

## Article

# Deployment of Model-Based-Design-Adaptive Controllers for Monitoring and Control Mechatronic Devices

Ramon Barber <sup>1,\*</sup>, David R. Rosa <sup>2</sup>, Antonio Flores-Caballero <sup>1</sup> and Santiago Garrido <sup>1</sup>

<sup>1</sup> Department of Systems and Automation, Universidad Carlos III de Madrid, 28911 Leganés, Spain; afcaball@ing.uc3m.es (A.F.-C.); sgarrido@ing.uc3m.es (S.G.)

<sup>2</sup> Department of Electrical, Electronic, Automatic and Communications Engineering, Universidad de Castilla-La Mancha, 45071 Toled, Spain; david.rrosa@uclm.es

\* Correspondence: rbarber@ing.uc3m.es; Tel.: +34-916248792

**Featured Application:** Mechatronics control and control engineering.

**Abstract:** The modeling and control of complex, non-linear, and time-changing mechatronic systems requires complex software development. They are carried out in the prototyping phase with engineering software development tools, generating models, and control algorithms that are not always easily exportable to control hardware. The following work offers an alternative that considers Model-Based Design using graphics-based languages, which facilitates programming tasks for modeling and controlling mechatronic devices and their transition from prototype to control hardware. Model-Based Design with high abstraction programming level capabilities provides the user with a fast coding and testing environment, suitable for laboratory prototyping and subsequent transfer to commercial embedded controllers. The proposed solution combines control hardware based on an STM32H7 microcontroller and a software development environment using graphics-based languages developed for MATLAB. The result is a solution that integrates control hardware and software in a hardware-in-the-loop paradigm. This solution provides robust and energy-saving controllers and demonstrates that an advanced control algorithm can be set up in a critical safety-compliant low-cost embedded controller via a custom model-based design. Algorithms and tools are transferred to the controller without losing the advantages gained in the prototyping phase. Finally, experimental results implementing an adaptive controller in a DC motor and in a pneumatic system are shown to validate the system.

**Keywords:** model-based design; adaptive controller; embedded controller; mechatronics control



**Citation:** Barber, R.; Rosa, D.R.; Flores-Caballero, A.; Garrido, S. Deployment of Model-Based-Design-Adaptive Controllers for Monitoring and Control Mechatronic Devices. *Appl. Sci.* **2023**, *13*, 12432. <https://doi.org/10.3390/app132212432>

Academic Editors: Miaolei Zhou and Rui Xu

Received: 18 October 2023

Revised: 10 November 2023

Accepted: 15 November 2023

Published: 17 November 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

### 1.1. Background and Literature Review

The need to control mechatronic devices integrating hardware and software, both at an industrial level and in research prototypes, requires new solutions that simplify the design and testing tasks of the controller and its transfer to the control hardware. In addition, it is crucial to provide user-friendly tools to facilitate engineers' training in these new techniques. Increasing industry demand for STEAM (science, technology, engineering, arts, and mathematics) competencies require innovative teaching and development approaches in order to minimize the fragmented nature of higher education and provide a way to conciliate the different fields of knowledge, such as mathematics, informatics, or mechanics [1,2]. Due to these demands, an affordable way to reach successful results in a constrained amount of time is to use high-abstraction-level programming tools, such as the ones provided via model-based design (MBD) methodology. National Instruments LabVIEW<sup>®</sup> and MATLAB/Simulink<sup>®</sup> are well-known MBD-based solutions. Other less-known solutions are Scicos and Scilab MBD-based systems [3]. In this work, an MDB system that develops

MATLAB/Simulink blocks optimized for hardware in the loop concept is proposed. This system uses a workbench design to simplify the control of mechatronics devices.

The MBD paradigm provides control algorithm developers and researchers with a rich development environment, where useful computer-assisted tools are available. Thanks to this support, they can develop and perform as many iterations on their control algorithm logic as they need, requiring little time per iteration. The high-level programming capability allows for the creation of extensive control programs and their rapid evaluation.

By using these programming methodologies, control algorithm developers and researchers have access to more advanced control techniques, such as adaptive PID controllers. To satisfy increased robustness requirements, non-classic PID controllers are needed. The loss in robustness for PID controllers is due to the constant gain parameters used in their implementation, which are not suitable for aging systems. The aging of the controlled plant modifies the system's behavior over time. If a controller does not take into account this fact, the effectiveness and stability of the controller response could be compromised.

The traditional development of mechatronic control systems usually uses hardware-in-the-loop (HIL)-enabled systems, especially those from companies such as dSpace and Typhoon [4,5]. Consequently, hardware and software development from a traditional HIL perspective follows a very restrictive but well-established working methodology [6]. This methodology is highly time and resources consuming when developing a system that goes beyond the laboratory technology readiness level (TRL) and qualifies as a mature TRL of level 5–6.

This article presents a software development environment based on custom MBD support, which facilitates the controller prototyping process for both industry and research. It allows for the development of high-end but low-cost embedded controllers, applied over mechatronic devices. In addition, the proposed method enables STEAM requirements; mathematics related to the control algorithm design are being correlated with the electronics and informatics knowledge required for the real implementation and testing of the controller.

## 1.2. Motivation and Contribution

All relevant aspects of how the implemented graphical model is carried out remain hidden when using one of these commercial HIL devices. The transition from prototyping to use in a real-world environment is an extensive, manual, and tedious process that often lacks the precision or continuous time solver solution capabilities used in the prototyping system.

Furthermore, the development and testing process needs to be reconsidered from a software point of view, as it is not feasible to invest many years in developing software for embedded platforms given the current life expectancy. Mandatory security measures related to integrity, stability, error-free coding, and code quality assurance (e.g., MISRA C coding guidelines) must be considered. Nowadays, there is an increase in agile methods for software development, where testing is not complicated and occurs naturally after partial implementation of the code. The higher the level of abstraction in programming, the easier it is to test and find algorithm errors. Consequently, low-level hardware drivers are provided as graphical blocks, coded as a one-time implementation, and focusing on integrity and security reliability for this low-level and hardware-related part of the code.

In trying to solve these problems and deficiencies, this work proposed a HIL workbench that is based on a COTS platform. It is designed to be suitable for prototype development and the deployment of controllers within a software development environment that includes custom MBD support. Additionally, the controller prototyping process is allowed for both industry and research purposes.

Moreover, it can be employed for implementing control solutions and teaching control concepts in hands-on classes. This approach allows trainees to attain an appropriate level of complexity within a limited number of practical sessions, such as modeling and controlling actuators. Students themselves experiment with the results of their work on the HIL

testbed and find that some algorithms, while useful in theory, are difficult to use on the HIL testbed due to computational limitations; or, conversely, they are only feasible with a highly abstract language.

One of the main advantages of the proposed system is that when using a COTS-enabled HIL benchmark, the developer only needs to focus on solving the real problem and can abstract from the low-level hardware. Furthermore, another important advantage of the proposed system is that it generates an optimized C source code and avoids the complexity of having to program the algorithms and associated continuous-time solvers, such as Runge–Kutta. It should be pointed out that implementing these types of solvers without CAD-assisted tools take considerable time.

In this sense, the proposed solution combines control hardware based on an STM32H7 microcontroller and a software development environment using graphics-based languages developed for MATLAB. This solution provides robust and energy-saving controllers and demonstrates that an advanced control algorithm can be set up in a critical safety-compliant low-cost embedded controller via a custom model-based design.

Algorithms and tools are transferred to the controller without recoding tasks, avoiding losing the advantages gained in the prototyping phase.

### 1.3. Paper Organization

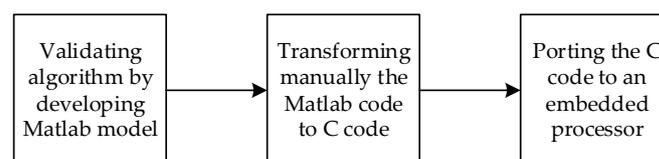
The article is organized as follows: Section 2 describes the concept of graphics-based programming and the implementation carried out under this paradigm. Section 3 describes the hardware/software of the developed workbench. Section 4 proposes the implementation of an adaptive controller as an application in mechatronic devices, including identification, adaptive PID design and stability study of the controller. Section 5 describes the implementation of the controller using the proposed system, including blocks related to the input/outputs of the plant, plant identification, and the controller. Section 6 presents the experimental results over two laboratory prototypes: a DC motor and a pneumatic arm. Finally, Section 7 summarizes the conclusions that arise from the work presented in this article.

