Mondragon Unibertsitatea

Faculty of Engineering

DC motor control validation with HIL platform

Master in Smart Energy Systems

Mondragon Unibertsitatea Goi Eskola Politeknikoa

CONTENTS

1 DOCUMENT PURPOSE

The objective of this document is to show the development and implementation of a HIL platform for an electric scooter.

Figure 1 shows a simplified block diagram of the scooter.

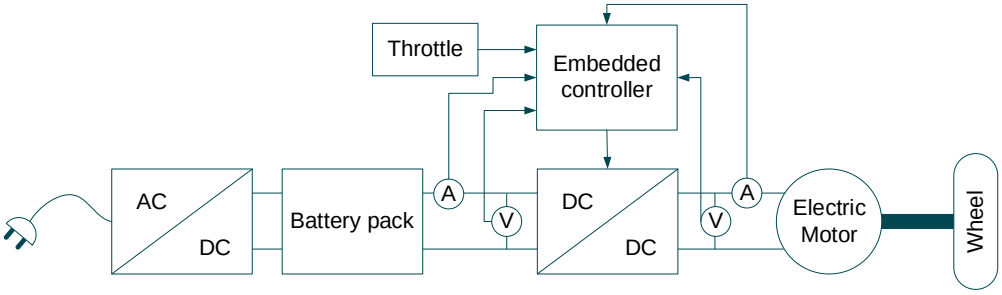

Figure 1 Electric scooter traction system

Both hardware and software development of the Hardware-in-the-Loop platform have been addressed.

As for the hardware, the HIL platform consists of two Texas F28379D boards. One acts as the controller and the other emulates the plant in real time (battery and AC/DC converter are not considered).

For the software development, a conventional MIL simulation has been used as a starting point, which has been transformed for real-time execution.

For the HIL results to be as close as possible to the MIL, different factors are considered that may influence the real-time simulation on the boards, such as: the offset of the analog inputs and outputs, ensuring the correct order of the calculations in the real-time simulation, the resetting of the integrators, or the execution time.

Once the HIL simulation has been adapted, an external simulation that monitors the data from the PC (HOST) has been used to collect the necessary data to compare the results with those of the MIL simulation.

2 SIMULATION DEVELOPMENT

2.1 General description

The development of the HIL platform has started from a MIL simulation of the control of a DC motor used for an electric scooter. Figure 2 shows the structure of the simulation.

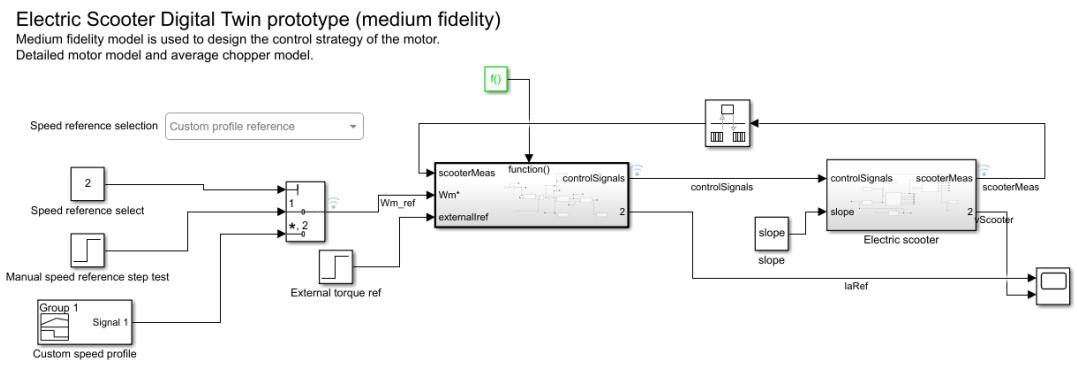

Figure 2 Simulation of electric scooter

The plant is composed of a subsystem that simulates the behavior of a DC motor, to then obtain the torque generated and calculate the speed that the electric scooter will achieve by simulating the mechanical model of the scooter.

In the case of the DC motor and the mechanical system, its behavior is described by the equations (1), (2), (3) y (4):

$$\frac{dI_a}{dt} = \frac{V_L}{L_a} = \frac{V_a - \omega \cdot K_e - R_a \cdot I_a}{L_a} \quad (1)$$

$$T_{em} = I_a \cdot K_t \quad (2)$$

$$T_m = T_{em} - \omega \cdot K_f - \omega \cdot D \quad (3)$$

$$\frac{dv_{scooter}}{dt} = \frac{2}{m_{total}} \cdot \left(T_m - \frac{n_{gear}}{eff_{gear} \cdot R_{wheel}} - F_{drag} - F_{hc} - F_{rolling} \right) \quad (4)$$

The control uses the current measured in the plant to compare it with the reference; and with the error achieved, make a control by means of a PI controller. To obtain the duty cycle (δ) of the chopper, an average model is used, which is represented by (5):

$$\delta = \frac{V_a}{V_{bat}} \quad (5)$$

2.2 Simulation results

To verify the performance of the MIL, several simulations have been carried out. The following figures show the current control and speed evolution of the scooter. In this case the current control has been simulated without any speed control.

