Framework for Real-Time Simulation of Hardware in the Loop Applied to Primary Frequency Control

[1]Barrios-Gomez, J.A.; [1]Sanchez, Francisco; [1]Gianfranco, Claudio; [2]Gonzalez-Longatt, Francisco; [3]Acosta Montalvo, Martha Nohemi; [4]Denysiuk, Serhii; [4]Strelkova, Halyna

[1]Centre for Renewable Energy System Technology - CREST Loughborough University

[2]Department of Electrical Engineering, Information Technology and Cybernetics - University of South-Eastern Norway

[3]School of Mechanical and Engineering - Universidad Autónoma de Nuevo León

[4]Power Supply Department - IEE Igor Sikorsky Kyiv Polytechnic Institute

Publisher's version: DOI: 10.1109/ESS50319.2020.9160023

Framework for Real-Time Simulation of Hardware in the Loop Applied to Primary Frequency Control

J.A. Barrios-Gomez, F. Sanchez and Gianfranco Claudio
Centre for Renewable Energy System Technology – CREST Loughborough University
Loughborough, United Kingdom
J.A.Barrios-Gomez@lboro.ac.uk

F. Gonzalez-Longatt
Department of Electrical Engineering, Information Technology and Cybernetics University of South-Eastern Norway
Porsgrunn, Norway
fglongatt@fglongatt.org

M. N. Acosta
School of Mechanical and Engineering Universidad Autónoma de Nuevo León
Nuevo León, México
martha.acostamnt@uanl.edu.mx

S. Denysiuk and H. Strelkova
Power Supply Department IEE Igor Sikorsky Kyiv Polytechnic Institute
Kiev, Ukraine
spdens@ukr.net

***Abstract*— This research paper proposes a framework to perform (hardware in the loop) HIL simulation in real-time (RT), the framework includes steps for the formulation of the Target RT Computer and Open Source System, which are important elements in RT modelling. In order to carry out tests, the implementation of the frequency response of a single area electric power system, including primary frequency control, is proposed. The electric power system was prepared to be implemented in the RT-Lab software that is compatible with MATLAB Simulink, in order to simulate in real-time with the Opal OP4510 device. Besides, the controller was designed in Simulink to be able to implement through the Arduino programming platform; they were also connected to work in HIL in RT. The results of the simulation and performance analysis show that for the case proposed with HIL, the primary control happens within a few seconds of the frequency variation. The control helps to re-establish the balance between the generated power and power demanded, thus stabilizing the system frequency. Elements formulated using this framework were successfully used in the primary frequency control.**

Keywords*— Hardware in the loop, Arduino DUE, Opal RT-Lab, real-time simulation, frequency response

I. Introduction

Systems theory is an interdisciplinary theory dedicated to the study of systems; where a system is defined as a group of interacting or interrelated entities that form a unified whole. A system could be something physical as a single organism, any organization or society, or any electro-mechanical or informational artefact; but also, more abstract as the financial system. *System of systems* (SoS) is not a new approach; however, this is an opportunity for the systems engineering community to define the complex systems of the twenty-first century. An SoS can be seen as the viewing of multiple, dispersed, independent systems in context as part of a more extensive, more complex system, as shown in Figure 1. There are several definitions of SoS, some of which are dependent on the particularity of an application area. In this paper, the SoS, also known as "*super system*" is defined as an integration of complex systems or dedicated systems coordinated together, that pool their resources and capabilities together to create a new system which offers more functionality and performance, to achieve a broader objective with even greater importance. The methodology for defining, abstracting, modelling, and analyzing systems of systems is referred to as systems of systems engineering [1], [2], [3].

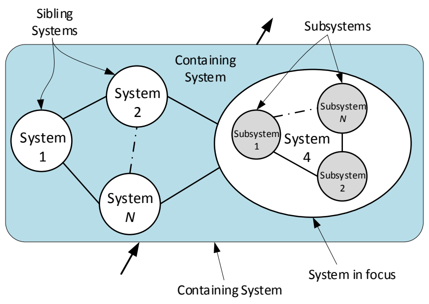

Figure 1. Schematic representation of an SoS inside a containing system.