## 2. Graphics-Based Programming

A graphics programming language offers a high-level abstraction, enabling those familiar with the MBD environment to create programs for supported hardware without the need for expertise in textual-based languages such as C.

This programming paradigm commonly has many disadvantages, such as the ones clearly reported in Valera's dissertation about current-day commercially and non-commercially available MBD hardware support [7]. These disadvantages are based on limited embedded hardware I/O features accessible from MBD software. They are also based on the absence of multitasking support and on the poor or non-implemented target-oriented optimizations at the source code generation stage. This often limits the use of digital signal processing (DSP) features within the embedded controller. Unfortunately, due to their disadvantages, high cost, and the few real ready MBD-supported embedded controllers, some authors think that the final implementation is out of their scope due to the amount of work, time, and knowledge needed for text-based programming [8].

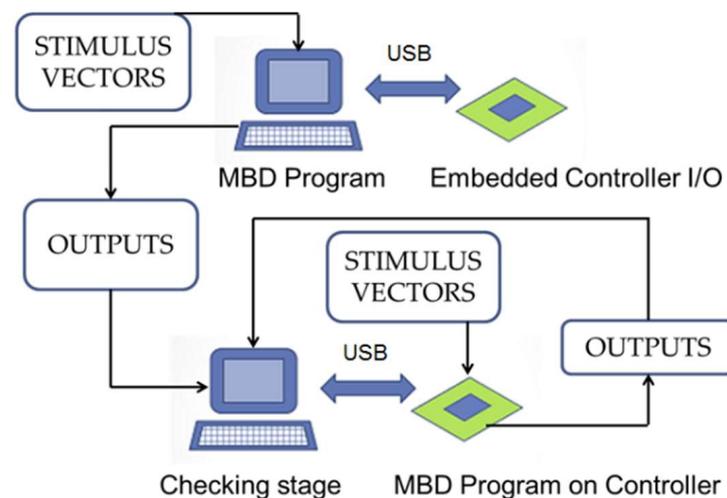
Due to these cited obstacles, the most common way of developing an MBD controller is shown in Figure 1.



**Figure 1.** Typical MBD work scheme at laboratories.

Commercial MBD-supported hardware, suitable only for laboratory purposes, is used due to its cost, size, and power consumption. Once the graphics program developed using the MBD environment meets the required standards, it is manually converted to the target hardware for embedded systems. Detailed examples and descriptions of this methodology can be seen in [9]. This last methodology is only advantageous during the laboratory prototyping stage, as the final implementation relies on handwritten programming. The embedded controller behavior cannot be compared to the original MBD-based program as they are not the same, and controllers with safety restrictions must comply with this condition according to IEC 61508 [10].

This article presents custom MBD hardware support for MATLAB/Simulink®. This support enhances I/O, multitasking, and optimizes DSP features for embedded controllers, eliminating the need for manual programming stages in the development process, as successfully demonstrated in [9]. This custom MBD support enables the embedded USB controller port as the data exchange interface for checking and comparing the controller behavior with the original MBD program in compliance with critical safety directives. Figure 2 illustrates the checking features that needed to be checked for critical safety controllers using an MBD environment [11]. At first, a set of input stimulus vectors are applied to the MBD-based program. The embedded controller provides the I/O features.



**Figure 2.** Safety integrity level software verification using a MBD environment.

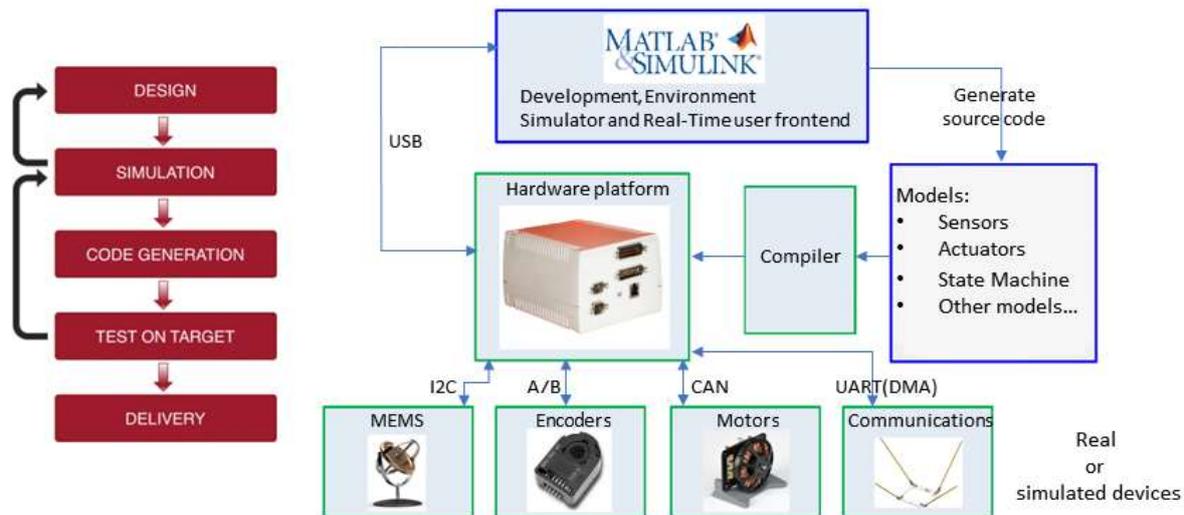
The interface between MATLAB/Simulink® and the embedded controller is a custom deterministic native high-speed USB interface [9]. The output obtained from the MBD program behavior is stored. Finally, these data are compared with the output of the MBD program running entirely on the embedded controller. All discrepancies need to be under a specific limit according to each level of the SIL directives [12,13]. SIL directives are historically applied and evaluated in different ways depending on the application scenario [14]. Concerning the control software implementation discussed in this article, safety features are incorporated into the actuator to control real-time task. A supervisor task with higher priority is responsible for sensor data validation. It conducts a system performance evaluation to detect any signs of the system going out of control, which could be caused by I/O issues, such as sensor wiring or actuation problems.

In this paper, an adaptive controller using the custom MBD hardware support is developed and discussed for a 600-watt DC motor and a didactic intended pneumatic system. The main goal is to provide enough feedback about the viability of the proposed programming methodology, replacing the last two sequence blocks in Figure 1 with automatic procedures.

### 3. Hardware/Software Description of the Developed Workbench

The selected COTS are based on the STM32H7 family from ST microelectronics manufacturer (Geneva, Switzerland) [15]. From a hardware point of view, the HIL hardware is the so-called mockup. It has a large number of I/O peripherals, including analog I/O, PWM I/O, digital I/O, quadrature encoder inputs, and several communication protocols (RS-232, RS-422/485, Ethernet, SPI, I2C, USB, and CAN2.0B). Accordingly, all these interfaces are protected against reverse polarity, overcurrent, and overvoltage.

The in-house software for MATLAB/Simulink integrates the target setup, input/output peripherals, MISRA C code generation, the C compiler, debugger environment, and real-time monitoring functions via specially developed dynamic link libraries. This enables an agile working methodology, as described in Figure 3.

