Didactic Hardware-in-the-Loop Platform: A Low-Cost Open-Source Approach

S. Ojeda-Mancera, J. Carranco-Martínez, V. Sámano-Ortega, J. Martínez-Nolasco, C. Martínez-Nolasco, and M. Santoyo-Mora

***Abstract*—In evaluating and validating a physical system, real-time Hardware-in-the-Loop (HIL) emulation offers advantages such as time and cost reduction, fault prevention, and the ability to conduct validations in an environment similar to its final application. On the other hand, low-cost technologies such as microcontrollers, digital signal processors, and FPGAs have been employed to leverage the advantages of HIL emulation in the teaching process. This article describes the implementation of a low-cost, open-access HIL didactic platform for use in subjects such as differential equations, systems dynamics, and control systems, among others. The platform is based on a Raspberry Pi Pico development board and features a graphical user interface (GUI). In the GUI, the user can visualize graphs of the emulated system's variables and real-time animation of its state and export the acquired data to a comma-separated file. The functionalities offered by the platform make it an affordable tool that allows users to evaluate the response of a dynamic system, whether it is open-loop or closed-loop, without the need for classrooms or specialized equipment. Unlike similar works where HIL techniques with low-cost hardware are employed for educational purposes, the proposal in this work is more cost-effective and integrates the described GUI. All the files necessary for implementing the didactic platform are openly available in a public repository, including those needed for PCB manufacturing.**

Link to graphical and video abstracts, and to code: https://latamt.ieeer9.org/index.php/transactions/article/view/9619

***Index Terms*— Control systems, Education, HIL simulation, Open-source hardware, Open-source software**

I. Introduction

HARDWARE in the Loop (HIL) refers to a real-time simulation where a physical replica is used to emulate the behavior of a system or part of it. In an HIL simulation, the replica (emulator) replaces the system it emulates, allowing interaction between this replica and other physical systems; such interaction is typically a closed loop. If a dynamic system emulator interacts in a closed loop with a physical controller, the HIL simulation is called Controller-HIL or C-HIL. In a C-HIL simulation, the dynamic system and its emulator are indistinguishable from the controller, making it possible to evaluate the controller under similar conditions to those it will face in its final application. This feature of C-HIL reduces the design and validation of a controller and allows safety testing, detecting unforeseen failures prior to implementation, thereby preventing resource losses caused by unexpected malfunctions [1,2]. Commercial platforms dedicated to implementing HIL simulations that integrate specialized high-latency software, preloaded system models, and user-friendly programming interfaces; however, their high cost represents a disadvantage despite the benefits of HIL. Research has been conducted to implement this technique using low-cost hardware to solve the high cost of HIL platforms [3,4]. Since HIL involves hardware with relatively high latency, commonly used devices for its development include microcontrollers [5], digital signal processors (DSPs) [6], and Field Programmable Gate Arrays (FPGAs) [7,8].

An area where HIL has been explored in conjunction with low-cost hardware is education. The following are examples of such works: In [9], the development of a series of practices integrating Problem-Based Learning with HIL is described. A HIL platform based on a TMSF2837x DSP was developed to implement the practices. Case studies include emulating a Ball-Beam (B-B) system, a heat exchanger, and a SEPIC power electronic converter. In the case of the B-B system, an app in MATLAB was developed that includes an animation of the system, but this animation is performed as a preliminary step before embedding the model into the emulator. The authors concluded that the platform is low-cost and can be used outside the classroom, although an oscilloscope is recommended for analyzing the response of dynamic systems.

In [10], a low-cost HIL platform was developed to emulate power electronic converters as a teaching tool for the power electronics course. This platform employs the C-HIL technique by emulating the converters and implementing physical control. The converter models are simulated in MATLAB/Simulink, and an MCU TMS320F28335 with serial communication is used as the emulator's physical component, which is interconnected with the physical controller. The platform was evaluated by students, who concluded that it is affordable, easy to use, and helps them understand the theoretical concepts of the subject.

In [11], the development of a HIL platform as a teaching tool for Dynamic Systems is described. The platform was

The associate editor coordinating the review of this manuscript and approving it for publication was Roberto S. Murphy (*Corresponding author: Víctor Sámano-Ortega*).

S. A. Ojeda-Mancera, J. G. Carranco-Martínez, Víctor M. Sámano-Ortega, J. J. Martínez-Nolasco, C. Martínez-Nolasco, and M. Santoyo-Mora are with the Departamento de Ingeniería Mecatrónica, Tecnológico Nacional de México / Instituto Tecnológico de Celaya, Celaya, México (e-mails: 19030254@itcelaya.edu.mx, 19031250@itcelaya.edu.mx, victor.samano@itcelaya.edu.mx, juan.martinez@itcelaya.edu.mx, coral.martinez@itcelaya.edu.mx, and mauro.santoyo@itcelaya.edu.mx).

developed with a Texas Instrument dual-core DSC TMS320F28379D. Both cores are used so that one emulates a dynamic system to be controlled, and the other executes the control. The platform is implemented in a PCB with the DSP, a digital-to-analog converter (DAC), and a display as a user interface that shows specific system data. An oscilloscope is used to evaluate the emulator's response. A methodology is proposed, but the results of its implementation are not mentioned.

Finally, the work presented in [12] does not refer to an educational application but is included since the developed HIL platform is similar to the previous ones. In this case, an Arduino DUE is used to emulate a dynamic system and another one as a physical controller, forming a C-HIL simulation. To validate the platform, an RLC circuit and a microgrid were emulated, and an oscilloscope was used to evaluate the emulator's response.

In Latin America and the Caribbean, although the enrollment of higher education students has increased in recent years, retention remains a concern due to a high dropout rate caused, among other factors, by a lack of financial resources [13]. With this in mind, this article proposes a didactic HIL platform (DHP) as an alternative to harness the advantages of HIL simulation in the teaching-learning process, but with low-cost and open-access hardware and software. The DHP emulates a physical system to evaluate its response to an input (open-loop) or in a closed-loop with a controller (C-HIL), making it useful for subjects such as differential equations, system dynamics, and control systems, among others.

Similar to [11], the developed platform employs multicore programming to execute parallel processes. In this case, the secondary process establishes serial communication with a Windows application that serves as a graphical user interface (GUI), allowing the user to visualize the evolution of the process variable and the input or control signal of the emulated system. Unlike [9-12], the GUI developed in this work includes graphs of the system variables and real-time 2D animation of the emulated system, aiming to improve students' understanding of the effects of physical elements and their behavior on the system's response. Additionally, the GUI can export data (process variables and input or control signal) as a comma-separated file for further analysis. By including data export functionality, there is no need for a signal acquisition system, such as an oscilloscope, which could contradict the goal of proposing a low-cost platform. This feature also allows users to conduct experiments on the system without requiring a laboratory or classroom equipped with specialized instruments. In summary, in addition to the inherent advantages of HIL simulation, the proposed platform offers the following:

- A low-cost platform with open-access hardware and software.
- Real-time animation of the emulated system.
- Graphs showing the evolution of the system's variables.
- The ability for users to export data as a comma-separated file.
- No need for laboratories or specialized instruments to conduct experimentation.

The article is divided into seven sections: section I corresponds to the introduction, section II provides a general description of the DHP, section III describes the HIL emulator, section IV details the GUI, section V discusses the costs of the DHP, section VI presents the results, and finally, section VII presents the conclusions and future directions for this work.