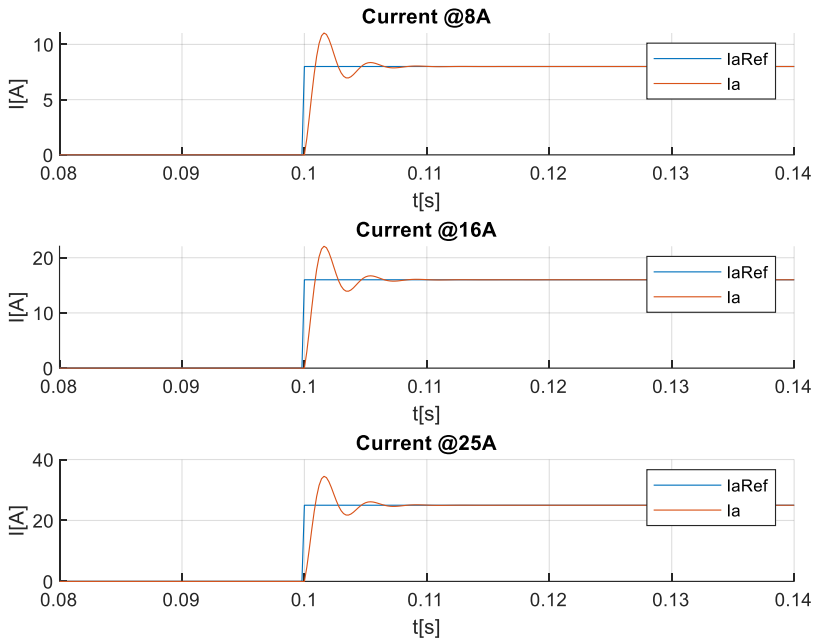

Figure 3 Current control for different references

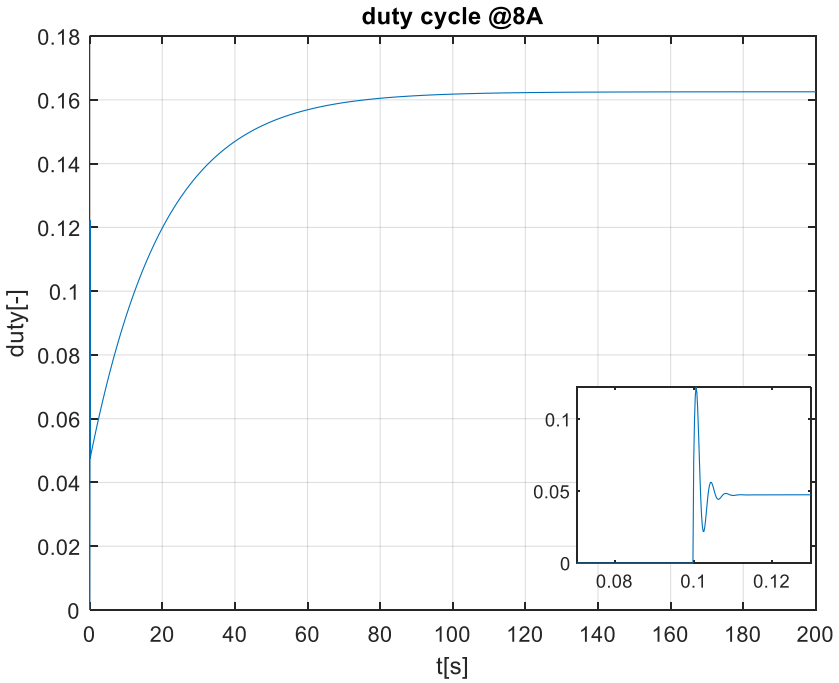

Figure 4 Chopper duty cycle for 8 A

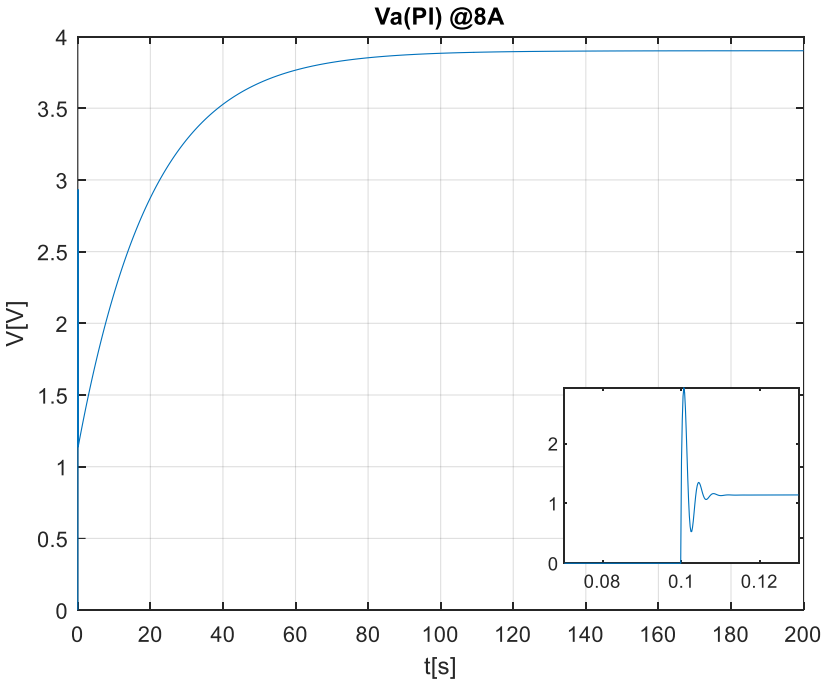

Figure 5 Motor voltage with 8 A

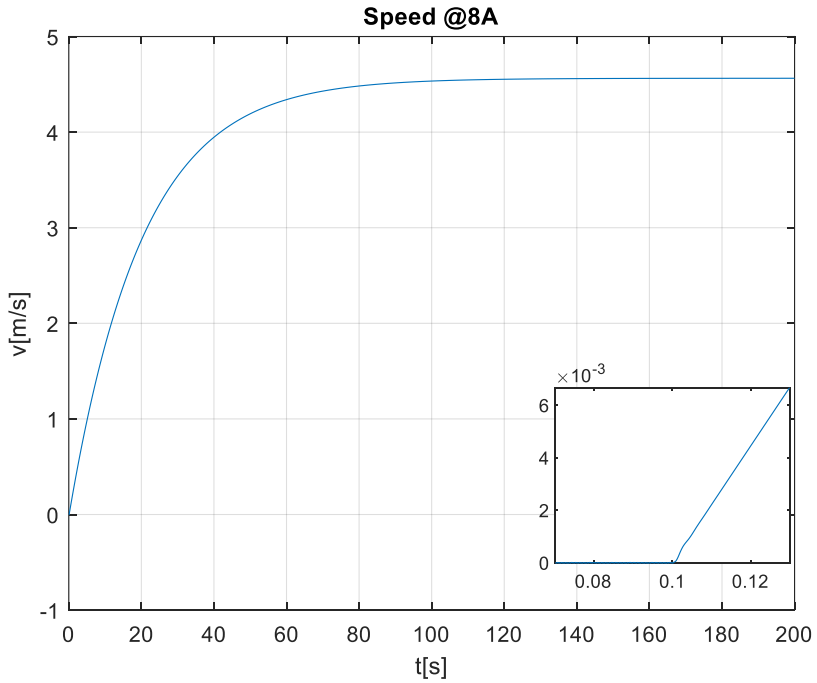

Figure 6 Speed with 8 A

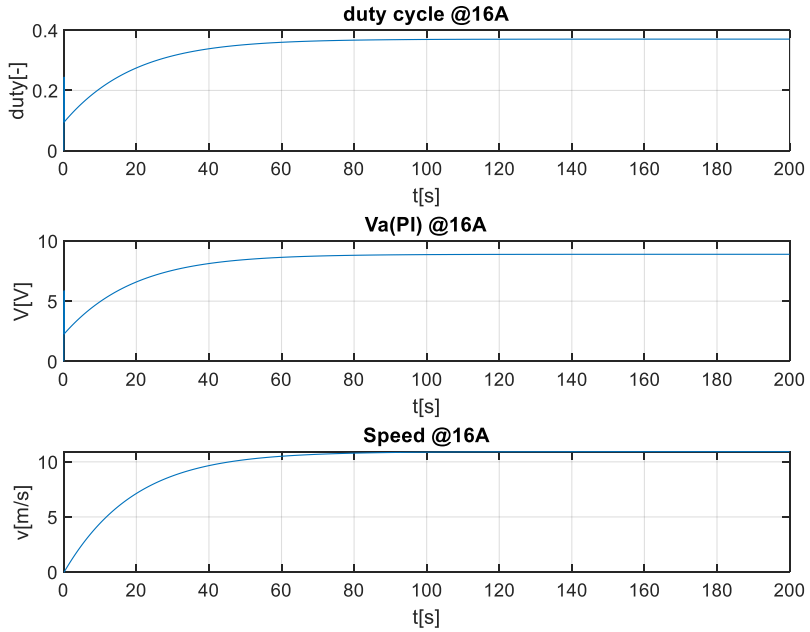

Figure 7 Duty cycle, voltage and speed of the motor at 16 A

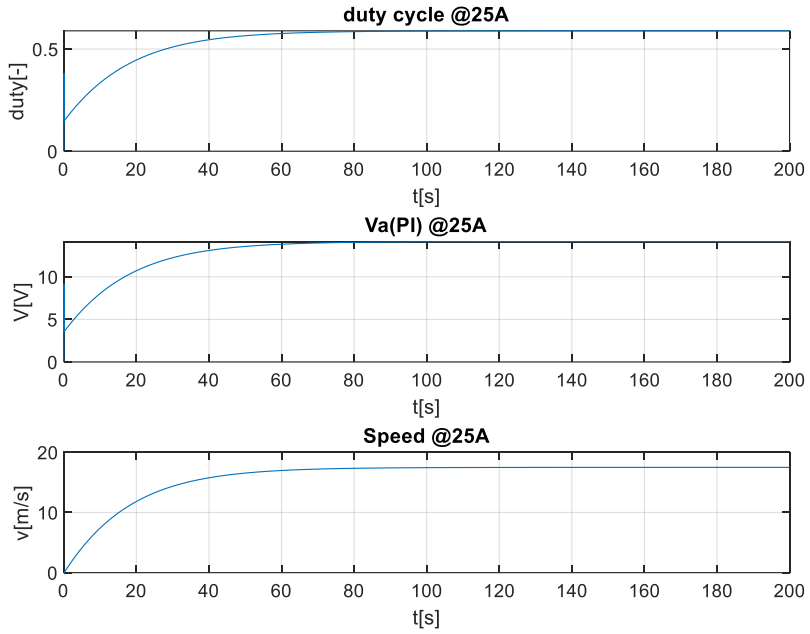

Figure 8 Duty cycle, voltage and speed for 25 A

