Sensors **2014**, *14*, 19176-19199; doi:10.3390/s141019176

OPEN ACCESS
sensors
ISSN 1424-8220
www.mdpi.com/journal/sensors

Article

Platform for Real-Time Simulation of Dynamic Systems and Hardware-in-the-Loop for Control Algorithms

Isaac D. T. de Souza, Sergio Natan, Rafael Teles and Marcelo A. C. Fernandes *

Department of Computer Engineering and Automation, Center of Technology, Federal University of Rio Grande do Norte—UFRN, Natal 59078-970, Brazil; E-Mails: isaacdiego@gmail.com (I.D.T.S.); s.natansilva@gmail.com(S.N.); rmteles@gmail.com (R.T.)

* Author to whom correspondence should be addressed; E-Mail: mfernandes@dca.ufrn.br; Tel.: +55-84-3215-3771; Fax: +55-84-3215-3738.

External Editor: Xue Wang

Received: 6 July 2014; in revised form: 3 September 2014 / Accepted: 30 September 2014 / Published: 15 October 2014

Abstract: The development of new embedded algorithms for automation and control of industrial equipment usually requires the use of real-time testing. However, the equipment required is often expensive, which means that such tests are often not viable. The objective of this work was therefore to develop an embedded platform for the distributed real-time simulation of dynamic systems. This platform, called the Real-Time Simulator for Dynamic Systems (RTSDS), could be applied in both industrial and academic environments. In industrial applications, the RTSDS could be used to optimize embedded control algorithms. In the academic sphere, it could be used to support research into new embedded solutions for automation and control and could also be used as a tool to assist in undergraduate and postgraduate teaching related to the development of projects concerning on-board control systems.

Keywords: real time; simulation; dynamic systems; embedded systems

1. Introduction

The development of new embedded algorithms and control techniques for dynamic systems, such as motors, industrial processes, automobiles and aircraft, amongst others, normally requires real-time tests in the devices to be controlled. However, such tests using real systems can be very expensive and time consuming. For example, the refinement of a control algorithm for an automobile injection valve requires tests under real conditions that can last for hours or even days. In other cases, it is necessary to construct small prototypes in order to perform the tests, as in the case of aircraft and ships. Many of these tests under real conditions can be performed using off-line simulations (which are not performed in real time), although such simulations do not fully eliminate real-time testing.

Methods for the development of platforms for real-time simulation have been extensively studied for many years. However, research in this area has recently accelerated following advances in terms of speed and the ease of development associated with new hardware platforms. Methodologies for real-time simulation have included the use of hardware, such as digital signal processors, general-purpose processors and even reconfigurable computational solutions employing field-programmable gate arrays (FPGAs) [1–9]. The benefits of using real-time simulation platforms have been demonstrated in various areas of graduate teaching [10,11]. The combination of real-time simulation with hardware-in-the-loop (HIL) has also been widely investigated in recent years and has brought many benefits in terms of the optimization and prototyping of new control systems [12–17]

In [1], a low-cost real-time simulator (RTS) that can be constructed in the laboratory is proposed for electronic power systems. The system is implemented in a computer with a multi-core processor coupled with data acquisition and signal conditioning boards. The studies described in [2,3] present distributed solutions for RTS in software, where several computers are used in an Ethernet network in order to simulate real-time systems. Signal processing techniques able to reduce the sampling frequency and computational cost associated with RTS are presented in [4], where it is shown that the computational cost can be reduced up to 100-fold, without compromising the precision of the simulation. In [5], a methodology consisting of seven steps is proposed for the validation of real-time simulation models. In [6], a real-time simulation model is proposed for testing AC electrical drives, where the simulation is performed using a Hypersim real-time simulator platform provided by the company, OPAL-RT [18]. Hypersim uses a supercomputer with more than 2000 cores, together with data acquisition boards, and is widely used to simulate large-scale power systems [19]. The use of onboard systems with DSPs is a new approach described in [7], where the dynamic system is implemented using DSPs.