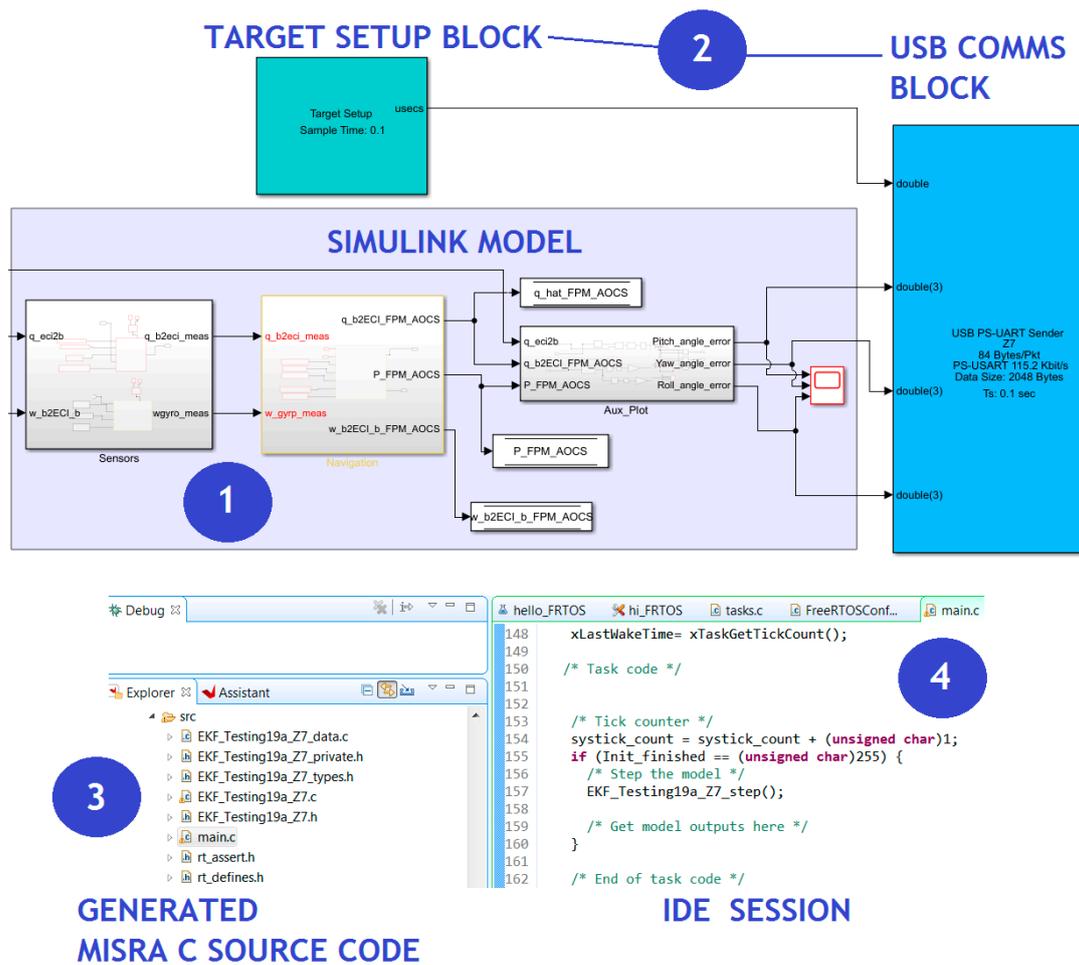


**Figure 3.** Testbench software and hardware environment.

The working methodology uses MATLAB/Simulink as the development and simulation environment and the STM32H7-based system as the HIL-enabled target. Since the hardware board operates autonomously, the scheme in Figure 1 includes additional integration stages, such as compilation, deployment, execution, and data retrieval. If the hardware support block is set for this HIL mockup, it enables us to obtain experimental results and compare them with the simulated results.

In addition, the implemented Simulink models do not wait for the response from the computer to start executing, unless the user has precisely set this function via graphical programming (or the text-based MATLAB code within a MATLAB function block). Intentionally, for further development, deployment, and technology transfer purposes, the generated MISRA C qualified code and safety-programmed (SIL3-/SIL 4-oriented) low-level hardware drivers remain accessible to the user. On the other hand, this HIL mockup differs from [16] and most of the HILs systems at [17] because the purpose is to use the presented COTS HIL test bench hardware for verification, simulation, and deployment purposes, as discussed in [9].

Figure 2 illustrates the use of the HIL mockup software. In accordance, it shows as the first required item a Simulink model. In this example, an attitude and trajectory control algorithm receives inputs from the space simulator. This model can be easily integrated into the HIL mockup by adding the target setup block (as shown in Figure 4) and a communication block, such as a USB channel, to retrieve and visualize information about the Simulink model on the hardware target.



**Figure 4.** Test bench software procedure description: (1) First the model is developed in Simulink. (2) Blocks required in order to manage hardware and I/O features are added. (3) The code is generated automatically. (4) The generated source code remains accessible to be modified or verified.

However, the automatic compilation and execution processes can be disabled since full automation may introduce excessive abstraction that is unsuitable for certain training stages. The source code generated from the Simulink model is shown as item 3 in Figure 4.

The trainees use the integrated development environment (IDE) provided by STM32 (point 4 in Figure 4) to compile and load the binary program. Subsequently, the whole process is illustrated, from the development of algorithms/programs in a high-level abstraction language to the visualization and guided inspection of the resulting source code. Furthermore, it is interesting to illustrate the differences and relevance between non-normalized C programming and the source code that follows the usual MISRA C guidelines. As a result, developers and students can understand that implementing a Simulink model on target hardware relies on pre-existing, reconfigurable, low-level drivers for the target, and this process can be seamlessly integrated with traditional debugging using the IDE.

To demonstrate this last point, the block set includes an intentionally broken hardware driver block to provide practical knowledge on how to locate the point of failure using IDE and the debugger. This can be interesting as some developers and students are not familiar with debugging sessions. This experimental practice complements the training in two ways: on the one hand, it provides a basic knowledge about the usefulness of an IDE; and on the other hand, it shows that a Simulink block, despite its graphical appearance, can contain errors.

The proposed mockup provides the following advantages:

- High-level software capabilities: In alignment with the dual purpose of the HIL mockup, which includes prototyping and setup, the provided software functions cater to a wide range of needs, offering programming capabilities through graphical blocks. These blocks cover a wide range of functions, from basic I/O management to the creation of real-time tasks. Furthermore, advanced low-level functions, such as direct memory access (DMA) and interrupt functions, are seamlessly integrated into the graphical block functionalities. Importantly, users are not required to manage these hardware aspects.
- The available functions are directly related to the name of the category. Under the tag “analog I/O”, you can find the blocks for analog I/O. In the communication category, you can find all available digital channels, such as inter-integrated circuits (I2C), the serial peripheral interface (SPI), and so on.
- The device configuration contains the target setup blocks responsible for generating code for the HIL mockup hardware. Under the tag “digital I/O”, you can find external interrupts, general digital lines, pulse width modulation (PWM) output, and quadrature encoder inputs. In addition, under the miscellaneous heading, you can find the delay and a block for retrieving the elapsed time (in nanoseconds). This block is helpful for measuring the execution times of any part of the model. Under the label “Multitasking”, you can find the option to create and use additional real-time-based tasks and to use idle time.
- The execution of any real-time task can be controlled with an additional block, pausing or resuming it as necessary. The computer page label contains blocks for USB configuration and use of the communication channel to exchange data with the HIL mockup when connected to the computer.

#### 4. Control Application of Mechatronic Devices: Adaptive Controller

Traditional controllers are developed for dynamic systems with invariant behavior. These systems usually maintain stable internal parameters around their calibrated operating point, which represents an ideal scenario that does not always hold true. In the real world, there are perturbations, such as vibrations or actions of near actuators, that induce changes in the plant model parameters.

Actuator systems often exhibit varying behaviors depending on the working load, such as torque, which can naturally fluctuate. For instance, when a straight metal bar is attached to the end of the actuation system, traditional controllers cannot be used since the torque changes with the bar position. Therefore, the entire system is severely affected by the controller performance.

In this sense, adaptive controllers [18], also known as self-tuning regulators (STR), arise as a solution for the limitations of traditional controllers [8]. The goal of these controllers is to recalculate the regulator parameters at each iteration of the control algorithm to provide accurate system behavior. Figure 5 shows the general scheme of these types of controllers.

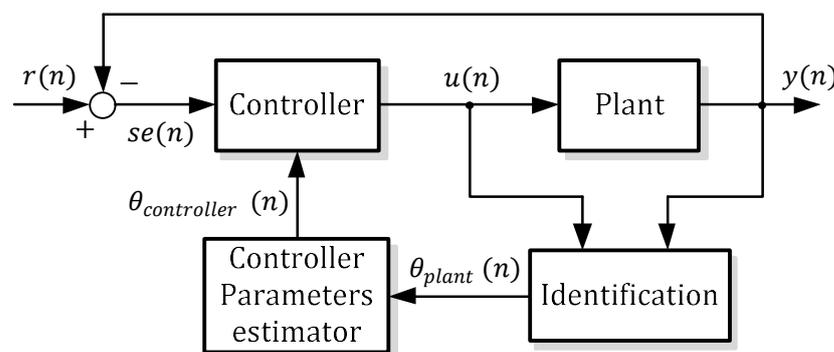


Figure 5. Scheme of an adaptive controller.

This scheme presents a typical control in a closed loop and adds a parallel blockchain for the online plant model identification and parameters estimation. These estimated parameters for the controller are calculated at each iteration based on the online-identified plant model. Therefore, the entire system’s behavior is dynamically adjusted.

4.1. System Identification

To implement an adaptive controller, it is necessary to have access to the inputs and outputs, labeled in Figure 3 as  $u(n)$  and  $y(n)$ . With these acquired data, the online identification block obtains an estimated mathematical model of the real plant, denoted as  $\theta_{plant}(n)$  in Figure 3. The estimated model of the plant can be obtained using several algorithms; one of the most commonly employed is the recursive minimum square method [19]. The controller parameters estimator uses the identified plant model to adjust the controller gains and obtain the desired behavior according to the estimated plant. Therefore, it performs a dynamically adjusted controlled system, providing a controller with better robustness against aging, disturbances, and minor mechanical problems.