As can be seen, each current step causes an exponential increase in speed, until the permanent regime is reached, which is the result of the above formulas.

3 HARDWARE IN THE LOOP PLATFORM

3.1 Hardware design

The HIL platform is composed of two TI Delfino F28379D LaunchPad boards, one simulating the control and the other the plant, connected by their ADCs and DACs. Figure 9 shows the general structure of the platform.

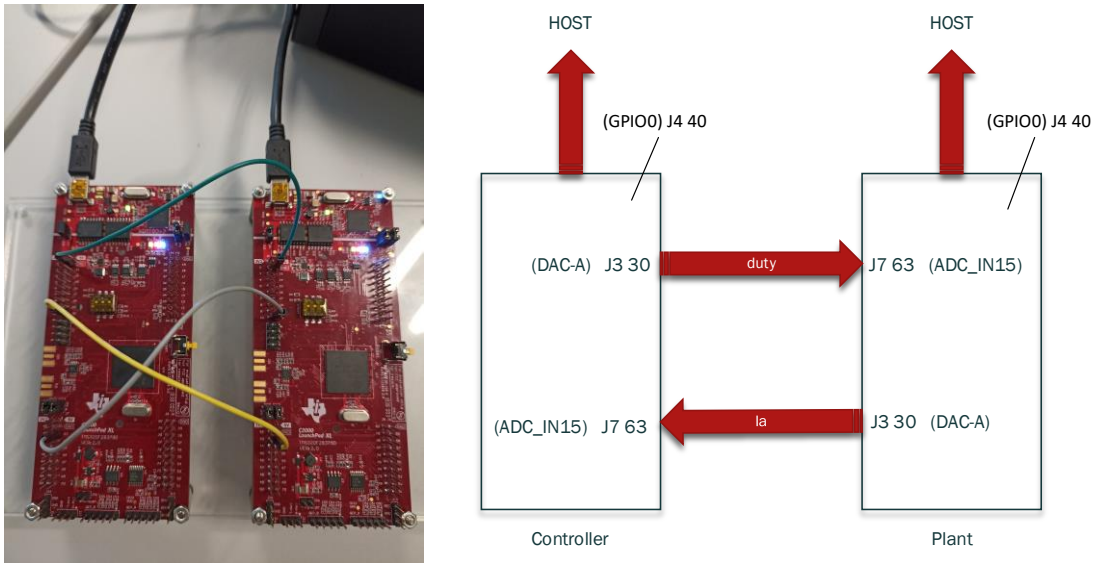

Figure 9 Hardware design

The first thing is to connect the GNDs of the boards so that they have the same reference.