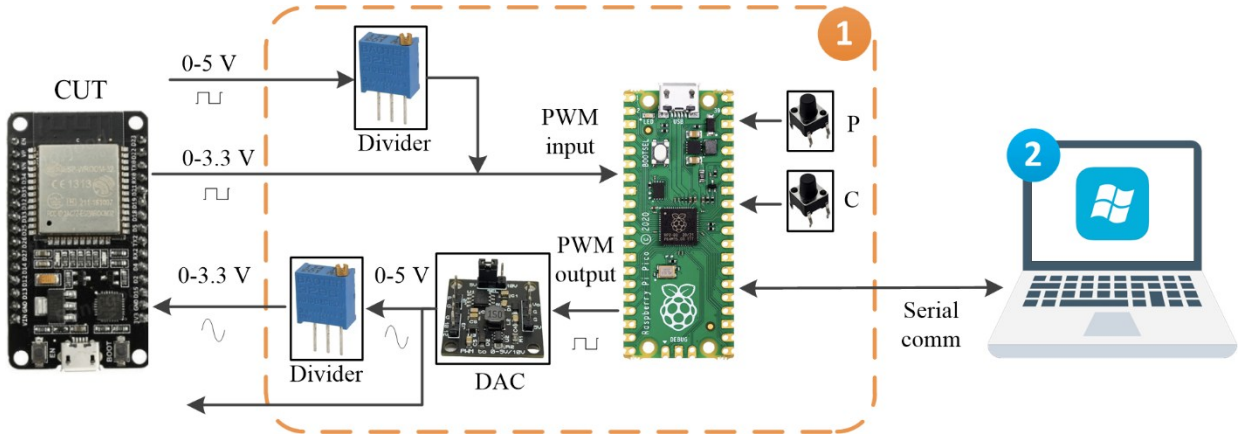


Fig. 1. Didactic Hardware-in-the-Loop platform schematic composed by a Controller under test (CUT), a HIL emulator (1), and a graphical interface (2).

II. HIL Platform Description

As shown in Fig. 1, the DHP consists of two main components: 1) A HIL emulator based on a Raspberry Pi Pico development board, with a digital input with PWM and an analog output ranging from 0 to 3.3 V and 2) a Windows application developed in Visual Studio as the GUI, which communicates with the HIL emulator via USB. In the HIL emulator the PWM input represents the input of the emulated dynamic system, while the analog output represents the process variable. The emulator can be tested both in open-loop

and closed-loop. To validate the DHP, a B-B system, a DC motor speed control system, and a DC motor position system were emulated, and their closed-loop behavior was evaluated using a controller under test (CUT). The CUT can be any digital development board, and in this work, an ESP32 DEVKIT V1 was employed, as shown in Fig 1.

III. HIL Emulator

The HIL emulator was implemented on a Raspberry Pi Pico development board, which includes an RP2040 microcontroller based on a dual-core Arm Cortex-M0+ processor. The emulated systems are SISO, so the emulator only includes one input and one output; the input signal is PWM (980 Hz), and the output signal is analog. Since the Raspberry Pi Pico does not have analog outputs, a module based on the GP8101 chip was used to convert a PWM output into an analog signal with levels ranging from 0 to 5 V. The CUT is a board with analog inputs of 0 to 3.3 V, so the emulator also includes a potentiometer as a voltage divider. Two buttons, labeled P and C in Fig. 1, are included. Button P induces an instantaneous disturbance to the emulated systems, allowing the CUT's performance to be evaluated during this event. On the other hand, pressing button C sets a fixed duty cycle on the PWM output signal, enabling calibration of the divider to achieve the desired voltage level.

To emulate the systems, both processor cores were utilized: the primary core executes the discretized dynamic model of the systems, while the secondary core manages the input and output signals, sending them to the GUI via USB. The system, model, multicore programming, and inter-processor communication are described below.

A. Emulated Systems

The B-B model was based on a didactic system that uses an MG995 servomotor and a ping-pong ball on an MDF beam. Fig. 2 illustrates the block diagram of the B-B, where: $\Theta_b{}^*$ represents the desired angular position of the beam (input to the emulator), Θ_s is the angular position of the servomotor, and R is the position of the ball on the beam (output of the emulator). Due to the low inertia of the ball-beam pair, its effect on the servomotor's response is negligible; therefore, the system was modeled as two transfer functions in series. The servomotor transfer function was taken from [14], and the B-B transfer function was derived as described in [15-19] using the parameters of the didactic systems.

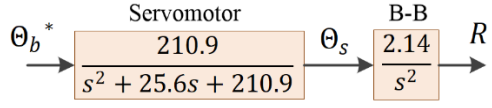


Fig. 2. Model of the emulated Ball-Beam system represented by two second-order transfer functions.

Discretizing the transfer function of the servomotor using the Forward Euler resulted in:

$$\theta_s(k) = c_1\theta_b{}^*(k-2) - c_2\theta_s(k-1) - c_3\theta_s(k-2) \quad (1)$$

where $c_1 = 210.90461T^2$, $c_2 = 25.65696T - 2$, $c_3 = 1 - 25.65696T + 210.90461T^2$, and T is the integration time used to solve the model in the processor. Similarly, by discretizing the ball and beam pair model the position of the ball on the beam in discrete time domain, $r(k)$, can be expressed as:

$$r(k) = c_1\theta_s(k-2) + 2r(k-1) - r(k-2) \quad (2)$$

where $c_1 = 2.14036T^2$.

The relationship between the DC motor's angular speed and its supply voltage was modeled as a first-order system with a -0.5 V to 0.5 V input dead-zone. On the other hand, the system that emulates the DC motor position is composed of a first-order system with an integrator at the output. The emulated system for the speed control case is presented in Fig. 3, whereas Fig 4 shows the system for the position control case.

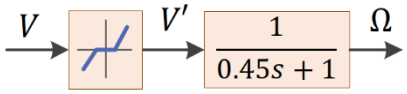


Fig. 3. Model of the emulated DC motor speed system represented by a dead-zone and a first-order transfer function.

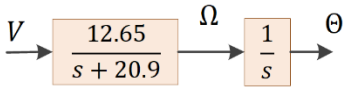


Fig. 4. Model of the emulated DC motor position system represented by a first-order transfer function and an integrator.

For both cases V is the supply voltage of the DC motor and Ω is the angular speed. In the position system Θ is the angle of the shaft. And for the speed system V' it is the voltage with the dead zone defined as follows:

$$V' = \begin{cases} \frac{5}{4.5}V + \frac{2.5}{4.5} & ; \quad V \le -0.5 \\ 0 & ; \quad -0.5 < V < 0.5 \\ \frac{5}{4.5}V - \frac{2.5}{4.5} & ; \quad V \ge 0.5 \end{cases} \quad (3)$$

By discretizing first-order transfer function of Figs. 3 resulted in:

$$\omega(k) = c_1 v(k) - c_2\omega(k-1) \quad (4)$$

where $c_1 = T/0.453576$, and $c_2 = (T/0.453576) - 1$. Similarly, discretizing the second-order resultant of the series of the two transfer functions in Fig. 4 gives:

$$\theta(k) = c_3 v(k-2) - c_4\theta(k-1) - c_5\theta(k-2) \quad (5)$$

where $c_3 = 12.650058\,T^2$, $c_4 = 20.8968 * T - 2$, and $c_5 = 1 - 20.89684 * T$.