The studies described in [8,9] offer RTS proposals for energy systems employing FPGAs associated with a computer, the main advantage of which is the parallel implementation of the dynamic systems simulated. The benefits of using RTS in the teaching of engineering in graduate and postgraduate courses are described in detail in the works of [10,11], where various commercial RTS tools are presented. Reviews of the advances achieved using real-time simulation schemes with HIL are presented in [12–14], where [13] focuses on applications in energy systems, and [14] focuses on aspects associated with the automotive field. In [15], a real-time simulation system is implemented for testing complex onboard systems with HIL, with the proposed RTS utilizing high-performance hardware with various data acquisition platforms. Finally, modeling and simulation methodologies associated with commercial

RTS platforms are presented in [16,17]. The work described in [16] used an RTS provided by the company, dSPACE [20], to simulate wind energy systems, and in [17], an eMEGAsim platform from OPAL-RT [18] was used in an all-electric ship project.

From the cited works, it can be seen that research involving RTS extends from new hardware and software proposals to new modeling techniques aimed at existing commercial systems. It can also be seen that the scope of this area is quite wide and that there is still considerable space for new solutions, especially in the context of simulations employing HIL. The proposal of the present work is therefore to present a new RTS and HIL platform applied to the optimization and testing of control algorithms. This new platform, here called the Real-Time Simulator for Dynamic Systems (RTSDS), possesses a distributed architecture composed of a range of different hardware platforms, enabling it to be constructed in the presence of various resource limitations. As an example, this paper presents a case study in which the RTSDS is composed of a personal computer (Intel DN2800MT Mini-ITX hardware) associated with low-cost and easily-accessible hardware platforms (with embedded software), such as the Arduino and CY8CKIT-001 kits [21–25]. It is important to emphasize that the RTSDS is a solution that does not provide a substitute for high-performance simulation platforms [18,20] that work with sampling times on the order of µs, although it can be used for a range of real systems that work with speeds on the order of ms [26].

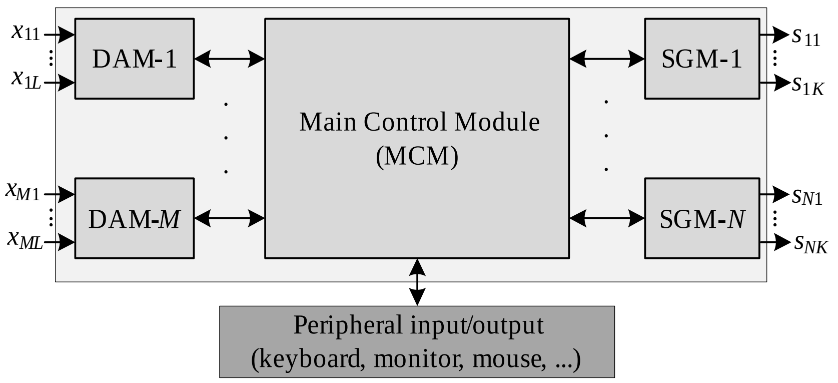
The main objective of the Real-Time Simulator for Dynamic Systems (RTSDS) is to provide a support tool for the developers of embedded control systems, focusing on reducing the costs involved in real-time tests. In addition to having industrial applications, the RTSDS could also be used in the academic sphere as a programmable didactic model in various curriculum modules of courses in engineering. Using the RTSDS, the graduate or postgraduate student could test control algorithms in various types of dynamic system, simply and at a cost that is relatively low compared to those of real didactic models. Another point that should be emphasized about the proposal presented here is that the RTSDS can also simulate the functioning of the drivers and sensors associated with the control of dynamic systems, which is an important characteristic from the point of view of the development of control systems projects [27–29].

In order to verify all of the relevant aspects of the RTSDS, this paper presents two case studies in which real-time simulations are made of a level control design and a design for the speed control of a vehicle following a longitudinal route. All of the details concerning the architecture and functioning of the RTSDS are presented here, and all of the source codes can be found in [30], hence enabling the use of this platform by other researchers and students.