One board runs the control part receiving the current from the other board with the use of the ADC and DAC. By defining the corresponding blocks in the simulation, we can send the duty cycle to the other plant from the DAC-A of the control and the ADC-IN15 of the plant. In turn, several variables are sent to the HOST program for analysis and monitoring.

The other runs the plant simulation, with the DC motor and mechanical system models. Upon receiving the duty cycle from the control board and doing the whole process, the motor current is obtained, and the same DAC and ADC bridge is made. Like the control, several variables are sent to the HOST to compare the results of the plant with those of the control.

Then, the HOST program running on the PC will send variables such as the reference current and the reset value of the integrators in order to manipulate the two simulations in real time from the same model. By obtaining the internal variables calculated by the cards, it is possible to visualize the evolution of the current or duty cycle directly on the HOST; being able to see the differences between the control and the plant, caused by factors such as offset.

3.2 Software design

3.2.1 Real time simulation structure

To simulate the two parts in real time, and to ensure the correct order of calculations in the simulations, we use a structure composed of enable subsystems that need to receive

data from a previous subsystem to start working. Three subsystems are sufficient for what is needed in this case (see Figure 11).

The first and the last one manages a state change signal for a GPIO, and then observe the execution time between the first subsystem and the last one by means of an oscilloscope (more details in section 3.2.3). Also in the first one, the signal that enables the execution of the second subsystem is created. This ensures that the tasks are executed in sequence.

The second contains the entire control/plant simulation explained above in the section 2 split in two just where the duty cycle and current are obtained (see Figure 10). In addition, SCI blocks are added that send data to the HOST outside the subsystem, since these run at different sampling times and an enable block does not allow different execution times.

In addition, in one of the models, a cycle counter is added in the final subsystem, which increases with each execution. In this way, having the value of the counter in the HOST, it is possible to calculate the time that each card has been running and the results can be displayed correctly. It should be noted that the execution of the HOST is not in real time, so the variables monitored through the HOST are represented with a different time axis.

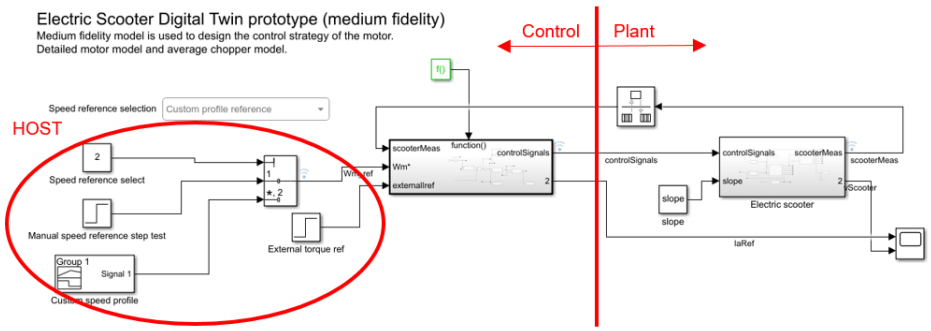

Figure 10 Simulation division for HIL implementation

3.2.2 MIL simulation adaptation for real time execution

The MIL simulation had to be adapted for real-time execution. Figure 11, Figure 12 and Figure 13 show the models prepared for real-time execution.