To emulate the models described by (1) to (5), during each execution cycle of the primary core of the HIL emulator, the state of the input signal is evaluated by measuring its ON time in µs, based on falling and rising edges. The ON time is normalized to obtain the input variable values ($\theta_b{}^*(k)$ or $v(k)$), which is used to compute models' equations. The value obtained for the output variable ($r(k)$, $\omega(k)$ or $\theta(k)$) is

normalized to ON time values that determines the duty cycle of the output signal.

The values of input and output signals are stored in a shared register with the secondary core, and finally, the memories of previous values of the variables from (1) to (5) are updated. Following this algorithm, an experimental minimum integration time of 100 µs was obtained for the HIL emulator. The latency implies that the system model is executed 10,000 times per second. Considering that a dynamic system has a settling time of four time constants and its model is solved at least 250 times per time constant, the proposed development board could emulate a system with a minimum settling time of 100 ms. However, the PWM-to-voltage converter has a response time approximately 1 ms, which was experimentally determined for different PWM frequencies. Therefore, for faster systems, its response would exhibit poor amplitude resolution.

Table I shows the constants resulting from the discretization of the system dynamics, as well as the integration and sampling times of the HIL emulator.

TABLE I
CONSTANTS OF THE SYSTEM DYNAMICS

Equation	Constant	Value
(1)	c_1	2.1090×10^{-6}
	c_2	-1.9974
	c_3	0.9974
(2)	c_1	2.1404×10^{-8}
(4)	c_1	2.2047×10^{-4}
	c_2	-0.9998
(5)	c_3	1.2650×10^{-7}
	c_4	-1.9979
	c_5	0.9979
Integration time (T)		100 µs
Sampling time		100 µs

B. Controller Under Test

To control the B-B system, the PD controller described as

$$u(t) = k_p e(t) + k_d \frac{de(t)}{dt} \quad (6)$$

was implemented, in this model u is the control signal, e is the error, k_p is the proportional gain, and k_d is the derivative gain. Subsequently, the Backward Difference Method [20,21] was used to discretize (6) resulting in:

$$u(k) = k_p e(k) + k_d \frac{e(k) - e(k-1)}{T_c} \quad (7)$$

where T_c is the integration time of the CUT. By grouping common terms of (7), the discrete model of the PD controller results in:

$$u(k) = \left(k_p + \frac{k_d}{T_c}\right) e(k) - \frac{k_d}{T_c} e(k-1) \quad (8)$$

For speed control, a PI controller was adopted, applying a similar discretization its model results in:

$$u(k) = u(k-1) + \left(k_p + k_i T_c\right) e(k) - k_p e(k-1) \quad (9)$$

where k_i is the integral gain.

Finally, for the position control, a P controller was used, whose discrete function is given by:

$$u(k) = k_p e(k) \quad (10)$$

The models in (8), (9) and (10) were embedded into the CUT for each of the systems, using an integration time of 1 ms to minimize discretization errors. On the other hand, the output update time was set to 2 ms, that is, for every two cycles of the CUT the control signal is updated only once. The output update time was set to this value for two reasons: First, the CUT's PWM output frequency was set to 980 Hz, as this is the predefined frequency for pins 5 and 6 on the Arduino UNO and similar boards, which are widely used, this implies that a change in its duty cycle takes approximately 1 ms to settle. Second, the analog output of the PWM-to-voltage converter takes approximately 1 ms to stabilize when a change is applied to the input duty cycle. Thus, the selected time allows the CUT's output PWM signal to stabilize (≈1 ms) and ensure the output of the PWM-to-voltage converter gets stable (≈1 ms).

Table II shows the controller gains used for each case, as well as, the CUT's integration and output update time.

TABLE II
CONTROLLERS GAINS

Controller	Gain	Value
PD	k_p	3.2316
	k_d	1.4913
PI	k_p	0.0013
	k_i	0.0056
P	k_p	16.6742
Integration time (T_c)		1 ms
Output update time		2 ms

C. Multicore Programming and Communication

Fig. 5 shows the flowchart of the emulator's multicore programming and communication. Since the system's input and output data must be received in the GUI, utilizing both cores of the development board's microcontroller is necessary. This approach allows serial communication without compromising emulation performance. The primary core (Core 0) is responsible for reading the input or control signal and solving the system model, while the secondary core (Core 1) manages data transmission to the GUI.

When Core 1 is initialized, an event is triggered using the interrupt handler instruction to ensure it can manage asynchronous tasks without interfering with other operations. This interruption is triggered immediately when Core 0 sends data through shared memory. Additionally, a continuous loop

generated by the tight_loop_contents() function maintains Core 1 in a low-activity state. In this way, Core 1 remains on standby and is only activated upon detecting an interruption in the shared memory, thus avoiding unnecessary resource usage. Core 0 operates with an integration time of 100 µs and, during each cycle, verifies if 17 ms (≈60 fps) have passed. When this condition is met, Core 0 updates the values in the shared FIFO with Core 1, triggering Core 1 to send the data to the GUI.