A critical aspect of the SoS is that it brings together a set of systems for a task that none of the systems can accomplish on its own. The energy system is an excellent example of an SoS: smart grid, smart houses, and integrated production/consumption, etc. When a controller is designed, the option to ensure that this control is really doing what is expected is to perform an offline simulation. Ideally, what can be expected is to perform the actual implementation of the control, something that is, in most cases, complicated. Currently, with the significant development of the technology in the area of simulation of systems, it is possible to perform simulation in RT to be able to make tests of implementation of controllers using sophisticated equipment, thus achieving software in the loop or Hardware in the Loop, the latter is what that this research paper focuses. The main objective of this paper is establishing the elements to create a structure that facilitates the implementation of a hardware controller in an RT environment. This can be done by modelling and simulating systems in RT, where software and hardware parts are included, also called co-simulation [4]. Two different structures are implemented, proposing specific steps for the design of a controller that will be programmed in open-source hardware using analog signals and the rest of the system that is the model of the plant will be placed in a simulator in RT HIL. This will result in full implementation, thanks to the asynchronous communication between the model of the plant in the HIL device and the controller in the open-source hardware. This is a case of application of SoSE. According to the configuration and application, RT simulation can be classified into three main categories: a) fully digital simulation; b) controller HIL (CHIL) simulation; and c) power HIL (PHIL) simulation. The application of DRTS (digital simulations in RT), is involved in several stages of product development in the industry, these stages are usually: Specify Feasibility, Design RT, Prototyping RT, implement RT, test RT [5].

The co-simulation is a platform that allows the analysis of monitoring and control system applications for MV/LV electrical grids [6]. These tests on real systems can be costly and time-consuming. Besides, it is necessary to build prototypes to perform the tests; these tests in real conditions can be done using offline simulations. Methods for RT simulation (RTS) include the use of hardware: digital signal processors, general-purpose processors, as well as reconfigurable computational solutions that employ Field-Programmable Gate Arrays (FPGA) [7].

II. System in the Loop (SS-I-L)

A. *Logic Layout*

In this section, a logical Layout of the SS-I-L architecture is described. In Figure 2, it can be seen, the *Target RT Computer* (TARGET-RTC) is connected to the Interacting *Open Source System* (OS-SYS) and coupled to a *Host Computer* (HOST-PC). TARGET-RTC is responsible for preparing, controlling and configuring the simulation platform in RT, and is also responsible for the entry and registration of data by the user to do this uses a high-performance processor.

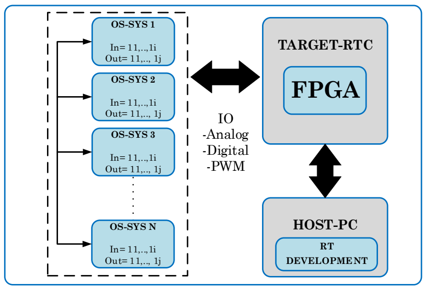

Figure 2. Logical Layout for SS-I-L

B. *Description of Main Blocks*

(i) TARGET-RTC: Is the system of interest is coupled to a set of N Open Source System, the function of the TARGET-RTC is to administer, control and configure the simulation platform in RT. (ii) OS-SYS: The OS-SYS is the interacting system, are auxiliary hardware whose function is to receive and generate signals related to the dynamic system that the TARGET-RTC will simulate. OS-SYS can be implemented by various hardware platforms, such as microcontrollers, digital signal processors, and other devices that have digital and/or analog inputs and outputs. Each OS-SYS can include i digital and/or analog inputs and outputs. Thus, each SS-I-L must have at least one OS-SYS module. (iii) HOST-PC: Is the development computer where RT models are designed, using specialized software for model and system design, and specialized software for RT model implementation.