2. Physical Architecture of the RTSDS

Figure 1 shows a block diagram of the architecture of the RTSDS. The main control module (MCM) is coupled to a set of M data acquisition modules (DAMs) and N signal generation modules (SGMs). The function of the MCM is to control, manage and configure the real-time simulation platform, and it is also responsible for the entry and retrieval of data by the user. In principle, the MCM should employ a high-performance processor, which in the present case was the Intel DN2800MT Mini-ITX [31], including an Atom N2800 processor.

Figure 1. Physical architecture of the Real-Time Simulator for Dynamic Systems (RTSDS) platform.



The DAMs and SGMs are auxiliary hardware whose function is to receive and generate signals related to the dynamic system to be simulated by the MCM. In practice, the DAMs and SGMs can be implemented by a variety of hardware platforms, such as digital signal processors, microcontrollers and other devices possessing digital and/or analog inputs and outputs. As shown in Figure 1, each m-th DAM can include L digital and/or analog inputs, and each n-th SGM is characterized by K digital and/or analog outputs. Each RTSDS must possess at least one DAM and one SGM.

2.1. Main Control Module (MCM)

The main control module (implemented with the Intel DN2800MT Mini-ITX hardware) is responsible for configuring and implementing the simulation of the chosen dynamic system. The MCM performs its task by means of an embedded application called the main application (MA), which will be described in the following sections. In addition to the MA, the MCM must possess a set of integrated interfaces and communication protocols in order to achieve communication with the DAMs and SGMs. The RTSDS platform does not demand any specific communication protocol, although the transmission rate R (in bps) is an important factor in the functioning of a real-time simulator, especially in the case of fast-response dynamic systems [27].

2.2. Data Acquisition Modules (DAMs)

The DAMs are hardware connected to the MCM by means of a data communication protocol transferring bits at a rate of R_1 bps. Each m-th DAM can comprise L inputs (analog and/or digital) and is responsible for capturing signals from the external environment that will be used by the real-time simulator executed in the MCM. The DAMs utilize an embedded software system, called the application DAM (ADAM). There is a different ADAM for each type of driver and actuator associated with the dynamic system to be simulated. The ADAM is responsible for transforming the L external signals connected to the m-th DAM into real values and transmitting them to the MCM. The l-th input signal of the m-th DAM is expressed by $x_{ml}(t)$ and, depending on the driver, may be a variable amplitude and/or frequency analog signal or a variable frequency and/or pulse width digital signal [28].

2.3. Signal Generation Modules (SGMs)

These modules are hardware platforms responsible for the generation of signals associated with the outputs of the dynamic system simulated in the MCM. Similar to the DAMs, the SGMs are comprised of K outputs (analog and/or digital), communicate with the MCM by means of a data communication protocol at a rate of R_2 bps and employ an embedded software system called the application SGM (ASGM), which converts the simulation data received from the MCM into external analog and/or digital signals. For each type of sensor to be simulated in the dynamic system, there is a different ASGM that converts the output information according to the instrumental characteristics of the sensor in question. The k-th output signal of the n-th SGM is expressed as $s_{nk}(t)$ and, depending on the type of sensor, may be an analog signal of variable amplitude and/or frequency or a digital signal with variable frequency and/or pulse width [29].