For the algorithm performance and paper/article purpose, a first-order model, with the following transfer function, is chosen:

$$G_p(s) = \frac{A}{s + B}, \tag{1}$$

where  $A$  and  $B$  are the parameters to be identified.

4.2. Adaptive Controller

To test the adaptability of the technique, the I-PD control scheme is implemented. This control system is a modification of the PID control scheme [20]. The I-PD control scheme is shown in Figure 6, where  $K_p$ ,  $K_i$ , and  $K_d$  are the proportional, integral, and derivative gains of the regulator.  $R(s)$  is the reference signal, and  $Y(s)$  is the output signal of the system.

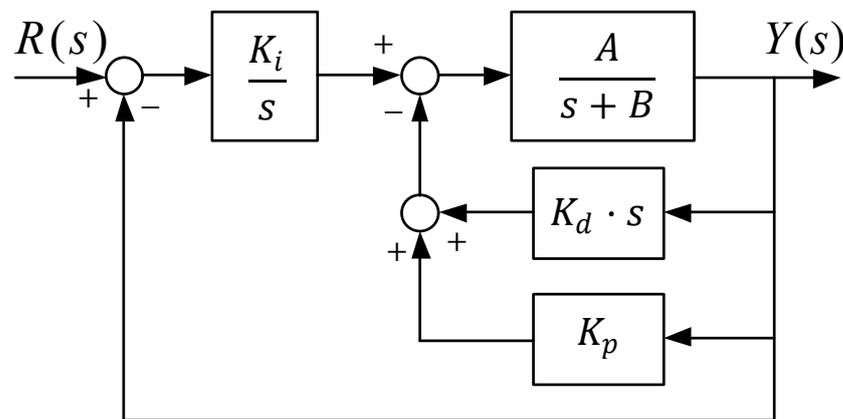


Figure 6. Scheme of the I-PD controller.

In this type of PID controller, the integral and derivative actions operate on the output signal of the system, rather than on the error signal as it occurs with classical PID controllers. The basic idea of I-PD control is to avoid large control signals which causes saturation phenomenon. By bringing the proportional and derivative control actions to the feedback path, values for  $K_p$  and  $K_d$  can be chosen with a greater range than those chosen when using a PID control scheme. This facilitates the convergence of the adaptive controller.

On the other hand, one disadvantage of this type of controller is their slower rise time compared to a classic PID controller when using the same gains values. In addition, these controllers become unstable more easily, so this should be taken into account when tuning them.

The equivalent transfer function of the hole system can be written as follows:

$$G_{sys}(s) = \frac{\frac{A \cdot K_i}{1 + A \cdot K_d}}{s^2 + \frac{B + A \cdot K_p}{1 + A \cdot K_d} s + \frac{A \cdot K_i}{1 + A \cdot K_d}} \tag{2}$$

It should be pointed out that the same transfer function is obtained for both modified and classical PID controllers, but without the zeros of the classical PID. Equation (2) can be compared with a second-order system, following the expression presented in [20]:

$$G_{2o}(s) = \frac{\omega_n^2}{s^2 + 2 \cdot \omega_n \cdot \zeta \cdot s + \omega_n^2} \tag{3}$$

where  $\omega_n$  is the natural frequency of the system, and  $\zeta$  the damping ratio. Notice that parameters are only used to calculate the coefficients; they do not have physical meaning.

By rearranging Equations (2) and (3) and adjusting the derivative gain based on stability criteria  $K_d$ , the I-PD gains can be calculated as follows:

$$K_i = \frac{\omega_n^2 \cdot (1 + A \cdot K_d)}{A} \tag{4}$$

$$K_p = \frac{2 \cdot \omega_n \cdot \zeta \cdot (1 + A \cdot K_d) - B}{A} \tag{5}$$

These adaptation laws are used to calculate the regulator gains according to the changing parameters of the plant.

Since three equations are obtained with two unknown parameters, the value for  $K_d$  was established empirically depending on the plant using Ziger Nichols Techniques [21], taking into account the stability rules to obtain a control action with a fast response and reducing the noise. However,  $K_d$  will be bounded with the stability condition of the system.

Thus, from the values obtained from the plant via online identification, it is possible to calculate the regulator gains according to the natural frequency  $\omega_n$  and the damping ratio  $\zeta$ , a second-order system with the desired specifications.

### 4.3. Study of the Stability

The stability of the system is directly related to the position of the roots of the characteristic equation of the transfer function of the system. In this sense, Routh–Hurwitz is a useful method to calculate the stability of the system [22].

The characteristic equation of the feedback system is obtained as follows:

$$s^2 + \frac{B + A \cdot K_p}{1 + A \cdot K_d} s + \frac{A \cdot K_i}{1 + A \cdot K_d} \tag{6}$$

The method indicates that there are no roots with positive real parts if all the terms in the first column of the Routh–Hurwitz table (see Table 1) are positive.

**Table 1.** Routh–Hürwitz criterion table to study the stability.

$s^2$		1	$\frac{A \cdot K_i}{1 + A \cdot K_d}$
$s^1$		$\frac{B + A \cdot K_p}{1 + A \cdot K_d}$	0
$s^0$		$\frac{A \cdot K_i}{1 + A \cdot K_d}$	

Applying the method, we can obtain the following expressions:

$$K_p > -\frac{B}{A} \tag{7}$$

$$K_i > 0, \quad (8)$$

$$K_d > -\frac{1}{A}. \quad (9)$$

These equations provide a method by which to determine, based on the values of the plant, the possible values of the gains that safeguard system stability in a closed loop.

Thanks to this, the stability of the system is ensured at all time, and the controller's gains are calculated by the laws of adaptation.

## 5. Implementation

The control system is developed using Simulink according to Figure 4. In order to program the controller, the user has to include in the Simulink model the blocks related to hardware resources (i.e., input/output peripherals such as analog to digital, digital to analog, serial ports) and configure it.

First, the controller model is executed in the computer, using the embedded controller as an I/O device, while the outputs are stored to be used later in the validation stage. Then, the model is directly implemented in the embedded controller according to MBD and SIL-related directives [14]. Thanks to this methodology, checking the output between the control program running on the computer, or on the embedded controller, is easier than with the traditional handwriting programming methodology. Moreover, detecting embedded implementation, code generation, and compilation errors during debugging is easier.

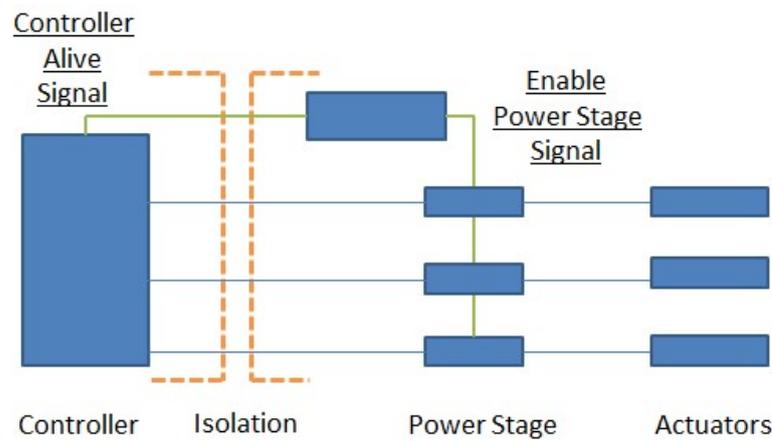
The described working methodology uses two MBD programs, the control model, and a data acquisition model. The data acquisition model runs only on the computer, and it is useful to acquire and show data from the USB interface of the embedded controller. In addition, this data acquisition program can be used to send data or orders to the embedded controller at execution time.

Safety depends on both software and hardware implementation [13]. The embedded controller can be damaged either by a mechanical accident or by electrical problems. This situation needs to be considered to ensure the entire system's safety and reliability. Typically, the controller is optocoupled from the power stage, but the power stage must detect if the controller is still working. Damaged controllers can still produce an uncontrolled output signal to the power stage, potentially leading to a catastrophic situation. Not all the safety-related aspects of the embedded software can address scenarios such as this. To prevent this potentially dangerous situation in a human-like scenario, as presented in this paper, the power stage is equipped with a controller to maintain signal detection circuitry. Figure 7 shows the entire controller with the power stage scheme. The signal controller is complex because it needs a fixed frequency for the analog output signal. In this way, if the embedded controller is damaged or a system failure takes place, it avoids output failure.