III. Framework for TARGET-RTC and OS-SYS

Transferring a simulation model from PC to simulation in RT is a process that may require a significant reconfiguration of the original model. The main objective of model preparation is to ensure that the model is capable of operating in RT. The model is possible in RT if it is sufficiently precise to generate simulation results that coincide with what is expected, based on theoretical models and empirical data. Also, fast enough to run on the destination machine in RT without overshoot. In the preparation of the model, the reference results are obtained, and the step size is determined to evaluate if it is probable that the model is possible in RT. In order to build an RT-HIL model and perform tests, the following necessary steps are proposed in the next subsections.

A. *Model design for RT*

A model should be created in a high-level programming environment to simulate the behavior of dynamic systems and configure their parameters to use a fixed-step solver that specifies a sample time compatible with the RT requirements of the model.

1) Offline model simulation. This simulation is carried out to guarantee that the model works correctly. The outputs can be calculated for given time values; the calculated time vector is not connected to a physical clock, so the outputs are calculated as fast as H-COMP can achieve. The elapsed time of the simulation may differ from the elapsed time of the real system.

B. *Parameter configuration: development and target equipment*

The configuration of the communication method between H-COMP and M-TARGET involves the construction of the model to implement in RT, the configuration of sampling time and environment of construction and IO configuration and connection.

1) Model construction for RT implementation. Set the configuration parameters of the model in values compatible with real-time execution.

2) Setting the built environment. The compilation environment must be configured. The environment includes the coder options, the RT compilation options, and the C compiler options.

3) IO Configure-connections to the physical hardware. Configuration the IO modules on the target computer and connect the modules to the physical hardware.

C. *Build and download the RT model*

The RT real an M-TARGET bust be created and downloaded.

1) Run the RT model. The RT software uses real-time resources on the target computer. Based on your sample rate, the RT software uses interrupts to step the model at the sample rate. With each new interrupt, the real-time application computes the block outputs from your model.

2) *Signal analysis.* Scopes created by RT scope blocks acquire data according to Simulink sample time rules. Scopes can gather data at the top level or in an enabled or triggered subsystem.

3) *Setting Parameters.* Parameters tuning such as time delays, input and output amplitudes, and input and output frequencies.

D. Model tests

It is necessary to make a test model, to verify that the model is running correctly.

IV. OPAL MODEL OP4510 RT SIMULATOR

The OP4510 simulator is a complete system that works with a Kintex7 FPGA programmable device; this simulator is designed to be used as a desktop computer, it contains a powerful and flexible computer, a powerful high-speed processor and a signal conditioning stage. Standard connectors (DB37, DB9) are used in the IO. The location of IO is identified by a slot or group number, a module number and a subsection number. Table I shows the details of the signal conditioning modules provided in the system.

TABLE I. LOCATION OF SIGNAL CONDITIONING MODULES

I/O group	**Subsection A**	**Subsection B**
1	Digital Input OP5353	Digital Outputs OP5360-2
2	Analog input OP5340 (1 MSPS)	Analog Output OP5330
Optional expansion module 6TX/6RX		
TX/RX 0-5	OP5969-1 RS422 DIO (OP4510-1)	

A. Bitstream configuration files for IO:

To be able to use IO in the simulation of models in RT using OP4510 it is necessary to use two types of files: *.Bin* and *.Conf*, which have the purpose of configuring inputs and outputs of the simulator, these are provided by the manufacturer of OPAL-RT.

.Bin file: The .bin file is provided together with the system by OPAL-RT; this file is intended to answer all the simulation needs in RT.

.Conf file: The .conf files are the configuration files, the device has slots and sections, the current bitstream is made for a hardware configuration for example with an analog output card in group 1A, an analog input card in group 1B, a digital input card in group 2A and a digital output card in group 2B.

These bitstream files are VHDL encoded files and need to be loaded on the FPGA card of the target computer so that they can be configured before real-time simulation execution. The Simulink RT-LAB blocks used to access the IOs in the OP4510 simulator use the following identification nomenclature: *Slot X Module Y Subsection Z.*

Assignment of the Simulink ID to the actual physical IOs in the OP4510, Table II is used as a reference. For example, Slot 2. Module B. Subsection 1. The following Figure 3 shows the arrangement of the signal subsections in the DB37 and DB9 connectors. For more details on the type of cards, reader can consult the reference sheets of OP4510 in [9], [10], [11] and [12].