3. Logical Architecture of the RTSDS

Figure 2 illustrates the logical architecture of the RTSDS, where an application located in the MCM, here known as the main application (MA), instantiates a class of software called the object dynamic system (ODS). An ODS represents the dynamic system to be simulated in real time and is composed, amongst others, of eight principal attributes expressed by $\{P, H, f(t), \mathcal{M}, \mathcal{C}, \mathcal{D}, \mathit{MethodODE}, t_a\}$, where:

- P represents the number of inputs of the dynamic system.
- H represents the number of outputs of the dynamic system.
- $f(t)$ defines the system of ordinary differential equations (ODEs) of the dynamic system to be simulated in real time.
- $\mathcal{M}$ defines a set of adjustable parameters associated with the dynamic system.
- $\mathcal{C}$ characterizes the set of embedded codes (the ADAMs) in the DAMs associated with the inputs of the dynamic system.
- $\mathcal{D}$ characterizes the set of embedded codes (the ASGMs) in the SGMs associated with the outputs of the dynamic system.
- *MethodODE* is an attribute that defines which ODE numerical resolution method should be used in the real-time simulation. The RTSDS can implement various resolution methods, including those of Euler, Runge–Kutta, and others [32,33].
- t_a represents the sampling time to be employed in the real-time simulation process.

The RTSDS possesses a database, located in the MCM, composed of various ODSs that can be easily inserted by means of the MA.

The execution of the real-time simulation of an ODS, in the MCM, is achieved using three threads working collaboratively. These are the data acquisition thread (DAT), the simulation thread (ST) and the time control thread (TCT). The function of the DAT is to update all of the P inputs of the dynamic system under simulation, reading all of the input buffers associated with the data connections between the MCM and the DAMs. The P input variables are shared between the data acquisition and simulation threads. The ST updates the H output variables and executes the ODE numerical resolution method (defined by the *MethodODE* variable) in the system of equations, $f(t)$, in order to update the states of the dynamic system [27,32,33]. The execution of the simulation thread is controlled by the TCT, which uses semaphore logic [34] to regulate the execution time of the ST to a fixed value, t_s, so that the resolution method always occurs at fixed intervals. The use of a fixed interval, t_s, is not essential in numerical resolution methods, but is fundamental for parameterization of the simulation platform, which must have a sampling time, t_a, that is much smaller than the response time, t_r, of the dynamic system to be simulated. In other words, the simulation process must respect the restriction:

$$t_a << t_r \tag{1}$$

Figure 3 illustrates the functional relationships between the DAT, ST and TCT.

Figure 2. Logical architecture of the RTSDS platform.

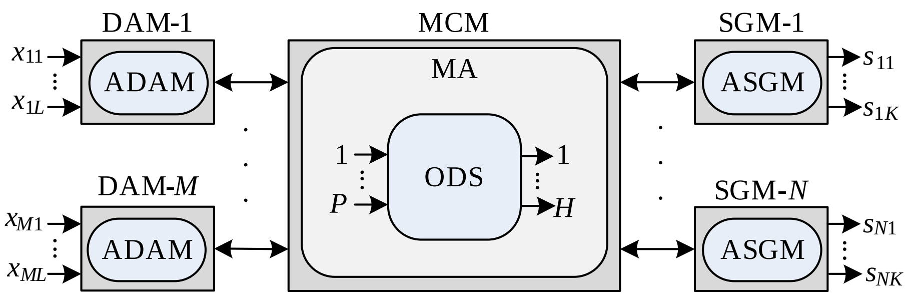
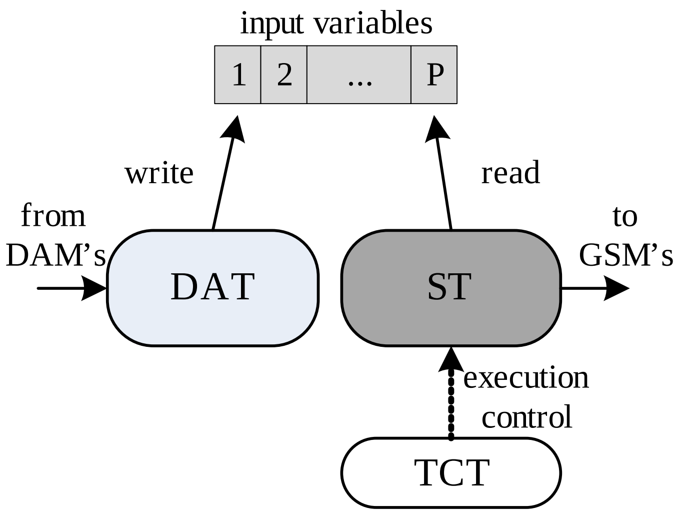


Figure 3. Relationships between the data acquisition thread (DAT), the simulation thread (ST) and the time control thread (TCT).