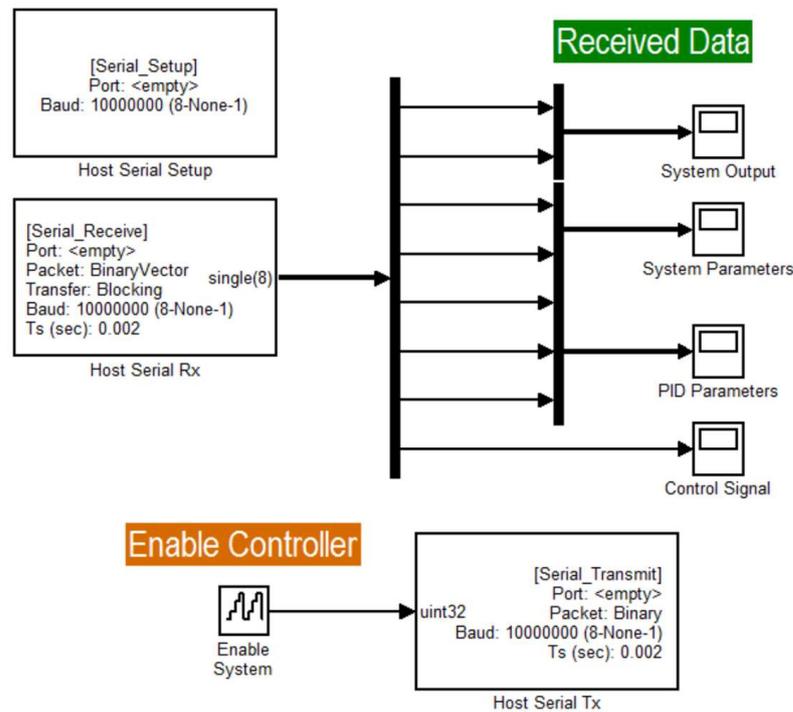
Figure 8 shows the Simulink scheme that runs on the PC, where all the variables required are displayed to the user through graphical scopes.

Figure 9 shows the Simulink model transferred to the microcontroller, an STM32F407 with a floating-point unit from ST Microelectronics® manufacturer. It consists of the algorithm that performs the actions of the adaptive controller. This model is divided into four parts: first-order system, identifier, calculation of PID gains, and adaptive PID controller. It also contains a data output block for sending controller data through the microcontroller's USB port.

The Simulink® model intended for the embedded controller must contain a target setup block, represented in Figure 6 at the top right corner. This block specifies the code generation process, the hardware architecture, and other related stuff, such as MISRA C-compliant features, to generate and compile a program suitable for the selected embedded controller.



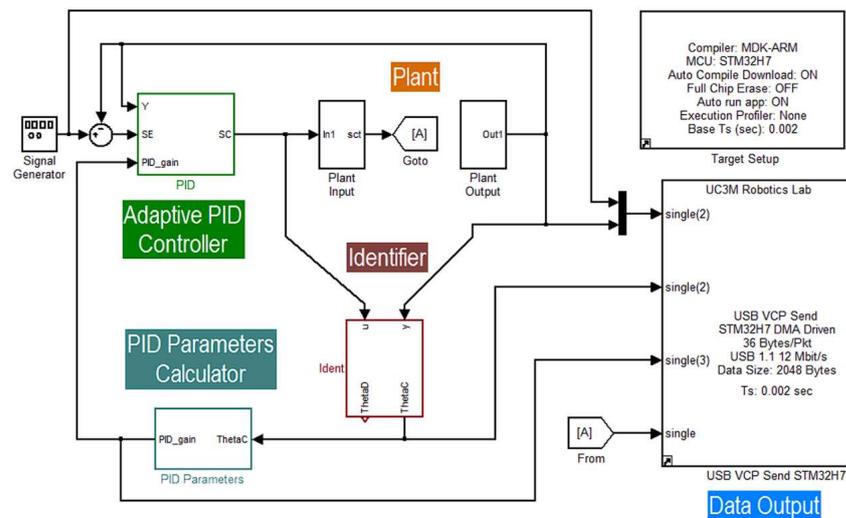
**Figure 7.** Security related hardware implemented. The blue lines represent I/O lines that connect the target hardware to external devices (sensors, actuators). The orange dashed line represents the galvanic isolation of the signals. The green line indicates a heartbeat signal to prevent the actuators from failing in case the controller hangs.



**Figure 8.** Simulink scheme to retrieve controller data.

The model works as the one described in Figure 5. Parameters are identified from the inputs and outputs of the system. These parameters are used to calculate the new regulator gains, and with those gains, the PID controller is modified to obtain a control action that meets the imposed specifications.

However, when working with real systems, an initial period is required in order to obtain a first estimation of the parameters. Therefore, a conventional fixed-parameter PID controller is used in these first stages. It makes the system stable and allows for the proper identification of the plant parameters. After this time, the adaptive control is launched, resetting the PID controller gains at each sampling time according to the obtained identification values.



**Figure 9.** Simulink simulation and target scheme.

### 5.1. Input/Outputs of the Plant

This part of the model interacts with the plant. It includes the I/O interfaces of the real system, allowing for interaction between the control system embedded in the hardware and the physical system.

### 5.2. Identifier

Online identification of the plant is necessary for implementing adaptive control, requiring estimation of the plant parameters at each sampling time. Prior knowledge of the plant behavior is needed, and in this case, it is modeled as a first-order transfer function.

The identification algorithm is the recursive least squares with a forgetting factor. The identification results correspond to the parameters A and B of the plant according to Equation (1).

The identifier requires some time in order to obtain a correct estimation of the parameters and converge to the identified values. This time varies depending on the plant that is identified.

### 5.3. PID Parameters Calculator

From the identified values of the plant, and using the adaptation laws described in Equations (4) and (5), the I-PD parameters are recalculated at each sampling time. The gains of the regulator are calculated to fit in a second-order system, where the damping ratio and natural frequency can be tuned. A conventional PID controller is previously used in the first stages because the system requires an initial time interval for the correct identification of the plant. These regulator parameters are obtained by off-line identification of the plant working without load to give an estimation that allows for the stability of the system. Meanwhile, the first stable values are obtained online by identifying the plant.

### 5.4. Adaptive I-PD Controller

The adaptive I-PD controller consists of a modified PID controller with variable gains. As a result, the block receives as input the value of  $K_p$ ,  $K_i$ , and  $K_d$  gains, obtained from the calculation of the gains block, and readjusts the PID in each sampling time.

## 6. Experimental Results

Several tests were carried out to validate the adaptive controller implemented using the proposed MBD tool. Two models were used: a speed-controlled DC motor and a pneumatic arm controlled in position.

The DC motor output signal provides revolutions in volts. The pneumatic arm consists of a double-effect cylinder and a linear resistor coupled to the arm. It provides, as an output signal, the arm position in volts.

In both systems, the fixed sampling time is determined based on the system characteristics, incorporating it for both identification and control, within all blocks of the control system. This approach ensures real-time control suitability.

### 6.1. DC Motor

The DC motor model used for the experiments, shown in Figure 10, uses a potentiometer to experimentally change the plant's gain.



**Figure 10.** DC motor for use in the experiments.

The potentiometer allows three gains,  $k_1$ ,  $k_2$ , and  $k_3$ , where

$$k_1 < k_2 < k_3 \quad (10)$$

A step signal with an amplitude of 1 V and 0.1 Hz is used as reference for the control study of the motor speed.

In Figure 11, the motor response is displayed while changes in the operating point were set off every 10 s by changing the gain of the motor. Static and dynamic responses remain constant despite the changes in motor gain. The controller adapts the control signal to the set changes in order to keep the values of the damping ratio and natural frequency. For the test, the output signal has a damping ratio of  $\zeta = 1.5$  and a natural frequency  $\omega_n = 10$  rad/s. The actuation time of the conventional controller is 1.5 s, and the simulation period is 30 s.

In addition, due to the motor noise, the gain of the plant increases; as a consequence, an increment of the output ripple is observed.

The gain values of the I-PD controller are shown in Figure 12. As shown in the experimental results, the I-PD controller parameters are fixed using the parameters of a conventional PID. This helps force convergence in the initial moments, facilitating the transition from the conventional to the adaptive controller.