TABLE II. LABELS SIMULINK IO BLOCK AND OP4510 IOS

IO nomenclature of the Simulink block	**Identification IO (ID) nomenclature in OP4510**
Slot	Group (Subsection)
Module A	1
Module B	1
Module A	2
Module B	2

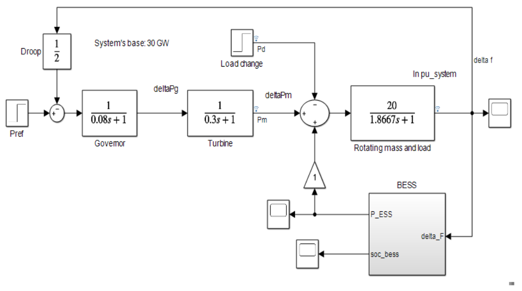

Figure 3. Singular area electrical system with an energy storage device.

To work with this model in RT and perform HIL the model of Figure 3 has to be rearranged, the electric model will be charged in OPAL-RT, and the storage model BEES will be loaded in an Arduino DUE model Figure 4.

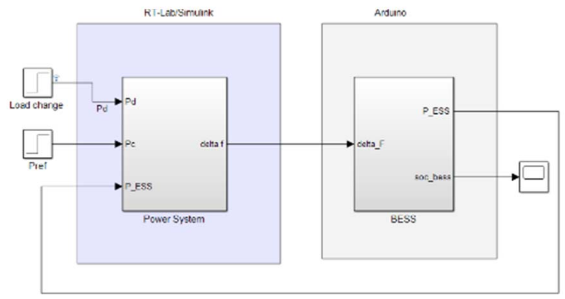

Figure 4. Connection diagram between Opal and Arduino.

From the system of Figure 4, it is necessary to separate it into two subsystems to work in OPAL-RT, and it must be taken into account that some components of the model will be in the subsystem SM (computing) and SC (interface) the computing components only they are in the SM subsystem, Figure 5.

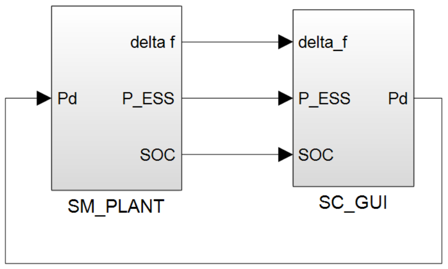

Figure 5. Master and console subsystems for use in RT-LAB

B. Master subsystem

In this SM master subsystem, all the computational elements of the mathematical operations model, IO blocks, etc. are placed. So here the power system will be placed. OPAL-RT has designed the OpComm block to allow communication between

Command Station, which is the computer where Opal-RT is installed and Target which is the OP4510 real-time simulation equipment in a distributed simulation, this can be seen in Figure 6. The blocks to configure the Analog IO for this model are described below:

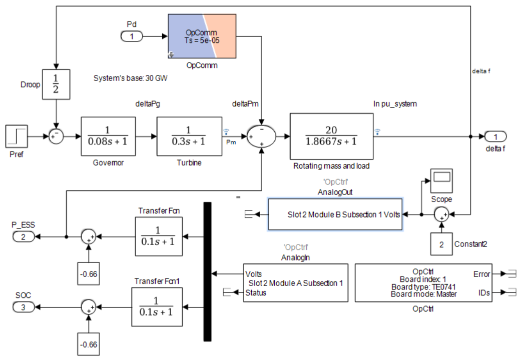

Figure 6. SM master subsystem.

OpCtrl Block: The OpCtrl block is used to select the FPGA by the board index; it also serves to program the FPGA with the specified bitstream file and controls the hardware synchronization of the model. It is necessary to use the Bitstream files .bin and .conf, to configure the FPGA card with this it will be known in which pins of the DB37 connector they have to send analog signals and where they will be received. These files must be in the same folder where the Simulink file that is used in RT-LAB is located.

