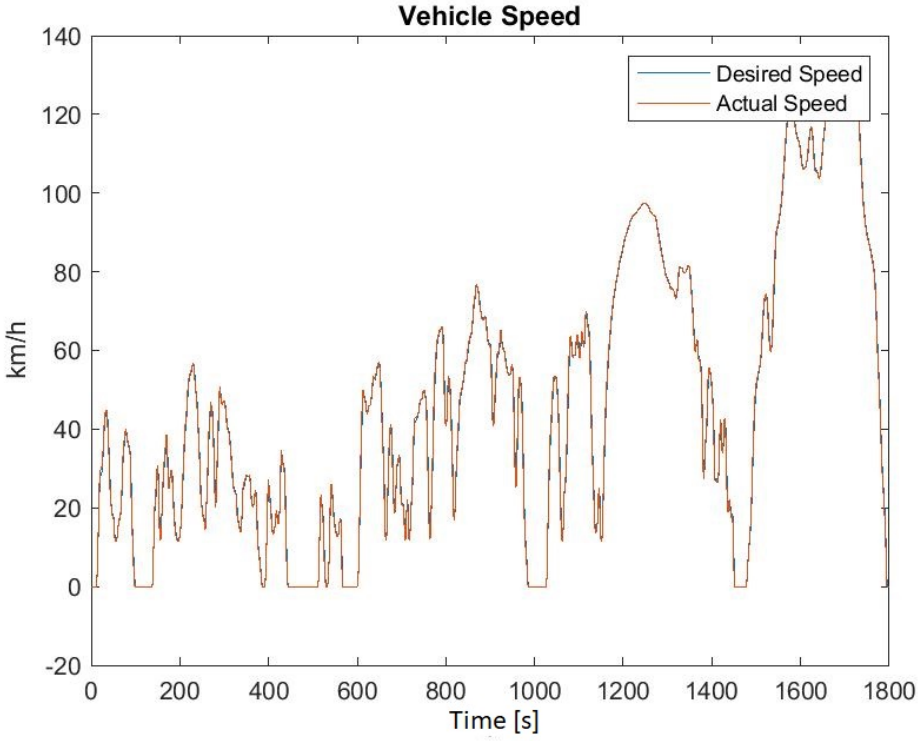


Figure 53: Vehicle Speed

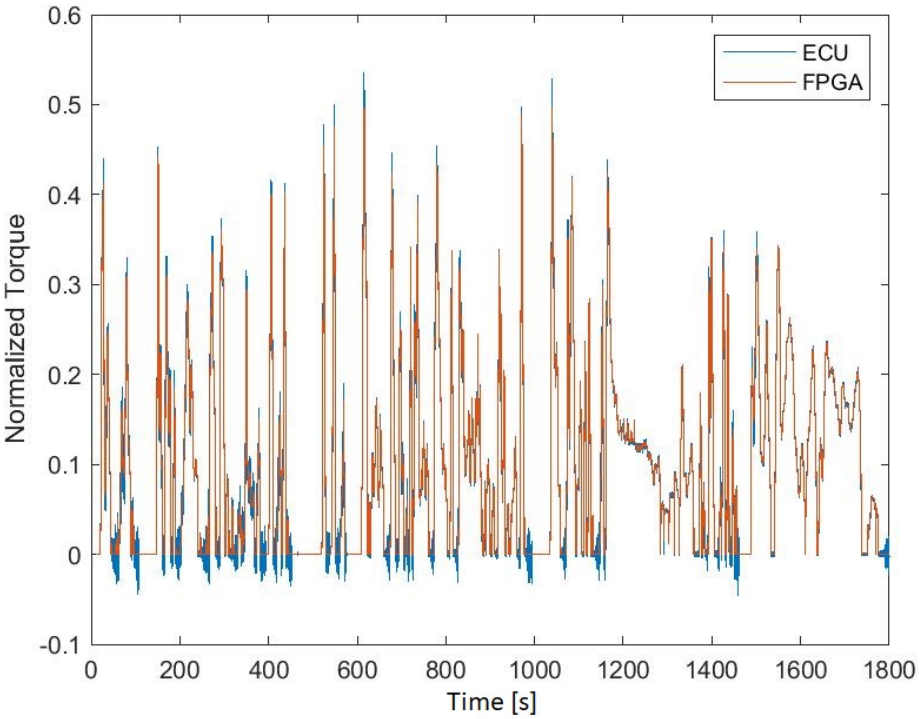


Figure 54: Torque

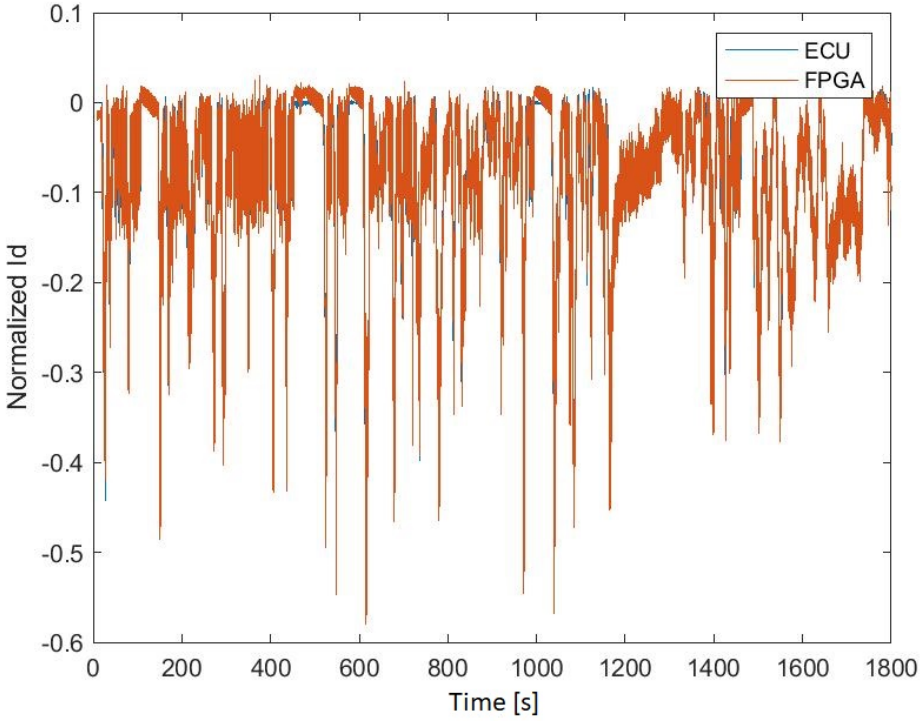


Figure 55: Direct Current

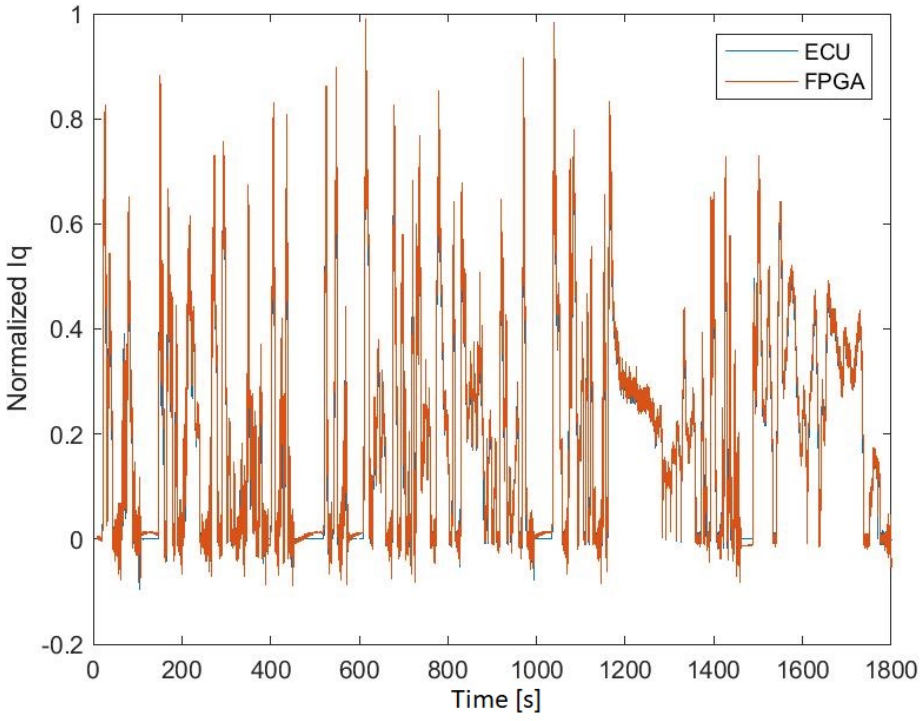


Figure 56: Quadrature Current

6 Conclusions and future application

The HIL application has been configured and it is operational in Dyno mode, even though some faults are still present on the EVCU. The root causes of those faults are under investigation, but they don't compromise the project operation.

As it has been shown, the simulator has been configured firstly to simulate a PMSM motor with constant parameters, using the OPAL-RT add-on for Veristand. The HIL is connected to the EVCU by some I/O intefaces, such as digital and analog I/O, which is, on its turn, connected to the Gate Driver Board, that is necessary to generate the duty cycles of the IGBTs.

As it has been illustrated, the agreement between the EVCU estimated operating parameters and the model results is satisfactory, even though some deviations are observed mainly due to a quite simple electric motor modelling approach.

For this reason, an important future approach could be the substitution of the constant parameters model with the introduction of a table-based motor model, which mapped parameters are spaced in the motor working points.