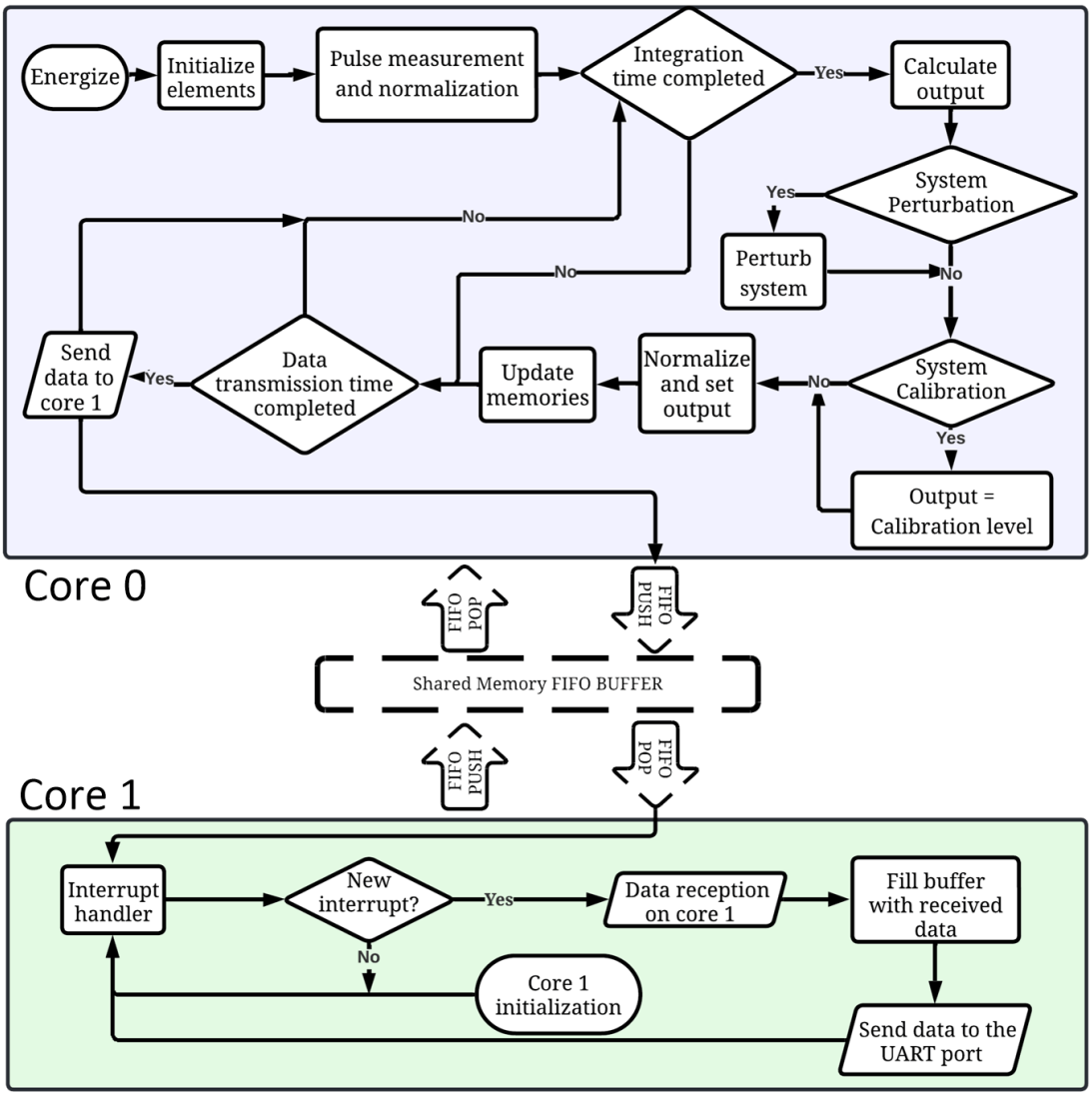


Fig. 5. Multicore programming and communication flow chart.

IV. User Interface

The GUI shown in Fig. 6 was programmed in C# using Windows Forms from the .NET Framework library. It consists of five main parts: a data plotting area, a system animation area, a button for the data exporting functionality, serial communication controls, and a system selector that modifies the interface based on the emulated system.

The interface design is responsive; when the window is resized, the elements maintain their positions and scale to the new dimensions. Each component, along with its functionality and communication principles with the HIL emulator, is detailed below.

A. Data Plotting Area

A time-series graph is implemented so that the oldest point is automatically removed when the maximum number of allowed points is reached. The data displayed in the graph is not stored on the computer, ensuring the efficient use of system resources and maintaining an up-to-date and dynamic representation of the monitored variables. This way, the user only requires their computer and emulator to perform laboratory practices and experiments. The plotted variables include the input and output of the emulated system, transmitted from the emulator via USB.-The data plotting area includes plot legends to identify each graph. The range of the input and output signals depends on the emulated system, so

the amplitude of this graph is adjusted based on the system selected in the "System selector" ComboBox.

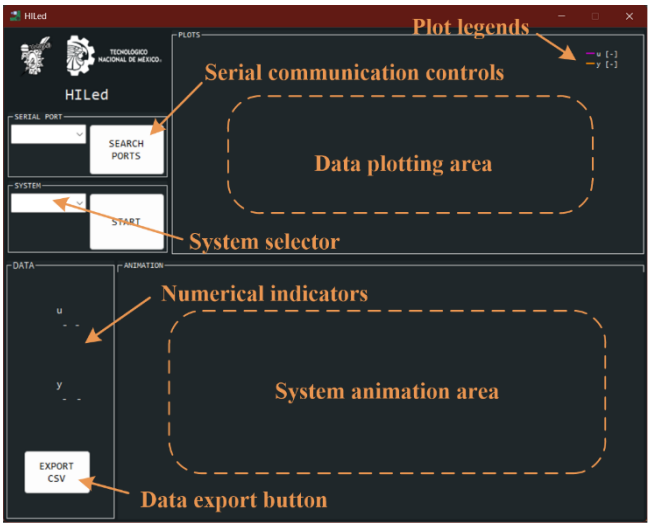


Fig. 6. Structure of the platform's Graphical Interface indicating all its parts.

B. System Animation Area

This area, a real-time 2D animation of the emulated systems is visualized. For the B-B system the animation includes a line that tilts around a central pivot, representing the beam, and a circle that moves along this line, representing the ball. For both the velocity and position systems, the output variable is displayed using a gauge, with a range customized according to the specifications of the emulated system. Animation updates are managed by a dedicated class, enabling scalability and future integration of additional systems.

The animation is performed under a canvas event where the paper, pen, and figures are initialized with their position coordinates. Each time data is processed, the animation is updated. The animation updates at a 60 fps rate based on the values sent to the GUI by the emulator. Additionally, numerical indicators are included to the left of the animation, presenting real-time information about the input and output signals of the emulated system. The names and units of these indicators also change depending on the selected system.

C. Data Export

Data export from the graph is performed using the "EXPORT CSV" button. Data capture operates similarly to that of an oscilloscope. Before using this function, serial communication must be stopped so that the time series graph remains static and captures the desired behavior. When the button is pressed, the points corresponding to the controlled variable (meters, degrees or rpm; depending on the system selected) and the time in seconds are saved. The user selects the destination where the file is stored, facilitating the management of the exported data.

D. Serial Communication

The interface includes two buttons and a ComboBox to control communication with the emulator. The "SEARCH PORTS" button identifies all available COM ports to establish a connection and adds them as selectable elements of the ComboBox to select the one corresponding to the emulator. Once the appropriate port is selected, the "START" button initiates communication with the emulator. If communication is active, the "START" changes to "STOP," allowing the communication with the emulator to be stopped to end an experiment or to export data.

Fig. 7 shows the flowchart of the GUI programming corresponding to data acquisition from the emulator, graphing, and animation. The development board and the interface are communicated by activating the DTR (Data Terminal Ready) signal, which indicates readiness to establish serial communication. The computer sends the DTR signal to request communication, and the board responds, enabling data transmission. The data received in the interface is stored in a buffer and then segmented for graph and animation processing.

V. Costs

Table III shows the cost of the components necessary to construct the HIL emulator, including a PCB to mount them.

TABLE III
COST OF THE DEVELOPED PLATFORM

Component	Cost (USD)
Raspberry Pi Pico	$4.00
Push button (x2)	$0.20
PWM to voltage converter	$5.00
Potentiometer (x2)	$0.30
Headers	$0.09
Sockets	$1.50
PCB	$1.30
Capacitor	$0.10
Total:	**$12.49**

On the other hand, Table IV shows the cost of the devices used, or similar ones, for the emulator in [9-12] and the cost of the platform proposed in this work. The costs shown were consulted in December 2024.

TABLE IV
COSTS OF THE DEVICES USED IN SIMILAR WORKS

Component	Cost (USD)
Arduino DUE	$53.74
LAUNCHXL-F28379D	$67.42
TMS320F28335ZJZS	$33.35
Proposal	$12.49

VI. Results

Initially, to validate the functionality of the DHP, two closed-loop experiments were conducted with the CUT. The emulator's output signal was acquired using an oscilloscope and compared with a software simulation in MATLAB/Simulink; the Simulink block diagrams used for these experiments are provided in [22]. The simulation includes two responses: one from the continuous B-B model and another from the discrete model.

The first-experiment was conducted with a proportional (P) controller. The beam in the real system includes limits at its ends, so even when the system with the P control is theoretically unstable, its output oscillates continuously instead reach unmeasurable values. Because of this, the emulator system also includes output limits. In Fig. 8, the comparison of the experimental result of the B-B response with this controller is presented.