1) *AnalogOut block:* The AnalogOut analog output blocks apply their input values to the channels of the analog output module installed in OP4510 device connected to the TE0741 card. Each block controls 8 channels of the analog output module. In the section "Slot infos" it shows the identification data of this block "Slot 2 Module B Subsection 1", this means that the signal will come out of a connector that will be in position 2B (Slot 2 Module B), and can contain the first 8 channels (subsection 1) and also indicate how many outputs (channels) will be used with "Number of Aout Channels" which for this case is 1. The voltage range is set to 5 V to be compatible with Arduino.

2) *AnalogIn Block:* The AnalogIn analog input blocks receive the acquired data from the analog input module installed in the OP4510 carrier connected to the ML605 card. Each block receives the voltage values of 8 channels of the Analog In module. Here "Slot infos: Slot 2 Module B Subsection 1", indicates that the signal is expected to enter through the connector that will be in position 2A (Slot 2 Module A), within the first 8 channels (subsection 1) and also indicates how many inputs (channels) will be used with "Number of Ain channels" which for this case are 2.

3) *Operation of Master subsystem:* The Master subsystem contains the complete model to simulate in RT, as well as the IO configuration blocks, in addition to the analog input blocks and output analogs. In Figure 6, the BESS storage subsystem is removed and later integrated into an Arduino DUE device. Due to this we have an analog output block which will send the signal "delta f" to an analog input of the Arduino, before this block we have an operation where a constant is added, this point is very important since Arduino only it reads positive values, so the whole signal that is sent has to be on the positive side, the size of the constant that in this case is two will depend on the size of each system, this can be seen in Figure 7 (b).

In the AnalogOut block, there are two inputs that come from the Arduino which have to be separated with a "demux" block, to these signals they have added a first-order filter as used in [13], to eliminate electronic noise. Arduino by default has a fixed value of Offset Voltage of 0.56 and as mentioned above the values that enter or leave the analogue signals of Arduino must be positive, so by the size of this signal in the outputs of the Arduino were added a constant of 0.1. In Figure 7 (a), after the filter, a rest of 0.66 is left, which is the sum of these two values.

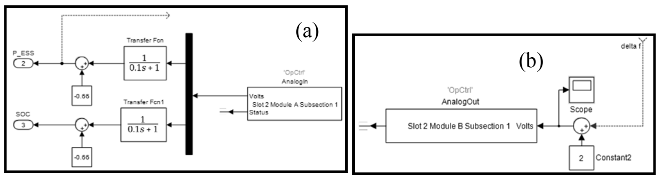

Figure 7. (a)Analog input and (b) Analogue output

C. *Console subsystem.*

The SC console subsystem contains user interface blocks (scopes, displays, manual switches, constants), runs on the host PC asynchronously from the compute subsystems. There is no generation of signals, mathematical operations or physical model [14]. It is necessary to have another OpComm block which will receive the signals coming from the Master block; there will be three signals, as seen in Figure 8.

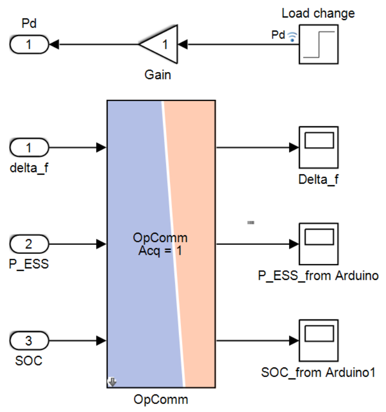

Figure 8. SC console subsystem

In the console subsystem shown in Figure 6, three signals are received that enter the OpComm block and then visualized in the Scopes when the model is running in RT, the delta signal f is received from the electric power model, P_ESS and SOC are signals that have been generated from the Arduino and received by OP4510, and that can now be seen in Simulink's real-time model.