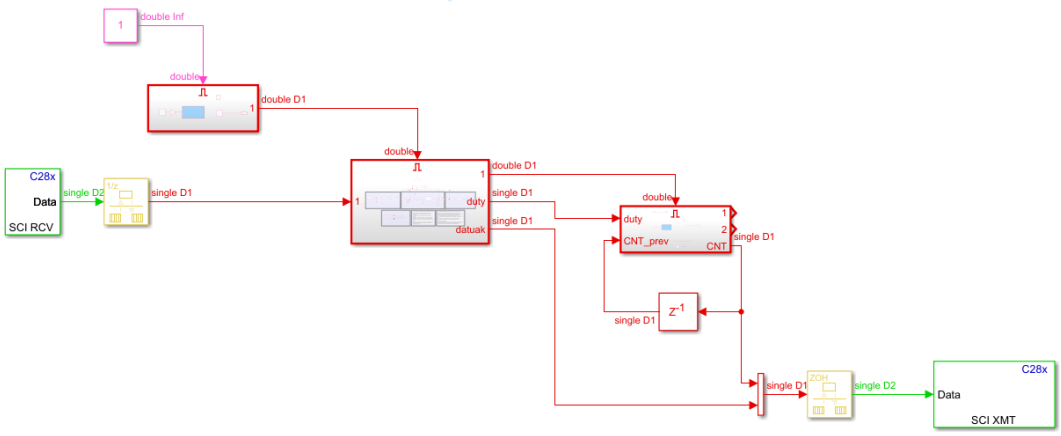

Figure 11 Real time simulation structure

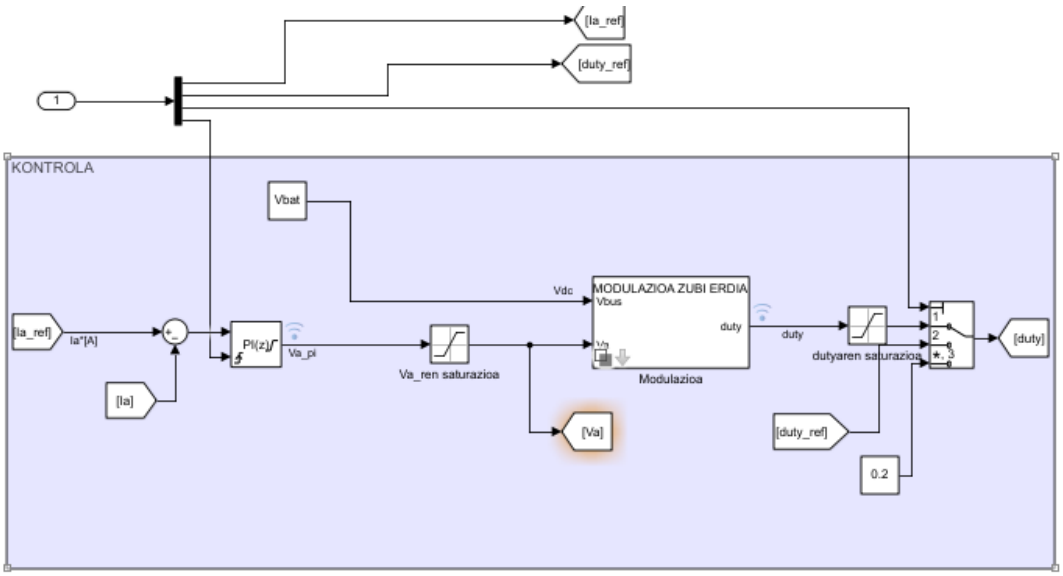

Figure 12 Current control structure

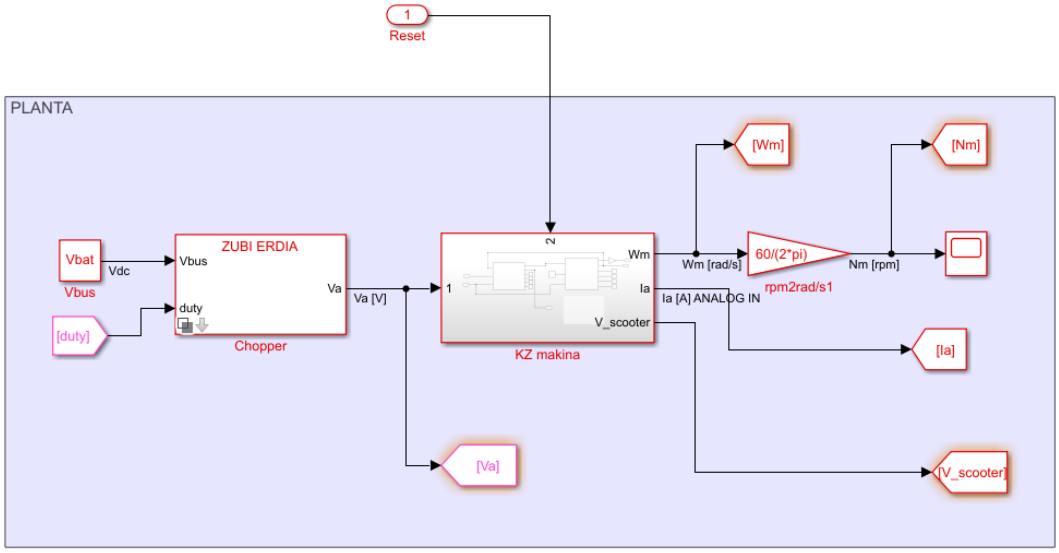

Figure 13 Plant structure

In these models the main adaptations have been the reading and writing of the signals through ADC/DAC channels and the resetting of the integrators.

The use of ADC/DAC channels has required their calibration to avoid measurement errors. The calibration of the offset of the analog IO has been done by collecting data in excel. Having the two boards; a constant is sent from the DAC of one to the ADC of the other; and seeing that the deviation was exponential, it has been decided to do the following: subtract the offset from the value at 0 and apply a gain that decreases this exponential rise, calculating the average gain necessary to approximate the data in a range of collected data.

Table 1 Board calibration

Board 1 DAC reference	0	50	100	200	500	1000	1500	2048	3000	3900	3934	4095
Boardo 2 ADC measurement	10	61	115	220	529	1040	1557	2118	3107	4061	4095	4095
Offset	10	11	15	20	29	40	57	70	107	161	161	0
Gain	0	0,98039216	0,95238095	0,95238095	0,96339114	0,97087379	0,96961862	0,971537	0,96867937	0,96272525	0,9630355	1,00244798