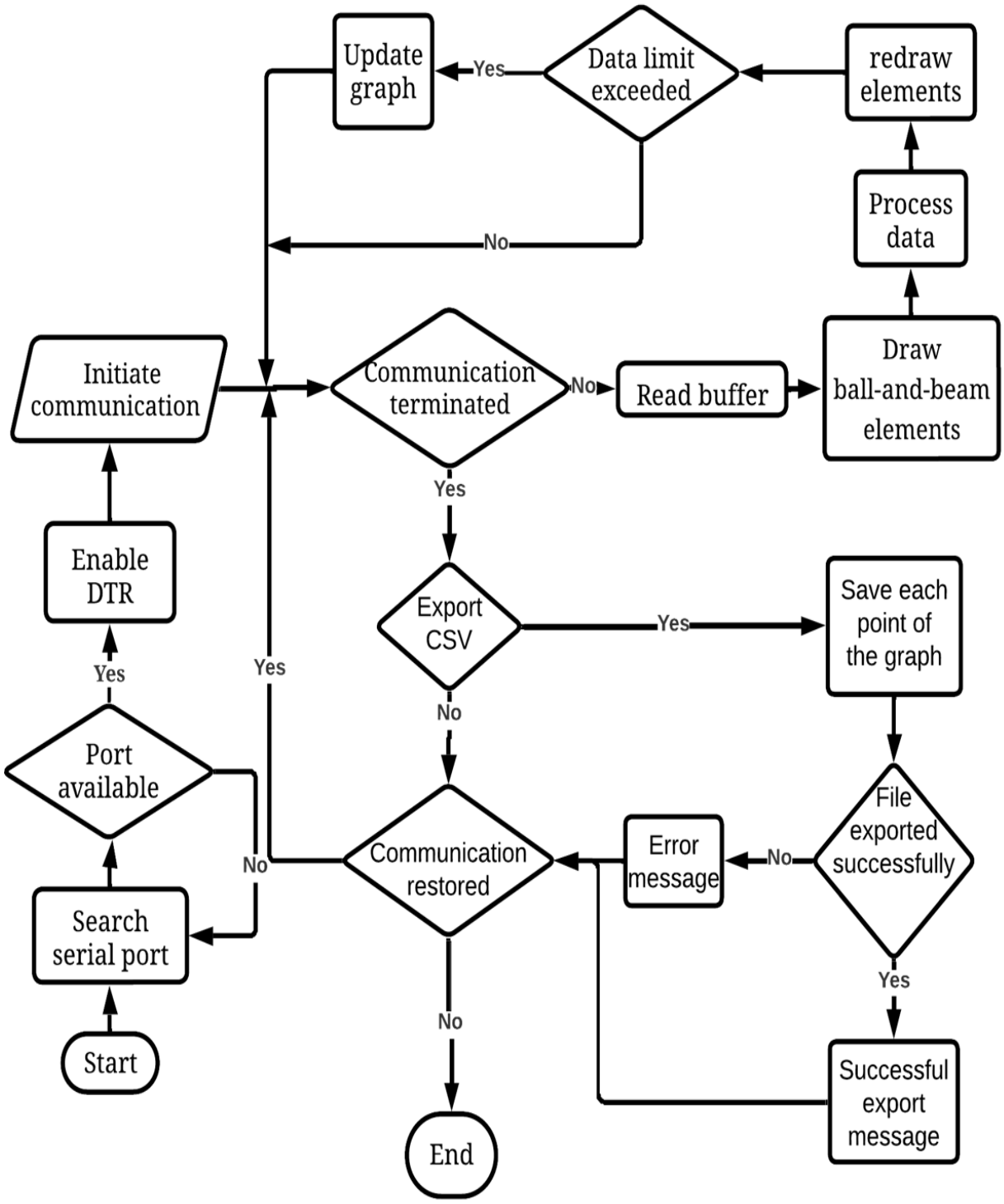


Fig. 7. Flow chart of communication and animation in the user interface.

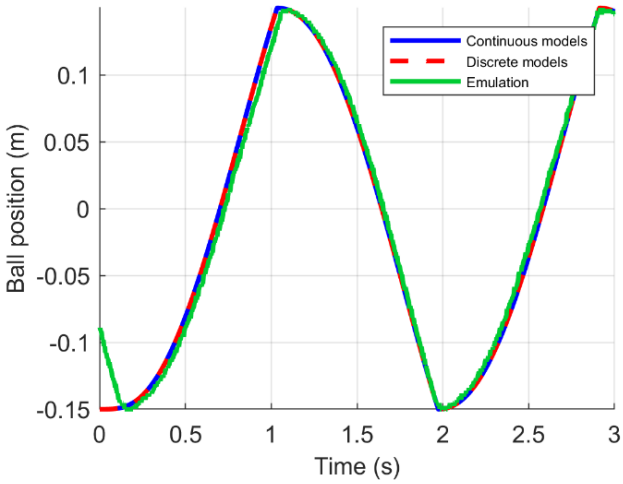


Fig. 8. Comparison between the emulated and simulated responses of the closed-loop B-B system with a P controller.

The second experiment was carried out using the PD controller, fig. 9 shows the B-B response in this experiment with an initial condition of 0.15 m.

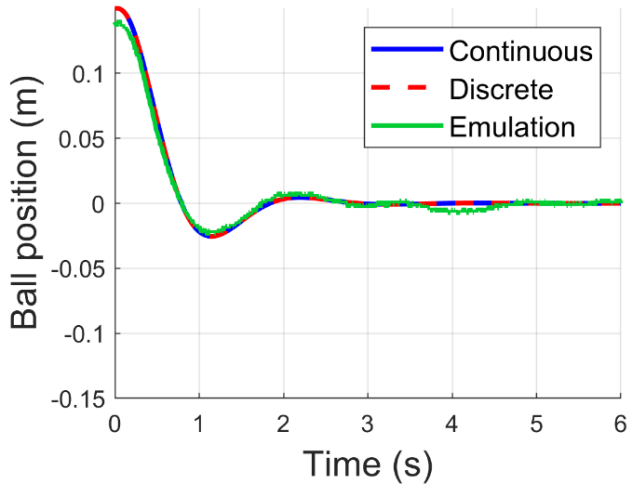


Fig. 9. Comparison between the emulated and simulated responses of the closed-loop B-B system with a PD controller under a 0.15 m initial condition.

The platform was also evaluated with the velocity and position control systems. The simulation considers the controller to operate in discrete time, with a 1 ms integration time and 2 ms output update time, matching the CUT, and that its output is saturated; this characteristic was also considered in the CUT. Fig. 10 shows the closed-loop speed response of the DC motor with the PI control, while Fig. 11 presents the closed-loop position response of the gear motor with the P control. In all cases, the figures display the response from the continuous model simulation in blue, the discrete model simulation in red, and the response obtained in the DHP in green.

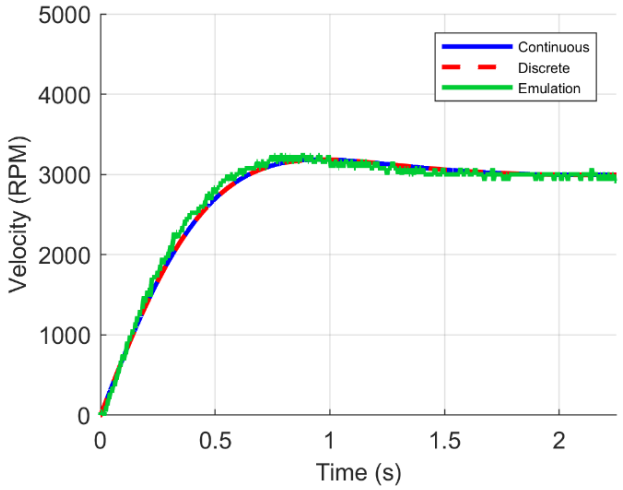


Fig. 10. Comparison between the emulated and simulated responses of the closed-loop motor speed system with a PI controller under a step input of 3000 RPM.