D. *Battery storage model Arduino*

Next, is presented the storage model with a battery that functions as control when changes are made in the frequency, which will be loaded in Arduino, to demonstrate its operation. One of the main points is to be able to include and synchronize in the HIL loop in conjunction with OPAL-RT and finally be able to simulate the complete system in RT.

1) Features Arduino DUE: Arduino Due is a device with a microcontroller based on the Atmel SAM3X8E ARM Cortex-M3 CPU and is the first Arduino board based on a 32-bit ARM core microcontroller. Arduino Due operates at 3.3 V. Therefore, the maximum voltage that the IO pins can tolerate is 3.3V, voltages higher than 3.3 V to any IO pin could damage it [15]. Since the model of Simulink Figure 6, developed for RT-Lab will generate in the IO analog signals with OP4510, it was necessary to choose Arduino DUE since it also has analog IO signals. Next, the Simulink model implemented in Arduino DUE is presented.

2) Implementation model storage battery Arduino with Simulink: Simulink has a package that supports Arduino hardware; this can be used to develop and simulate algorithms that run independently in Arduino. Is possible to detect Arduino hardware complements in MATLAB, as well as other communication and configuration features. It also has blocks of a simultaneous link to configure and access the inputs and outputs of Arduino and external mode for the adjustment of interactive parameters and monitoring of the signal to while the algorithm is executed on the device [16]. Arduino Due was chosen because it has analog IOs, the processing and storage capacity is more significant with respect to other Arduinos; these characteristics are essential for HIL interaction with OP4510. The operation of Arduino Due with OP4510 is not direct, so it is necessary to add some operations so that analog signals entering and leaving Arduino are compatible with OP4510, this is shown in Figure 9.

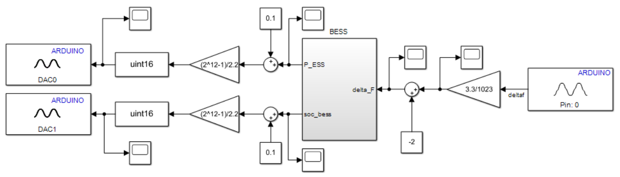

Figure 9. Battery storage model.

The signal received by Arduino Due in the Analog Input block (pin 0) defined as deltaf comes from the power system model. If the measured voltage is equal to the analog reference voltage, the output of the block outputs 1023. To perform the coupling of the analog input signal with the BESS subsystem, it is necessary to add a 3.3 / 1023 gain that modifies this signal to the required ranges. It is necessary to perform a correction of the offset that is added with a constant of 2, as it is seen in Figure 10.

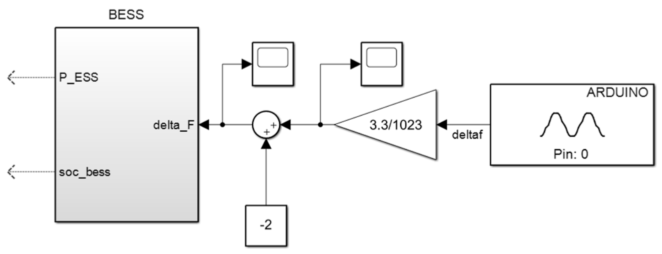

Figure 10. Analog input in Arduino Due.

From the BESS subsystem, the power signals that return to the electrical system are obtained: P_ESS and the battery discharge signal and need to be sent back to the OP4510. It is necessary to add an offset to pass the signal to the positive side; this will depend on the size of this, 0.1 is used for this model. The analog output block generates a voltage on the specified DAC pin of the Arduino Due that has two 12-bit DAC pins, DAC0 and DAC1. When the block is used, it accepts a value uint16 but considers only the 12 most significant bits for the conversion. This block emits voltage (V_{out}) on the DAC pin, such that:

$$V_{out} = V_{ref}\left(\frac{input}{2^N} - 1\right) + V_{offset} \tag{1}$$

where $V_{ref} = 2.2$, $input = 0 < input < 4095$, $N = 12$, $V_{offset} = 0.56$. Therefore, in order to obtain the output signal in the correct ranges, it is necessary to obtain input from equation (1), as shown in equation (2).

$$input = V_{out}\left(\frac{2^N - 1}{V_{ref}}\right) - V_{offset} \tag{2}$$

Substituting the known values have:

$$input = V_{out}\left(\frac{2^{12} - 1}{2.2}\right) - 0.56 \tag{3}$$

Before the Arduino sends the signal to the outside, an offset block of 0.1 must be added to have an entirely positive signal, a gain of (2^12-1)/2.2; also a block of data type conversion uint16, as can be seen in Figure 11.