At  $t = 1.5$  s, a sudden variation of the gains occurs because of the change from the starting conventional controller to the adaptive controller. In periods where there are changes in the dynamics of the plant, the  $k_d$ ,  $K_p$ , and  $K_i$  values of the adaptive controller tend to stabilize to those values that reach the appropriate control action. These values are adjusted automatically according to the new parameters identified by the motor. The  $K_d$  adaptive value has been empirically established by obtaining a fast response but avoiding the amplification of noise effects from the sensor signal.

Figure 13 shows the evolution of the parameters identified in the plant. Once the gain of the motor changes, the identified values tend to stabilize again to a new value.

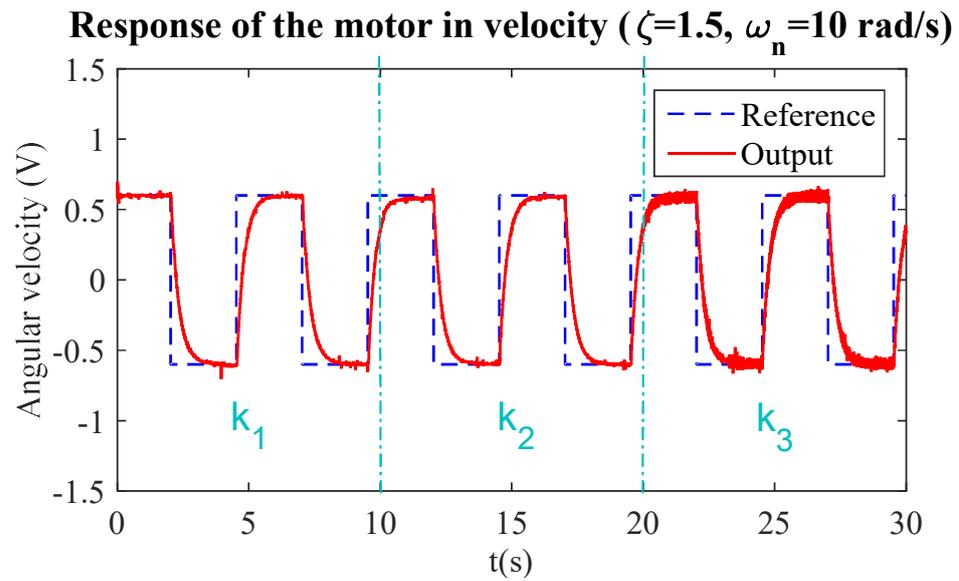


Figure 11. DC motor input and output signals with  $\omega_n = 10$  y,  $\zeta = 1.5$  y, and k variable.

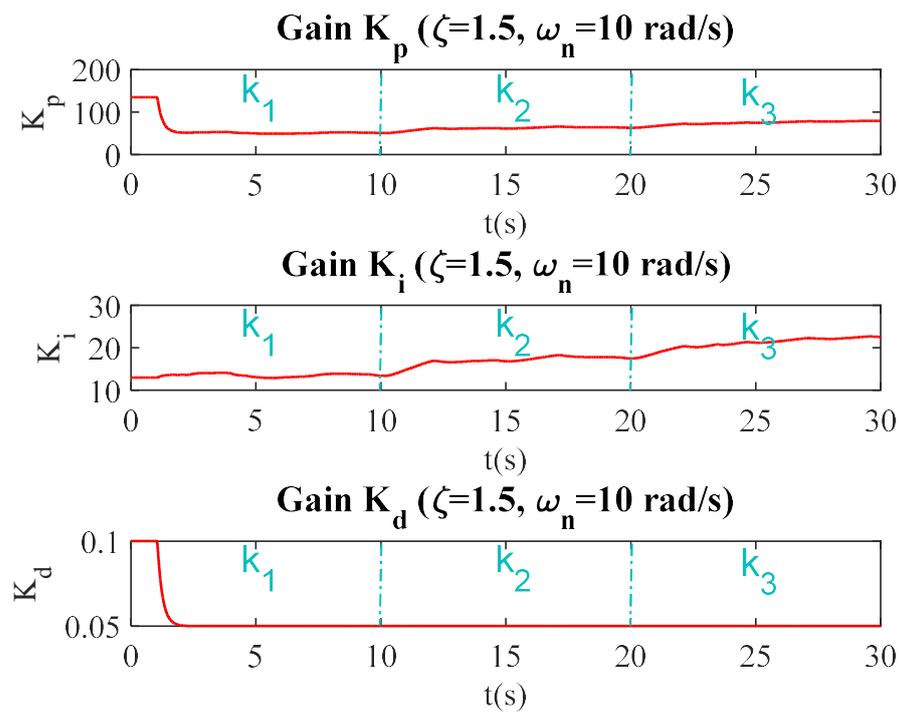


Figure 12. Evolution of the values of the controller gains of the DC motor with  $\omega_n = 10$  rad/s,  $\zeta = 1.5$ , and k variable.

In the instant when the change from the conventional controller to the adaptive controller occurs (approximately 4 s), a small oscillation appears in the output signal due to the adaptive adjustment of the regulator gains.

Figure 14 shows different responses achieved under different control requirements, such as varying the damping ratio and setting the gain of the DC motor. The controller changes the behavior of the plant by following the chosen natural frequency and damping ratio since it is the regulator premise.

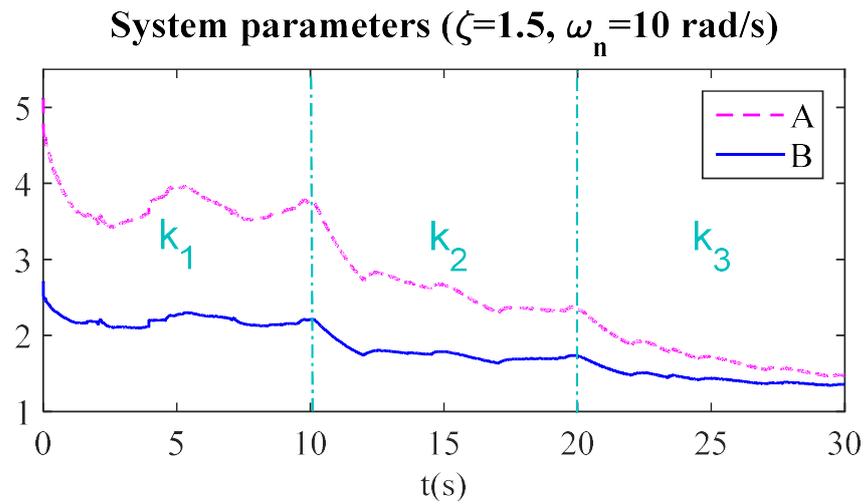


Figure 13. Values of the identified parameters of the DC motor with  $\omega_n = 10$  y,  $\zeta = 1.5$  y, and k variable.

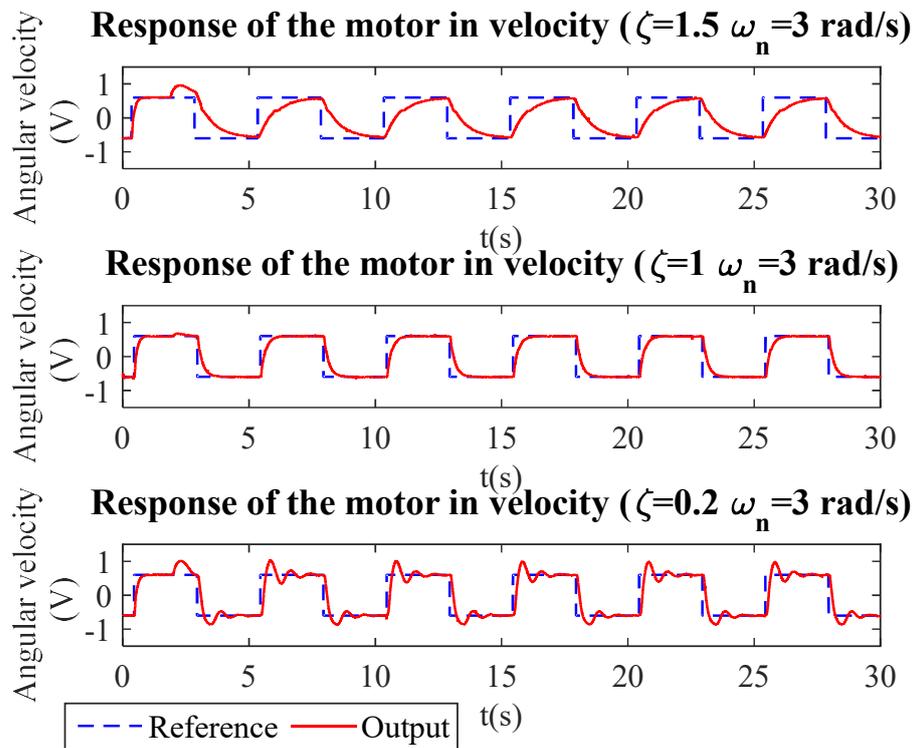


Figure 14. Responses of the DC motor speed for different  $\zeta$  with  $\omega_n = 3$  rad/s.