Average	0,96886025

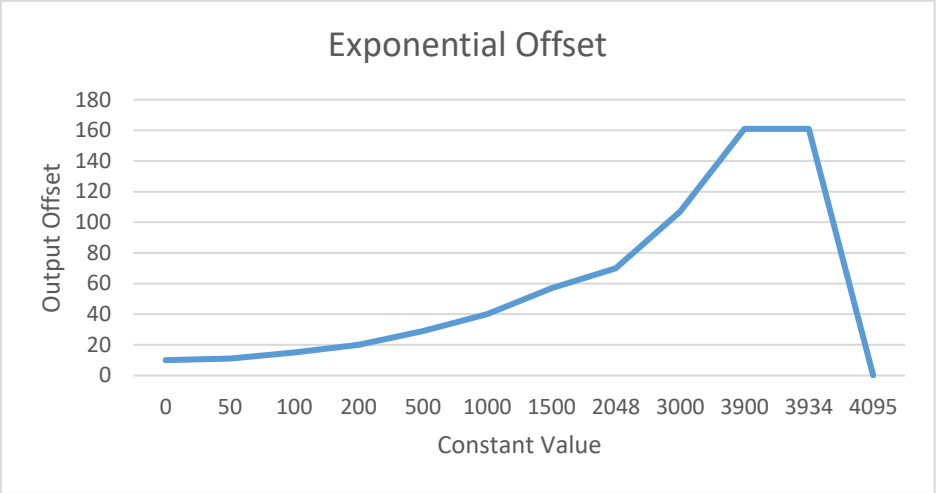

Figure 14 ADC offset exponentical increase

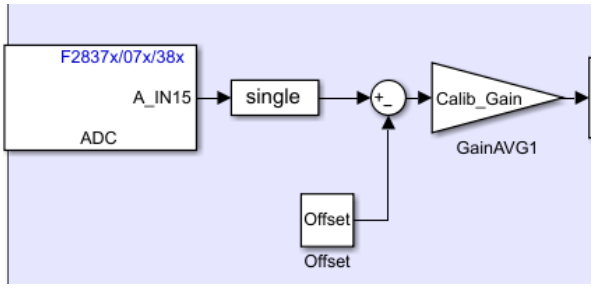

Figure 15 ADC calibration

Table 2 Calibration after correction

Board 1 DAC reference	0	50	100	200	500	1000	1500	2048	3000	3900	3934	4095
Board 2 ADC measurement	0,97	51,35	102,7	204,4	502,8	993,1	1490	2027	20965	3868	3909	3955

On the other hand, it has been seen that between tests the previous values of the integrators must be reset. If different tests are to be performed without having to reload and run the models again, the integrators of the controllers and the plant must be reset. Otherwise, the next test starts from the point where the previous one ended and wrong results are obtained.

3.2.3 Execution time measurement

As explained above, in the first and last enable subsystems the GPIO state is changed, and by measuring with an oscilloscope on the corresponding DO pin one can see the time it takes for the simulation to start and end the cycle, and also observe how often this process is repeated. The cycle repetition time should be the Sample Time defined in

the Solver to be correct, and logically; the more weight the simulation has, the more execution time it will need. Therefore, it is important to verify that the execution time of the tasks does not exceed the imposed execution step (see Figure 16).

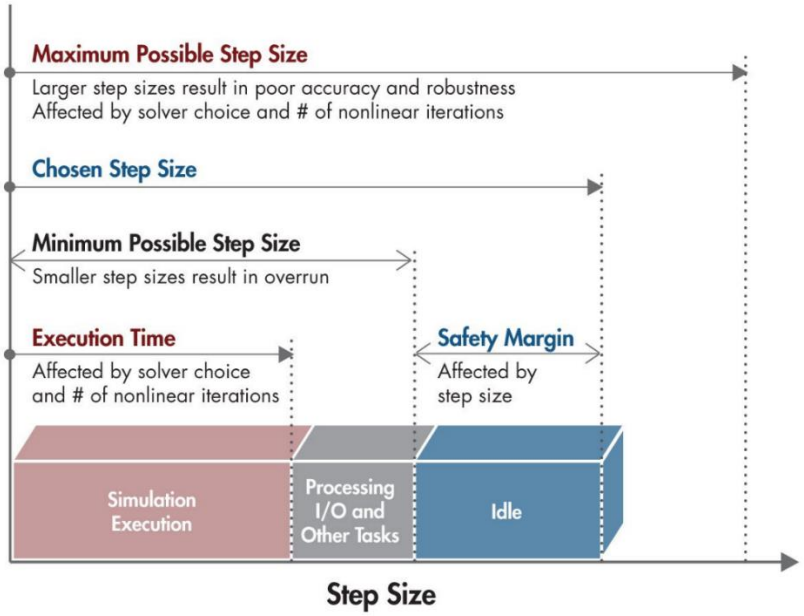

Figure 16 Real time execution time diagram (https://www.mathworks.com/company/newsletters/articles/real-time-simulation-of-physical-systems-using-simscape.html)

Below you can see the measurement of the execution time through a GPIO and the oscilloscope. In this case the tasks take 2.44 microseconds to be executed (Figure 18) which are the sum of the different processes of the simulation (plant/control model, analog inputs and outputs, sending signals to the GPIO...). Furthermore, having chosen a higher cycle time; 200 microseconds (see Figure 17), an idle time of 197.56 microseconds is obtained as a safety margin for processes that do not enter the boards, such as the SCI blocks that send data to the computer.

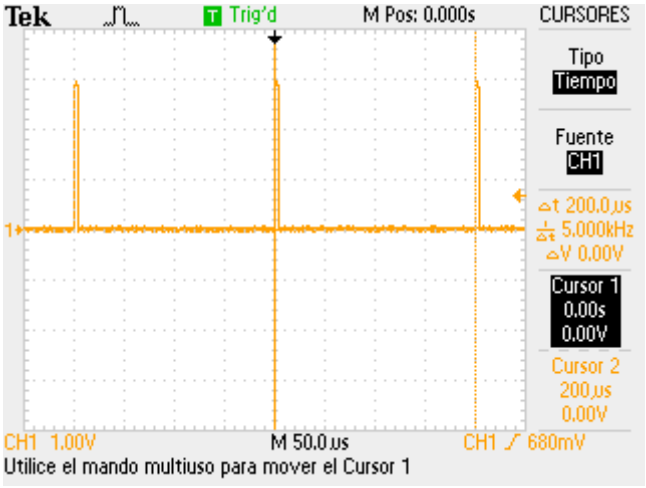

Figure 17 Step size test

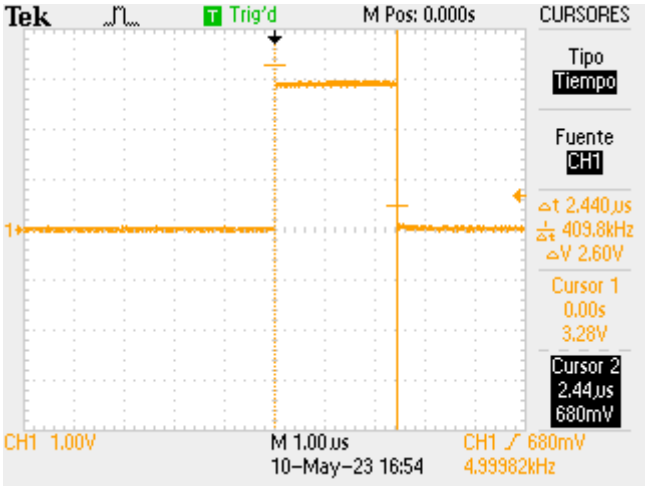

Figure 18 Execution time measurement

3.2.4 Monitoring

In order to visualize the internal variables of the boards, the HOST program explained above is used; it can send data with a Serial Send and receive them with a Serial Receive. The process is as follows: the data sent by the Serial Send will go out through the SCI Receive of the built simulations; and thus, obtain information to start the cycle. The data obtained are sent to the Serial Receive of the HOST through the SCI Transmit of the models.