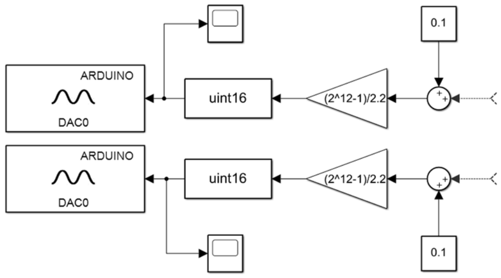

Figure 11. Analog output in Arduino Due.

V. Simulations and Results

The Simulations of Hardware in the Loop was experimentally validated by implementing the frequency response of a single area electric power system with primary frequency control. The electric power system was prepared to be implemented in RT-Lab software that is compatible with Matlab Simulink. The results of these tests were compared with those obtained with the MATLAB/Simulink platform in simulations that were not performed in RT. Figure 12 shows the equipment used to carry out the experimental tests.

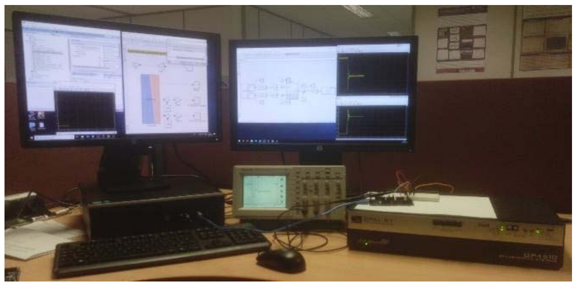
Figure 12. Equipment used to carry out the experimental tests.

A. Presentation of the result: Frequency Response

Results of the system are presented in Figure 13., it shows the results with the non-RT model obtained using MATLAB / Simulink, and in Figure 14 it shows the simulation obtained in HIL, and the image was captured from a Tektronix brand oscilloscope (TDS 210). The HIL simulation provided satisfactory results, with minimal errors in relation to the reference method. Basically, Figure 13 and Figure 14 show the results obtained for a frequency controller that uses a battery storage model integrated into an ATmega 2560 microcontroller external to the HIL platform. This experiment demonstrated the benefits of using HIL models in the development of integrated control algorithms, where the performance and feasibility of the new algorithms can be tested in RT, without the need to purchase expensive physical equipment.

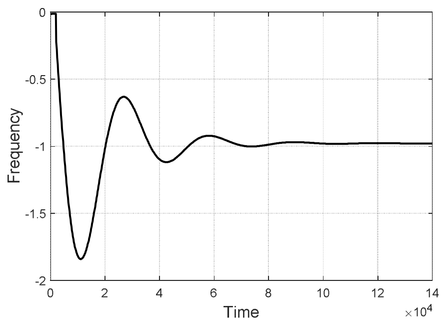

Figure 13. The non-RT model Frequency response obtained

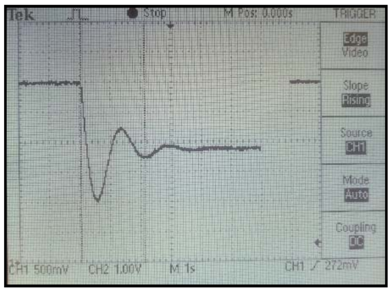

Figure 14. The RT model Frequency response obtained Tektronix scope.