In this way, it is supposed to improve the performances of the simulated model being as close as possible to the real motor, which is the main goal of the entire project.

The second approach shown is the introduction of a part of the vehicle model. It consists in three sub-models regarding the vehicle dynamics, the driveline and the virtual driver.

It has been introduces in Veristand to work in parallel with the motor model, with the objective to simulate a driving cycles selected by the user.

A virtual driver controller actuates the pedals to follow the pre-defined speed profile.

The pedals actuation is converted, as described in the previous chapter, to the desired torque, which is sent by the CAN network to the EVCU, which, on its turn, actuates the torque properly changing the IGBTs duty cycles.

Then, the actual torque read from the simulator is fed-back to the vehicle model, which, after some computation, closes the loop.

In this context, as a future work, the HIL application could be expanded up to the full vehicle domain and the inclusion of additional control nodes (real or emulated), such as the battery model including, for example, a recharge sub-model, and so on.

The main goal of all these future applications is to reproduce as close as possible the controller operating condition as in the real vehicle, to finally substitute the simulated parts with the real components in a real vehicle to perform the final test.

7 References

1. Professor Massimo Violante, lecture notes of "Networking technologies for Connected Vehicles"
2. https://en.wikipedia.org/wiki/Hardware-in-the-loop_simulation
3. https://it.mathworks.com/discovery/hardware-in-the-loop-hil.html
4. https://www.ni.com/it-it.html
5. https://wiki.opal-rt.com/display/DOCPEVS/Getting+Started
6. " Back-to-Reality: Crossing the Reality Gap in Evolutionary Robotics." by Zagal, J.C., Ruiz-del-Solar, J., Vallejos, P.
7. “Once More Unto the Breach: Automated Tuning of Robot Simulation using an Inverse Evolutionary Algorithm” by Bongard, J.C., Lipson, H.
8. "ePHASORsim Real-Time Transient Stability Simulator" by OPAL-RT Technologies.
9. "Virtualization of synchronized phasor measurement units within real-time simulators for smart grid applications" by Al-Hammouri, A.T; Nordstrom, L.; Chenine, M.; Vanfretti, L.
10. "Hardware-in-the-loop testing of DP systems" by Johansen, T. A.; Fossen, T. I.; Vik, B.
11. https://www.etas.com/en/