For this process to be done correctly it is important to check which COM ports the cards are on, and the amount of data being sent or received. Since the HOST is running on a computer and there are two ports configured (one per card), to load the programs you must first connect a USB port that connects to the card that will receive the control. After loading the control model on the board, the USB port is disconnected (since the program is loaded on the board) and the board that will carry the plant is connected and loaded. To be able to see both boards active at the same time; once the ADC/DAC and GND are physically connected, the HOST program is executed while the two ports are connected to the computer (the control board will have one COM and the plant one).

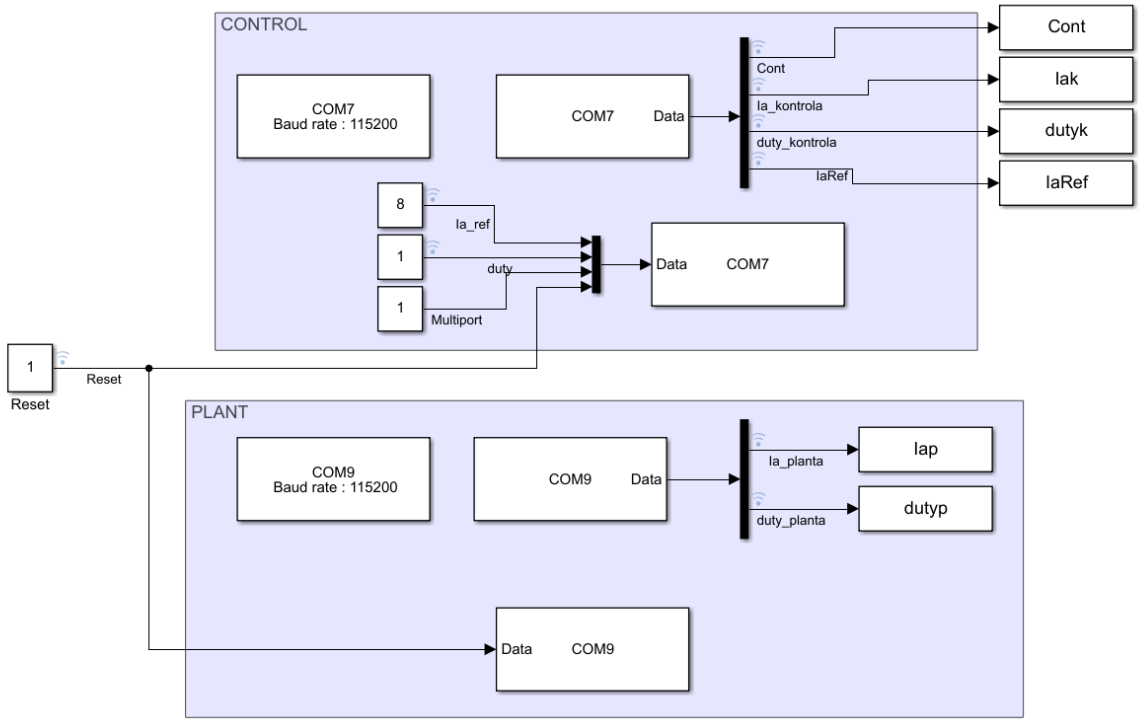

Figure 19 HOST program structure

It has been verified that this HOST configuration allows to monitor the variables of each card and compare them, but as explained in the section but the time axis for the real-time variables is not correct. The HOST is not running in real time and the internal variables of the cards represented on the HOST time axis show a behavior that is not real, as show in Figure 20.

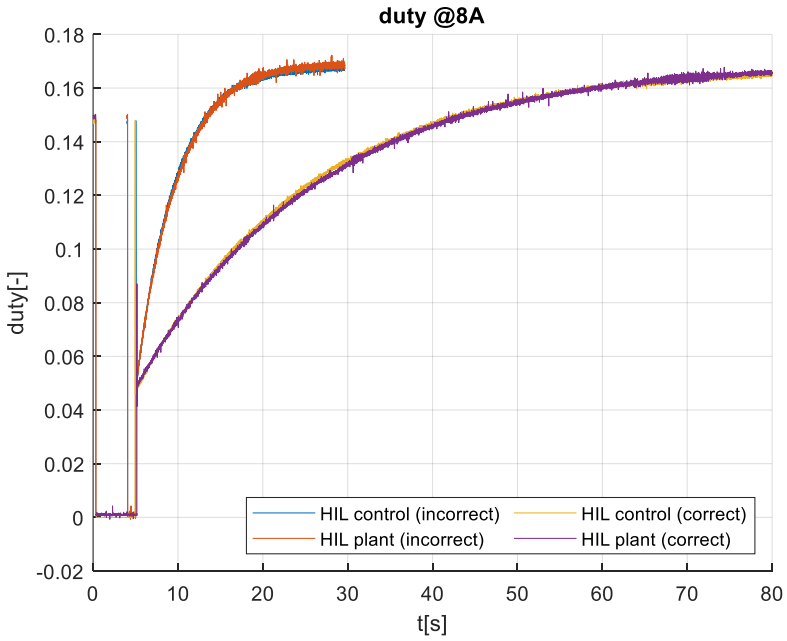

Figure 20 Different time axis for variables

To solve this problem, alternatives have been tried, such as:

- The use of Simulink Desktop Real time.
- Sending information in packets and Simulation pacing configuration.
- The external mode of Simulink.
- The implementation of a cycle counter.