6.2. Pneumatic Arm

Figure 15 shows the experimental platform used in the case of the pneumatic arm. In this case, a study of the control of the position of the arm is accomplished. The chosen reference signal is a step with an amplitude of 1 V and a frequency of 0.05 Hz. On the other hand, a damping ratio  $\zeta = 1$  and a natural frequency  $\omega_n = 1$  rad/s are chosen as control specifications. The time in which the conventional controller acts is 25 s, and the simulation period is 60 s.

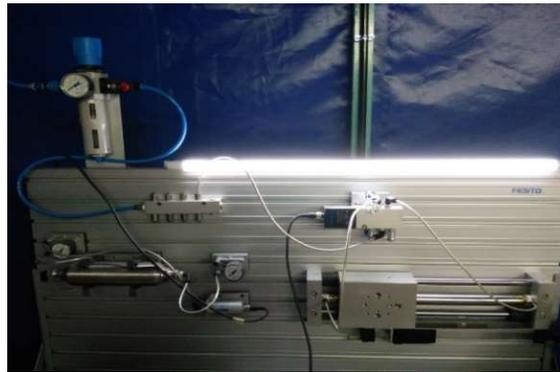


Figure 15. Pneumatic arm used in the experiments.

Figure 16 shows the values identified from the pneumatic arm. It is notable how the identified parameters stabilize at  $t = 12$  s.

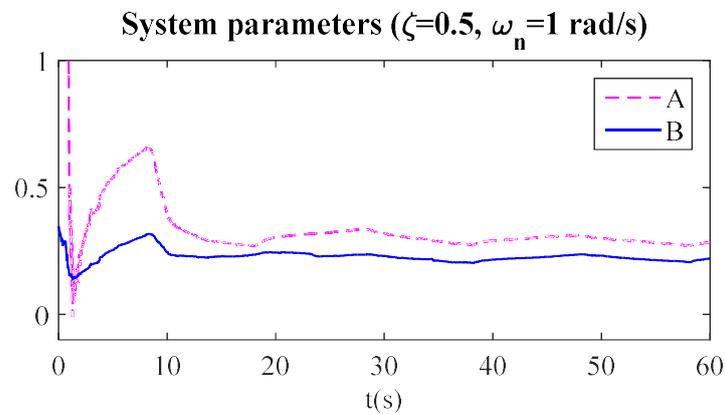


Figure 16. Values of identified parameters of the DC motor with  $\omega_n = 1$  rad/s and  $\zeta = 0.5$ .

Figure 17 shows the values for the gains of the controller. In the DC motor case, a sudden variation appears. At second 24, due to the change from the conventional controller to the adaptive one, the gain parameters  $K_p$  and  $K_i$  stabilize their values to obtain the imposed specifications.

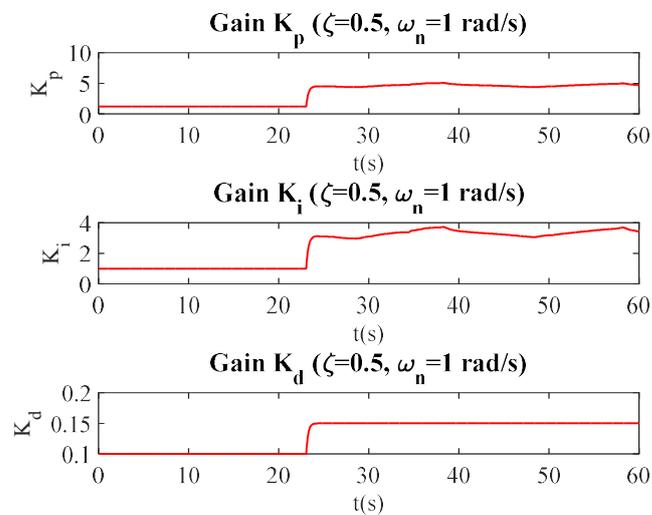
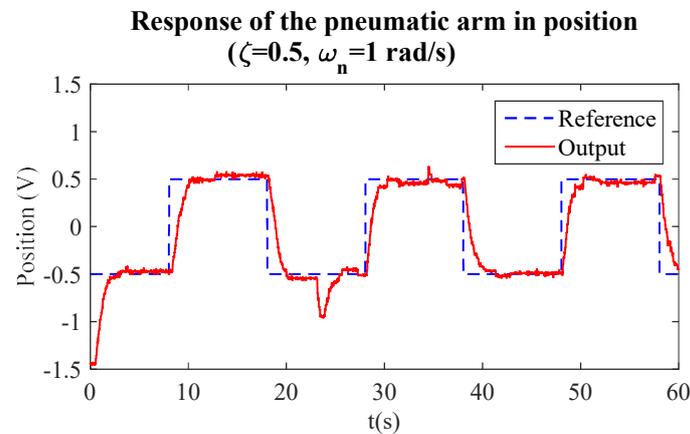


Figure 17. Evolution of the values of the controller gains pneumatic circuit with  $\omega_n = 1$  rad/s and  $\zeta = 0.5$ .

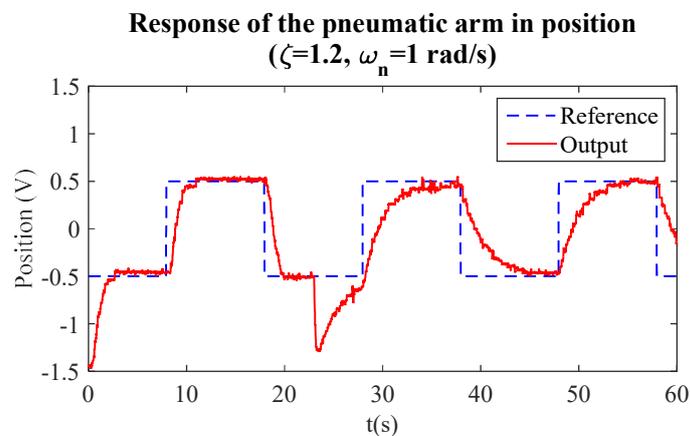
Finally, Figure 18 shows the system response to the step input. At second 24, when the conventional controller switches to the adaptive one, the output signal presents a peak because the adaptive controller is still tuning the parameters.



**Figure 18.** Input and output of the pneumatic circuit with  $\omega_n = 1$  rad/s and  $\zeta = 0.5$ .

Although the identifier needs time to estimate the parameters, the controller keeps the system under control, guaranteeing that the system reaches the control specifications.

Figure 19 illustrates the case of an overdamped system, where the adaptive controller requires more time in order to correctly perform the tuning of the parameters. This effect needs to be considered depending on the safety measures.



**Figure 19.** Input and output of the pneumatic circuit with  $\omega_n = 1$  rad/s and  $\zeta = 1.2$ .

The results from the experiments allow us to establish the following:

- The implemented identification system reaches the model convergence of the parameters even with sudden changes in the dynamics of the plant.
- It is possible to control the dynamics of the plant through the configuration of the damping and natural frequency. This allows for a much faster system with different responses, namely, overdamped, critically damped, and underdamped.
- Prior knowledge of the plant for the regulators (both conventional and adaptive) is necessary.
- For changes in the operating point of the plant, the laws of adaptation can recalculate the regulator gains. Therefore, the dynamic behavior of the system is always the same.
- Fast implementation is thanks to MBD hardware support and the easy configuration of the model in achieving the desired dynamic.
- The embedded algorithm can run the program independently from the computer in a low-cost controller.

- The real-time graphical interface on the computer allows for interaction with the embedded controller, allowing for both the monitoring and modification of system variables.
- The critical safety evaluation scheme for MBD for the developed controllers is correctly tested. The controller behaves as expected. It should be pointed out that each scenario (e.g., industrial or aeronautics systems) requires different approaches to safety measurements. Further research in this area is recommended due to discrepancies between the critical safety concepts.