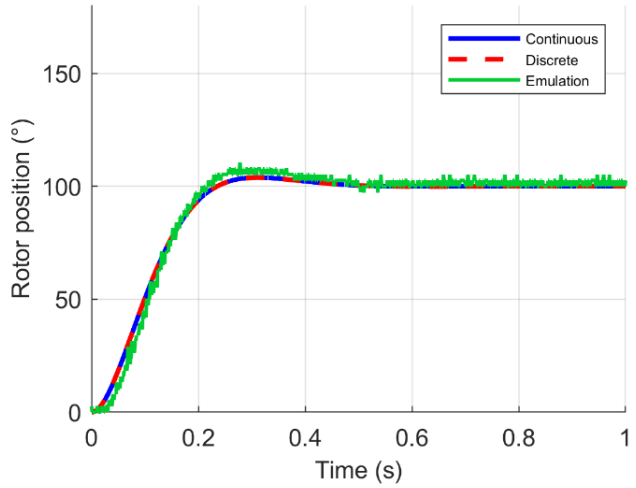


Fig. 11. Comparison between the emulated and simulated responses of the closed-loop motor position system with a P controller under a step input of 100°.

The Simulink diagrams used to validate the DHP responses are shown in Figures 12 to 14. Fig. 12 shows the closed-loop position system diagram, Fig. 13 shows the closed-loop velocity system diagram, and Fig. 14 shows the closed-loop B-B system diagram; In all three cases, the simulation of the continuous model is shown in the upper part of the figure and that of the discrete model in the lower part. In these simulations, the block diagram execution time is 100 µs, while the controllers were implemented as triggered subsystems that execute whenever a rising edge occurs on a square signal with a period of 1 ms.

In Fig. 15, the GUI displays the B-B's response with the PD controller to an instantaneous disturbance of 0.15 m. Fig. 16 shows the GUI with the step input response of the velocity control system, while Fig. 17 presents the GUI with the step input response of position control system.

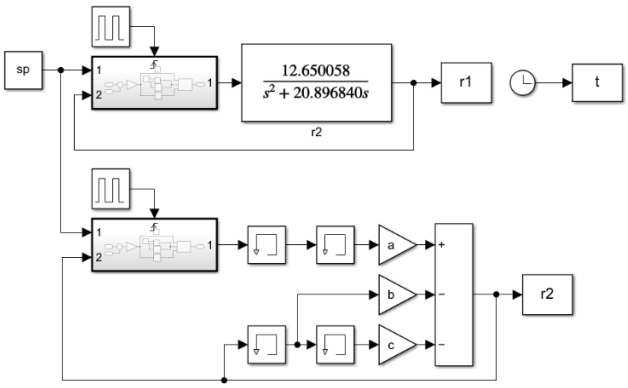


Fig. 12. Diagram to simulate the closed-loop motor position system with a P controller.

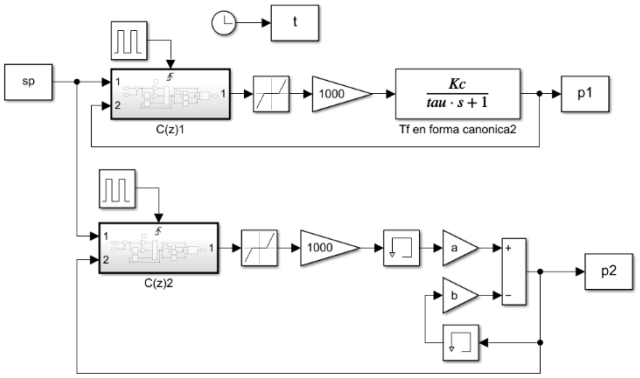


Fig. 13. Diagram to simulate the closed-loop motor speed system with a PI controller.

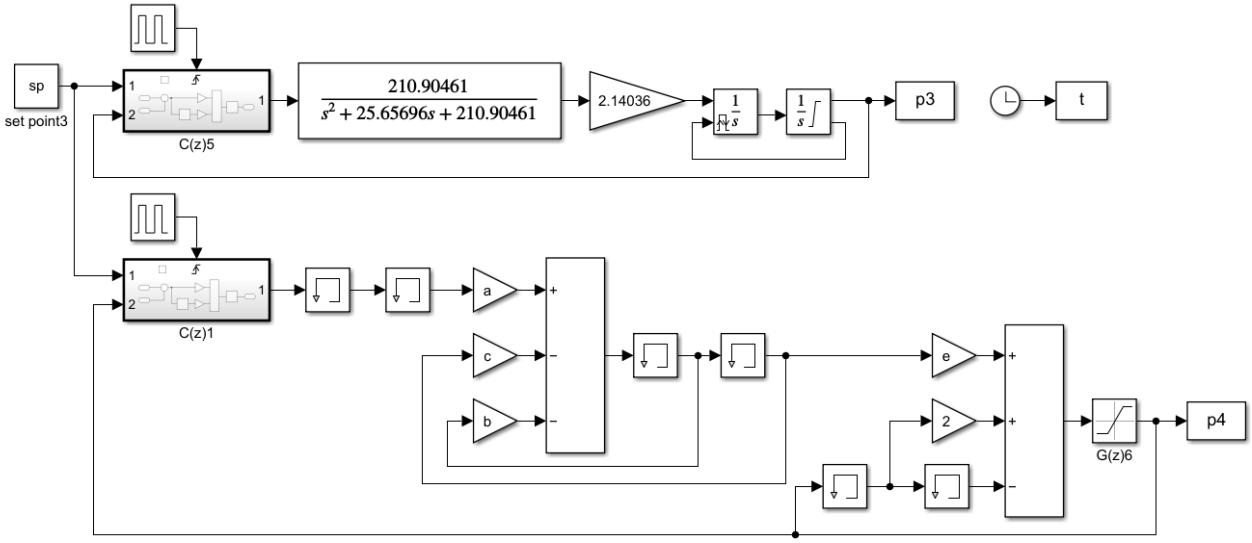


Fig. 14. Diagram to simulate the closed-loop B-B system with a PD controller.

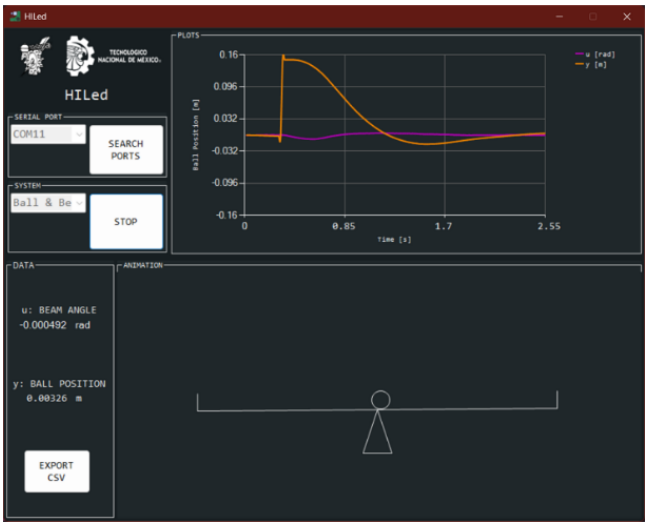


Fig. 15. Graphical user interface showing the B-B system.