In none of the cases we have obtained good results, so we have chosen to manually implement the counter mentioned above. This is how the results of section 4 were monitored.

4 HIL PLATFORM VERIFICATION

The HIL platform has been verified against the MIL platform results. Both the oscilloscope duty cycle response and the control and plant data have been collected for comparison with the MIL simulation. Comparing the duty cycle rise at different current references, the following results are obtained:

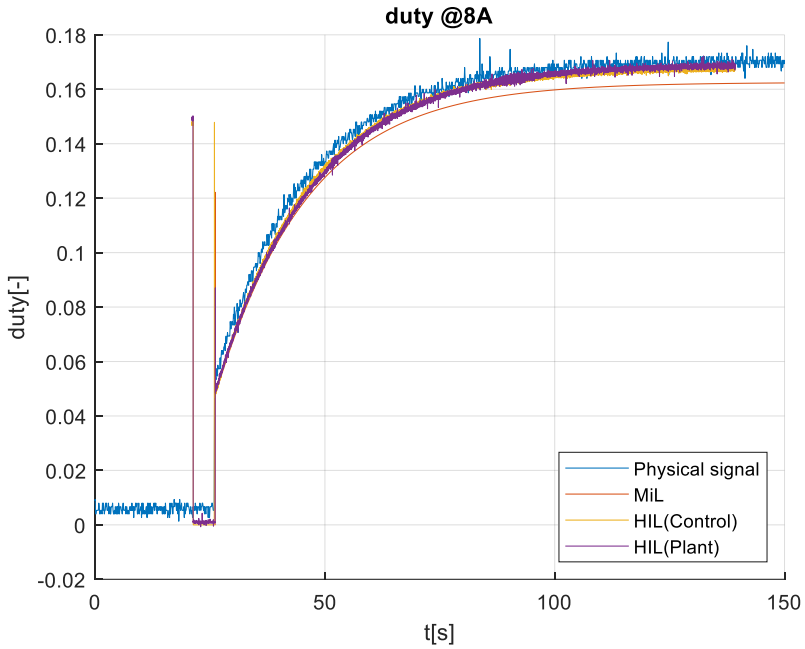

Figure 21 HIL simulation results for 8A

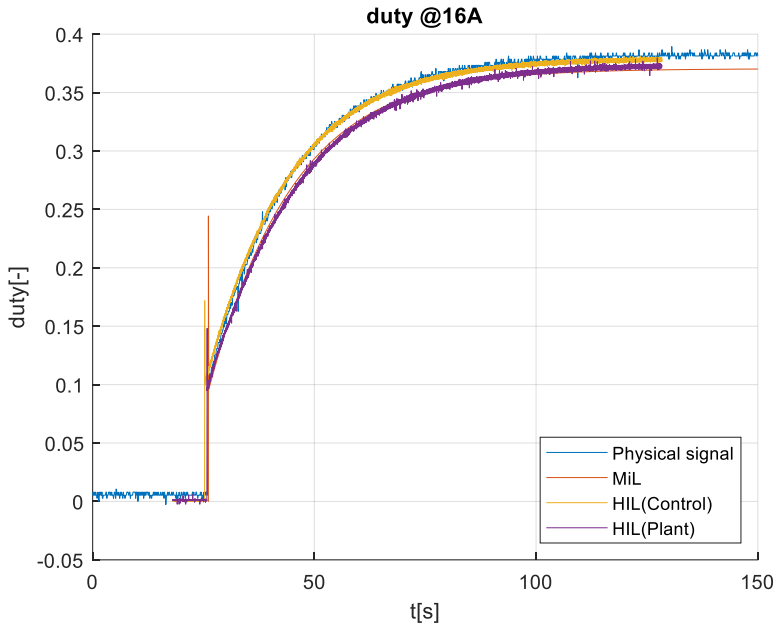

Figure 22 Duty cycle resutls with 16 A

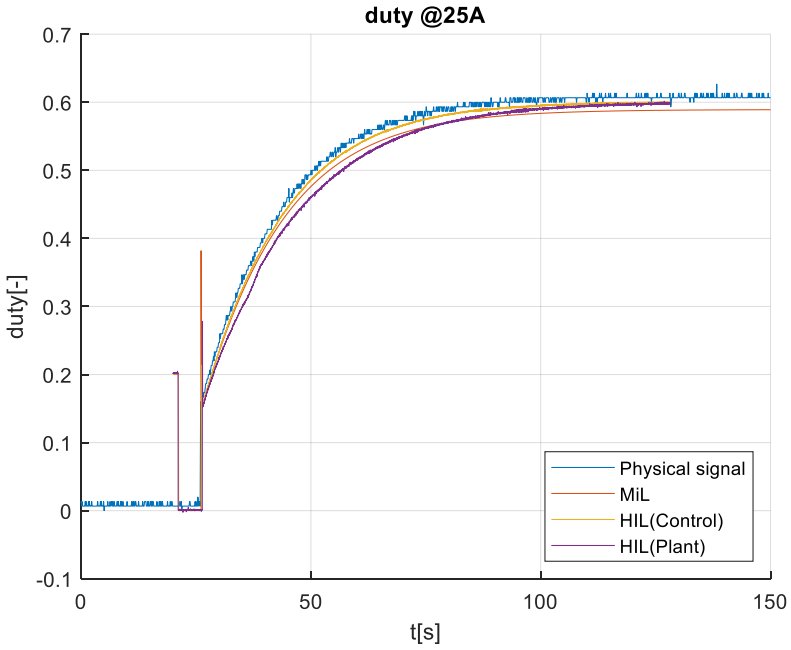

Figure 23 HIL simulation results for 25 A

In these results it is observed that the exponential increase is quite similar with a steady state error. This is due to negative factors such as offset. In steady state, in general, the HIL signals are higher than the MIL, which will later affect other data such as speed; but for the real application they are reliable since the error is of few decimal places.