VI. Conclusion

For the implementation was necessary to design a model for RT-Lab and configure the analog signals of outputs and inputs. It is necessary to design a model to be implemented in Arduino Due, since it has the possibility of accepting IO analog signals and also the high speed and information storage characteristics. The Arduino Due card has the capacity to communicate with OP4510, and for this, in the Simulink scheme it was necessary to carry out operations in order to have the IO signal in the necessary ranges, and therefore the BESS subsystem can work with the real values, after this to the output of the BESS subsystem it was necessary to add an offset, a gain to send voltage signals in the necessary ranges and a data converter block so that the output block can send the signal in the correct range. It was concluded that the controller implemented in Arduino, working in RT with the model of the electrical system implemented in RT-Lab, sent a control signal that benefits the recovery after the failure caused by an event. The framework was successfully evaluated by applying frequency response using a BEES storage model, implemented with the OPAL-RT and Arduino DUE devices.

Acknowledgment

J. Barrios-Gomez is grateful to CONACYT-Mexico for providing a Postdoctoral research fellowship.

References

[1] M. Jamshidi, "System of Systems - Innovations for 21st Century," in *2008 IEEE Region 10 and the Third international Conference on Industrial and Information Systems*, 2008, pp. 6–7.

[2] M. Jamshidi, "System of systems engineering - New challenges for the 21st century," *IEEE Aerosp. Electron. Syst. Mag.*, 2008.

[3] B. Tekinderdogan, "Engineering connected intelligence : a socio-technical perspective," 2016.

[4] R. Le Moigne, O. Pasquier, and J. P. Calvez, "A generic RTOS model for real-time systems simulation with systemC," in *Proceedings -Design, Automation and Test in Europe, DATE*, 2004.

[5] X. Guillaud *et al.*, "Applications of Real-Time Simulation Technologies in Power and Energy Systems," *IEEE Power Energy Technol. Syst. J.*, 2015.

[6] M. Armendariz, M. Chenine, L. Nordström, and A. Al-Hammouri, "A co-simulation platform for medium/low voltage monitoring and control applications," in *2014 IEEE PES Innovative Smart Grid Technologies Conference, ISGT 2014*, 2014.

[7] I. de Souza, S. Silva, R. Teles, and M. Fernandes, "Platform for Real-Time Simulation of Dynamic Systems and Hardware-in-the-Loop for Control Algorithms," *Sensors*, vol. 14, no. 10, pp. 19176–19199, Oct. 2014.

[8] B. S. Achary, S. Mishra, and A. Kumar, "Real time hardware in loop testing of single phase grid connected PV system," in *2014 18th National Power Systems Conference, NPSC 2014*, 2015.

[9] Opal-rt, "OP5353 16/32 Digital Inputs," in *User Guide*, .

[10] T. OPAL-RT, "Preparing simulink model real time execution." [Online]. Available: https://www.opal-rt.com/opal_tutorial/preparing-simulink-model-real-time-execution/.

[11] OPAL-RT(Technologies), "OP5360-2 USER MANUAL," *5 to 30V 32 Digital Output Module*. [Online]. Available: https://www.opal-rt.com/wp-content/uploads/2016/07/OP5360-2-User-Manual.pdf.

[12] OPAL-RT(Technologies), "OP5330 USER GUIDE," *Digital to Analog Converter Module*. [Online]. Available: https://www.opal-rt.com/wp-content/uploads/2016/07/OP5330_User_Manual.pdf.

[13] M. Gavran, M. Fruk, and G. Vujisić, "PI controller for DC motor speed realized with Arduino and Simulink," in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2017, pp. 1557–1561.

[14] OPAL-RT(Technologies), "Preparing simulink model real time execution." [Online]. Available: https://www.opal-rt.com/opal_tutorial/preparing-simulink-model-real-time-execution/.

[15] © 2019 Arduino, "Arduino Due." [Online]. Available: https://store.arduino.cc/due%0D.

[16] H. Support, "Arduino Support from Simulink." [Online]. Available: https://uk.mathworks.com/hardware-support/arduino-simulink.html.

[17] Matlab, "Analog Input Arduino," *Measure voltage of analog input pin*. [Online]. Available: https://uk.mathworks.com/help/supportpkg/arduino/ref/analoginput.html.