Fig. 16. Graphical user interface showing the DC motor speed system.

Finally, Fig. 18 shows the HIL emulator built for the experiments, and the materials used to construct it are listed in Table III. While Fig. 1 shows only one potentiometer, the constructed emulator includes two. This addition was made to enable the emulation of MIMO systems in the future, allowing compatibility with CUTs featuring I/O voltages of either 5 V or 3.3 V. These additional features are further detailed in the results section. All the files necessary to reproduce the DHP, including those required for the PCB shown in Fig. 18, are available in [22].

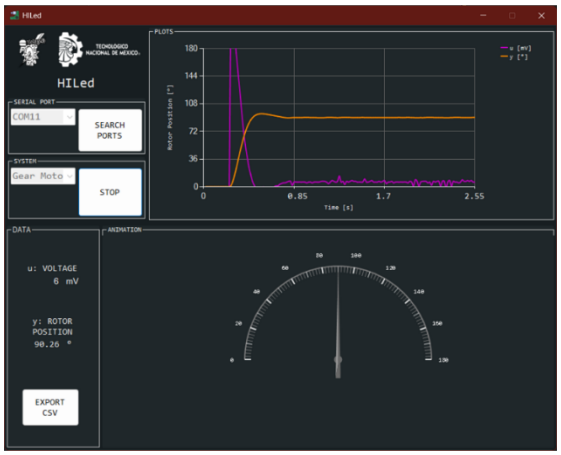


Fig. 17. Graphical user interface showing the DC motor position system.

Fig. 18. Ensembled PCB of the developed emulator.

VII. Conclusion

The comparison between the DHP results and software simulations validates the platform's accuracy in emulating a dynamic system's behavior. This accuracy is achieved using low-cost hardware reaching a latency of 10,000 operations per second. As discussed in Section III, this latency theoretically allows the emulation of systems with a minimum settling time of 100 ms. The systems were used in this article to demonstrate the functionality of the DHP were B-B, gear motor and DC motor, but it is not the only system that can be emulated. With the obtained latency, the platform can be used to emulate a wide range of systems, mainly mechanical, that are relatively slow. The integration time of 100 µs was experimentally determined for a fourth-order SISO system; however, when performing the same experiments with a first-order SISO system, an integration time of 46 µs was obtained; this implies that depending on the system, the DHP latency could be higher.

The emulator's latency was determined by emulating models without nonlinearity and uncertainty. Only in the case of the speed system was a dead-zone emulated which had no effect on the emulator's integration time. The fact that the integration time was not affected is due to the simplicity of the model of this non-linearity and to the fact that the speed system is a first-order system which, as mentioned earlier, can be emulated with an integration time of 46 µs, less than half the integration time set for the emulator for all systems. The main characteristic of HIL is the real-time solution of the models, which requires high latency. This feature is usually achieved using devices such as FPGs or DSPs. The use of processors in HIL limits latency since software is used instead of hardware. Emulating other more complex non-linearity and uncertainty involves higher computational costs, which reduces latency. At this point, a contradiction may arise: a more accurate model that is solved with lower latency, in turn, generates a greater error. In some cases, the latency could decrease to the point where the model no longer runs in real-time. By emulating only linear models, high latency is achieved even when using a processor. At the same time, using a processor significantly reduce the cost of the emulator, which is the main objective of this research, considering the economic situation of the Latin-American student community, briefly discussed in the introduction section. In other words, not emulating nonlinearity and uncertainty is a limitation directly linked to the low cost of the platform. Despite this, linearizing nonlinear models is a widely accepted and applied practice. Additionally, emulating a system using a model from literature allows for experimentation with a wide variety of systems, even when they are not physically available.

Although systems like those presented in [9-12] are proposed as low-cost alternatives, evaluating the results of these platforms involves using an acquisition system. In contrast, the platform described in this article does not require such things avoiding extra costs not considered in similar works. Even without this advantage, Table IV shows that the proposed platform is a more affordable alternative.

To validate the DHP, a SISO system with a PWM input and an analog output was emulated. However, the development board used has multiple digital I/O and analog inputs. The PCB design of the HIL emulator includes additional inputs and outputs, enabling the emulation of MIMO systems with different types of I/O in the future.

Unlike [9-12], our DHP includes a real-time animation of the emulated system. Future work will evaluate the effect of including the animation on aspects such as understanding the system. Additionally, more systems will be embedded to create a catalog. This way, using the same DHP, different systems and their characteristics can be studied throughout a

course. Among the systems planned to be embedded are a one-degree-of-freedom helicopter, an inverted pendulum, and an air levitation system. To emulate the different systems presented in this work, a class was implemented in the GUI that manages the animation and time-series graph of each system. This approach makes the GUI scalable, allowing more systems to be added without significantly modifying the rest of its programming, making the inclusion of additional systems entirely feasible.

Finally, the controllers evaluated in this work are presented as an example, but the platform is not limited to these types of controllers. The CUT can be modified, and the proposed platform can be used for its validation.

REFERENCES

[1] F. Mihalič, M. Truntič, and A. Hren, "Hardware-in-the-Loop Simulations: A Historical Overview of engineering challenges," Electronics, vol. 11, no. 15, p. 2462, Aug. 2022, doi: 10.3390/electronics11152462

[2] J. Montoya et al., "Advanced laboratory testing methods using Real-Time Simulation and Hardware-in-the-Loop techniques: A survey of Smart Grid International Research Facility Network activities," Energies, vol. 13, no. 12, p. 3267, Jun. 2020, doi: 10.3390/en13123267

[3] J. Martínez-Nolasco, V. Sámano-Ortega, J. Botello-Álvarez, J. Padilla-Medina, C. Martínez-Nolasco, and M. Bravo-Sánchez, "Development of a Hardware-in-the-Loop platform for the validation of a Small-Scale Wind System Control Strategy," Energies, vol. 16, no. 23, p. 7813, Nov. 2023, doi: 10.3390/en16237813

[4] L. Estrada, N. Vázquez, J. Vaquero, Á. De Castro, and J. Arau, "Real-Time hardware in the loop simulation Methodology for power converters using LabVIEW FPGA," Energies, vol. 13, no. 2, p. 373, Jan. 2020, doi: 10.3390/en13020373

[5] L. F. Quesada, J. D. Rojas, O. Arrieta, and R. Vilanova, "Open-source low-cost Hardware-in-the-loop simulation platform for testing control strategies for artificial pancreas research," IFAC-PapersOnLine, vol. 52, no. 1, pp. 275–280, Jan. 2019, doi: 10.1016/j.ifacol.2019.06.074

[6] R. F. Bastos, F. B. Silva, C. R. Aguiar, G. Fuzato, and R. Q. Machado, "Low-cost hardware-in-the-loop for real-time simulation of electric machines and electric drive," IET Electric Power Applications, vol. 14, no. 9, pp. 1679–1685, May 2020, doi: 10.1049/iet-epa.2019.0951

[7] A. Sanchez, A. De Castro, M. S. Martínez-García, and J. Garrido, "LOCOFloat: a Low-Cost Floating-Point format for FPGAs.: Application to HIL simulators," Electronics, vol. 9, no. 1, p. 81, Jan. 2020, doi: 10.3390/electronics9010081