## 7. Conclusions

This paper presents a mechatronic hardware and software controller designed for use in high-abstraction languages such as MATLAB and Simulink. This proposal is a solution that integrates control hardware and software in a hardware-in-the-loop paradigm using a software development environment based on custom MBD support, which facilitates the controller prototyping process for both industry and research. Since the integration functions have been developed, a wide range of I/O peripherals and solvers, such as continuous-time ones, are supported, enabling universal mechatronic control.

The main advantage of the proposed system is that the developer only needs to focus on solving the control problem and can abstract from the low-level hardware. The integration of hardware into the control loop facilitates the testing and evaluation of controller solutions in an efficient way. Furthermore, as low-level hardware aspects are avoided, it is possible to focus on control algorithm design.

Algorithms and tools are transferred to the controller without losing the advantages gained in the prototyping phase. In this sense, the same program is used for the simulation and for the real prototype. There is no need to make changes to the models or to the system's graphical user interface (GUI). The controller, after being embedded in the hardware, allows you to either work with the GUI or perform control without being connected to the GUI.

The natural modular capabilities of graphical, block-oriented programming allow for changes or modifications to selected sensors, actuators, or the control algorithm without requiring additional time.

Another important advantage of the proposed system is that it generates an optimized C source code and avoids the complexity of programming the algorithms and associated continuous-time solvers, such as Runge–Kutta.

At last, the system can be used for the implementation of control solutions in the teaching of control, where students can develop and test advanced algorithms in simulation and in real prototypes in a fast way without recoding. The platform is presented as a teaching platform for control engineering, converging mathematics, informatics, and mechanics skills, in order to fulfil STEAM competency development.

The experimental results show that it is possible to implement algorithms in embedded microcontrollers using the MATLAB/Simulink<sup>®</sup> tool to program and load the algorithm on it. Moreover, control actions are performed in real time on the physical system, making it possible to visualize their status and interact with them.

The main limitation of the proposal is the requirement for MATLAB and Simulink, including the need for a license. Additionally, users must acquire the proposed toolbox. For universities, this should not be a problem since most of them have agreements for student licenses.

**Author Contributions:** Conceptualization, S.G., A.F.-C. and R.B.; methodology, A.F.-C. and R.B.; software, A.F.-C. and D.R.R.; validation, D.R.R. and R.B.; formal analysis, S.G. and A.F.-C.; investigation, S.G., D.R.R., A.F.-C. and R.B.; writing—original draft preparation, A.F.-C., D.R.R. and R.B.; writing—review and editing, R.B., A.F.-C. and R.B.; supervision, S.G., A.F.-C. and R.B.; project administration, R.B.; funding acquisition, R.B. All authors have read and agreed to the published version of the manuscript.

**Funding:** The research leading to these results has received funding from RoboCity2030-DIH-CM, Madrid Robotics Digital Innovation Hub, S2018/NMT-4331, funded by “Programas de Actividades I+D en la Comunidad de Madrid” and cofunded by the European Social Funds (FSE) of the EU.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data presented in this study are available on request from the corresponding author. The data are not publicly available because liscence is needed.

**Acknowledgments:** We acknowledge the R&D&I project PLEC2021-007819 funded by MCIN/AEI/10.13039/501100011033 and by the European Union NextGenerationEU/PRTR and the Comunidad de Madrid (Spain) under the multiannual agreement with Universidad Carlos III de Madrid (“Excelencia para el Profesorado Universitario”—EPUC3M18), part of the fifth regional research plan 2016–2020.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Sandoval-Palomares, J. STEAM conceptions, competencies and attitudes in higher education: A pilot study. *J. Health Educ. Welf.* **2022**, *6*, 6–20. [CrossRef]
2. Henze, J.; Schatz, C.; Malik, S.; Bresges, A. How Might We Raise Interest in Robotics, Coding, Artificial Intelligence, STEAM and Sustainable Development in University and On-the-Job Teacher Training? *Front. Educ.* **2022**, *7*, 872637. [CrossRef]
3. González-Vargas, A.M.; Serna-Ramirez, J.M.; Fory-Aguirre, C.; Ojeda-Misses, A.; Cardona-Ordoñez, J.M.; Tombé-Andrade, J.; Soria-López, A. A low-cost, free-software platform with hard real-time performance for control engineering education. *Comput. Appl. Eng. Educ.* **2018**, *27*, 406–418. [CrossRef]
4. Scherler, S.; Liu-Henke, X. Model-based design of a multi-functional HiL test bench for investigations on a Range Extended Vehicle. In Proceedings of the 2018 IEEE International Systems Engineering Symposium (ISSE), Rome, Italy, 1–3 October 2018; pp. 1–7. [CrossRef]
5. Gan, C.; Todd, R.; Apsley, J. HiL Emulation for Future Aerospace Propulsion Systems. In Proceedings of the 7th IET International Conference on Power Electronics, Machines and Drives, PEMD 2014, Manchester, UK, 8–10 April 2014. [CrossRef]
6. Hogan, D.; Kelleher, B.; Foley, R.; Albiol-Tendillo, L.; Valdivia-Guerrero, V. Rapid-prototyping and hardware-in-the-loop laboratory platform for development and testing of electro-mechanical actuator controls. *J. Eng.* **2019**, *2019*, 4133–4137. [CrossRef]
7. Valera, A.; Soriano, A.; Vallés, M. Plataformas de bajo coste para la realización de trabajos prácticos de mecatrónica y robótica. In *Revista Iberoamericana de Automática e Informática Industrial*, 11th ed.; Elsevier: Amsterdam, The Netherlands, 2014; pp. 363–376.
8. Alexandre, E.; Cuadra, L.; Álvarez, L.; Rosa-Zurera, M.; López-Ferreras, F. Two-layer automatic sound classification system for conversation enhancement in hearing aids. In *Integrated Computer-Aided Engineering*, 15th ed.; IOS Press: Amsterdam, The Netherlands, 2008; pp. 85–94.
9. Flores-Caballero, A.; Marcos, A. Hardware in the Loop Test Bench suitable for New Space concept. In Proceedings of the 14th International Conference on Education and New Learning Technologies, Palma, Spain, 4–6 July 2022; pp. 223–231. [CrossRef]
10. IEC 61508; Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems. International Electrotechnical Commission: London, UK, 2010.
11. Schieferdecker, I.; Mosterman, P.J. *Model Based Testing for Embedded Systems*; CRC Press: Boca Raton, FL, USA, 2012.
12. Catelani, M.; Ciani, L.; Mugnaini, M.; Scarano, V.L.; Singuaroli, R. Definition of safety levels and performances of safety: Applications for an electronic equipment used on rolling stone. In Proceedings of the IEEE Instrumentation and Measurement Technology Conference, Warsaw, Poland, 1–3 May 2007.
13. Lundteigen, M.A.; Rausand, M. Assessment of hardware safety integrity requirements. In Proceedings of the 30th ESReDA seminar, Trondheim, Norway, 7–8 June 2006.
14. Iridin, M.; Aizpurua, X.; Villaro, A.; Melendez, J.; Legarda, J. Implementation details and safety analysis of a microcontroller-based SIL-4 software voter. *IEEE Trans. Ind. Electron.* **2011**, *58*, 822–829. [CrossRef]
15. Available online: <https://www.st.com/en/microcontrollers-microprocessors/stm32h7-series.html> (accessed on 30 October 2022).
16. Gaber, K.; El\_Mashade, M.B.; Abdel Aziz, G.A. Hardware-in-the-loop real-time validation of micro-satellite attitude control. *Comput. Electr. Eng.* **2020**, *85*, 106679. [CrossRef]
17. Hoyos-Gutiérrez, J.; Cardona-Aristizabal, J.; Muñoz-Gutiérrez, P.; Ramirez-Jimenez, D. A Systematic Literature Review on Rapid Control Prototyping Applications. *IEEE Rev. Iberoam. Technol. Aprendiz.* **2023**, *18*, 76–85. [CrossRef]
18. Tao, G. *Adaptive Control Design and Analysis*; Wiley-Interscience: Hoboken, NJ, USA, 2003.
19. Ljung, S.; Ljung, L. Error propagation properties of recursive least-squares adaptation algorithms. *Automatica* **1985**, *21*, 157–167. [CrossRef]
20. Ogata, K. *Modern Control Engineering*, 5th ed.; Pearson: London, UK, 2010.

21. Patel, V.V. Ziegler-Nichols Tuning Method. *Resonance* **2020**, *25*, 1385–1397. [[CrossRef](#)]
22. Clark, R.N. The Routh-Hurwitz stability criterion, revisited. *IEEE Control. Syst. Mag.* **1992**, *12*, 119–120.

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.