[8] X. Dai, C. Ke, Q. Quan, and K.-Y. Cai, "RFlySim: Automatic test platform for UAV autopilot systems with FPGA-based hardware-in-the-loop simulations," Aerospace Science and Technology, vol. 114, p. 106727, Apr. 2021, doi: 10.1016/j.ast.2021.106727

[9] A. Rosa, "Integrated PBL and HIL practices for real-time simulations applied in technical and engineering teaching using embedded systems," Prz. Elektrotech, vol. 97, no. 1, pp. 46–52, Jan. 2021, doi: 10.15199/48.2021.01.08

[10] X. Tang and Y. Xi, "Application of hardware-in-loop in teaching power electronic course based on a low-cost platform," Computer Applications in Engineering Education, vol. 28, no. 4, pp. 965–978, Jun. 2020, doi: 10.1002/cae.22274

[11] W. Jiang, L. Sun, Y. Chen, H. Ma, and S. Hashimoto, "A Hardware-in-the-Loop-on-Chip development system for teaching and development of dynamic systems," Electronics, vol. 10, no. 7, p. 801, Mar. 2021, doi: 10.3390/electronics10070801

[12] Y. Martinez-Armero, S. Lopez-Blandon, and E. Giraldo, "Low-cost Arduino-based Hardware-In-the-Loop Platform for Simulation and Control of Dynamic Systems," IAENG International Journal of Computer Science, vol. 50, no. 4, pp. 1312–1318, Dec. 2023.

[13] Y. P. C. González, S. Z. J. Mora, and R. G. M. Morillo, "Tendencias y desafíos políticos y socio culturales de la educación superior contemporánea en Latinoamérica.," Revista Boletín Redipe, vol. 11, no. 1, pp. 71–91, Jan. 2022, doi: 10.36260/rbr.v11i1.1628

[14] L. M. S. Santana, M. Maximo, and L. C. S. Goes, "Physical modeling and parameters identification of the MG995 servomotor," in 26th International Congress of Mechanical Engineering, Florianópolis, Santa Catarina, Brazil, 2021.

[15] A. Taifour Ali, A. M. Ahmed, H. A. Almahdi, O. A. Taha, and A. N. A, "Design and implementation of ball and beam system using PID Controller," Automatic Control and Information Sciences, vol. 3, no. 1, pp. 1–4, Aug. 2017, doi: 10.12691/acis-3-1-1

[16] B. Meenakshipriya and K. Kalpana, "Modelling and Control of Ball and Beam System using Coefficient Diagram Method (CDM) based PID controller," IFAC Proceedings Volumes, vol. 47, no. 1, pp. 620–626, Jan. 2014, doi: 10.3182/20140313-3-in-3024.00079

[17] S. Anand and R. Prasad, "Modeling and control of Ball and Beam system," International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE), vol. 4, no. 9, pp. 2394–6849, Sep. 2017.

[18] K. Sehgal and N. Harsh, "Modelling and control of dynamical ball and beam system using SA tuned PIDA and PIAD controllers," in 2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 2021, pp. 1–6.

[19] J. C. Castro-Mendoza, C. J. Cabrera-Velázquez, E. Ríos-Montiel, D. Pérez-Silva, and R. Villafuerte-Segura, "Sistema Bola-Viga: Construcción y aplicación de técnicas de control," PÄDI Boletín Científico De Ciencias Básicas E Ingenierías Del ICBI, vol. 10, no. 6, pp. 107–116, Nov. 2022, doi: 10.29057/icbi.v10iespecial6.9217

[20] Sreenivasappa, B., & Udaykumar, R. (2010). Analysis and implementation of discrete time PID controllers using FPGA. International journal of electrical and computer engineering, 2(1), 71-82.

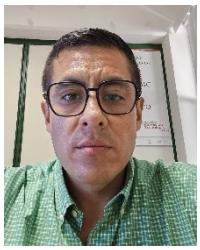
[21] Kocur, M., Kozak, S., & Dvorscak, B. (2014, May). Design and implementation of FPGA-digital based PID controller. In Proceedings of the 2014 15th International Carpathian Control Conference (ICCC) (pp. 233-236). IEEE.

[22] VictorSamano. Didactic-HIL, GitHub Repository. 2025. Available online: https://github.com/VictorSamano/Didactic-HIL (accessed on 21 January 2025).

Shadai Ararita Ojeda Mancera is a Mechatronics Engineer graduated from the Technological Institute of Celaya, where she received her B.Eng. degree in 2025. Her research interests include PLC programming, design and development of HMIs, vision-based projects, and creating initiatives that enhance education.

Jesús Gerardo Carranco Martínez is a mechatronics engineer graduated from the technological institute of Celaya, where he received his B.Eng. degree in 2025. His research interests include intelligent control, embedded programming, power electronics and dynamic systems emulators.

Víctor Manuel Sámano Ortega received his B.S. degree in Mechatronics Engineering from the Technological Institute of Celaya, Guanajuato, Mexico, in 2017; M.S. degree in Mechanical Engineering from the Technological Institute of Celaya, Guanajuato, Mexico, in 2019; and Ph.D degree in Engineering Science from the Technological Institute of Celaya, Guanajuato, Mexico, in 2024. He is currently a teacher in the Department of Mechatronics Engineering, Technological Institute of Celaya; his research interests include Intelligent Control, Direct Current Microgrids, and Energy Systems Emulators.



Juan José Martínez Nolasco received his B.S. degree in Electronics Engineering from the Technological Institute of Guzman City, Jalisco, Mexico, in 2007; his M.S. degree in Electronics Engineering from the Technological Institute of Celaya, Guanajuato, Mexico, in 2009; and a Ph.D. in Engineering Science from the Technological Institute of Celaya, Guanajuato, Mexico, in 2018. He is currently a teacher and researcher in the Department of Mechatronics Engineering, Technological Institute of Celaya; his research interests include intelligent control, direct current microgrids, applications of fuzzy logic control, and technologies related to agriculture 4.0.

Coral Martínez Nolasco received her degree in Electronic Engineering from the Technological Institute of Ciudad Guzmán in 2009. She completed her Master of Science in Electronic Engineering and her PhD in Engineering Sciences at the National Institute of Technology of Mexico in Celaya, obtaining her degrees in 2011 and 2023, respectively. Since 2011, she has been a professor at the National Institute of Technology of Mexico in Celaya. Her research interests include biotechnology, digital image processing, instrumentation, and control applications, as well as technologies related to agriculture 4.0. During her doctoral research, she implemented an aeroponic system that allows the development of lettuce and jalapeño pepper crops. She is currently the Coordinator of the Master of Science in Mechatronics Engineering program. As of January 2024, she receives recognition from the SNII as a Level 1 Researcher.

Mauro Santoyo Mora received the B.Eng. degree in Mechatronics Engineering and the M.Sc. in Electronics Engineering from the Celaya Institute of Technology (today Tecnológico Nacional de México en Celaya) at Celaya, Guanajuato, Mexico, in the years 2013 and 2016, respectively. Recently, in 2023 he received his PhD degree in Eng.Sc. at Tecnológico Nacional de México en Celaya. Since 2013 he has participation in the Mechatronics Engineering Department at Tecnológico Nacional de México en Celaya, where he develops different research projects at both undergraduate and postgraduate levels. His interests in research are related to implementing virtual environments for areas such as health, education, and industry, and the application of computerized vision systems.