Figures 4–6 detail the finite automata associated with the threads DAT, TS and TCT, respectively. The DAT (Figure 4) scans all communication interfaces buffers with DAMs and updates the P input variables of ODS (Figure 3). The DAT is asleep in a predetermined fixed time, t^{w}_{DAT}, to minimize collision problems in the input variables' update (write operation), since they are shared with the ST. The finite automata of ST (detailed in Figure 5) wait for the semaphore resource update (by TCT) to send the output variables to SGMs and then run the ODS. Finally, Figure 6 shows, by finite automata, the operation of TCT, which controls the insertion of the semaphore resource. The resource addition is

done every, t_s, seconds to ensure the run time of the ODS. The finite automata shown in Figure 6 also detail a system synchronization (using the variables d_a and d) that adjusts the run time in t_s.

The codes of the set $\mathcal{C}$ associated with an ODS act as transducers between the external input signal (which can be digital or analog) and one of the P inputs of the dynamic system. Each ODS has a one to many relationship with the DAMs and, if necessary, can use all of the M DAMs available. The decision of which and how many DAMs will be used by the ODS is made off-line by means of configuration parameters, with the objective of customizing the performance of the system in order to reduce the sampling time, t_a. An important point is that the DAMs can simulate the effect of real drivers and actuators, for example the commercial drivers that work with signals possessing variable frequency and pulse width.

Similar to the DAMs, the role of the set $\mathcal{D}$ of codes related to the ODS is to act as transducers between the outputs of the dynamic system and the external signals (digital and/or analog) to be generated. Each ODS also has a one to many relationship with the SGMs and can, if necessary, utilize all N SGMs associated with the RTSDS. The configuration of which and how many SGMs will be used during the real-time simulation process is configured off-line, as for the DAMs. The SGMs can also simulate the responses of real commercial sensors with variable frequency, amplitude or pulse width.

Figure 4. Finite automata associated with DAT.

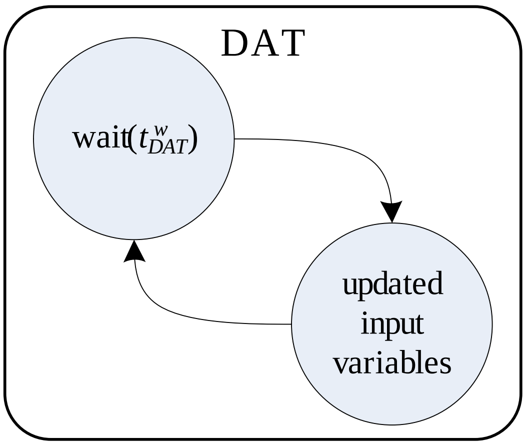


Figure 5. Finite automata associated with ST.

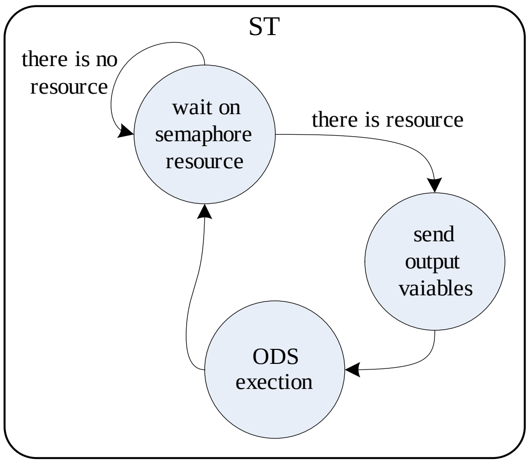
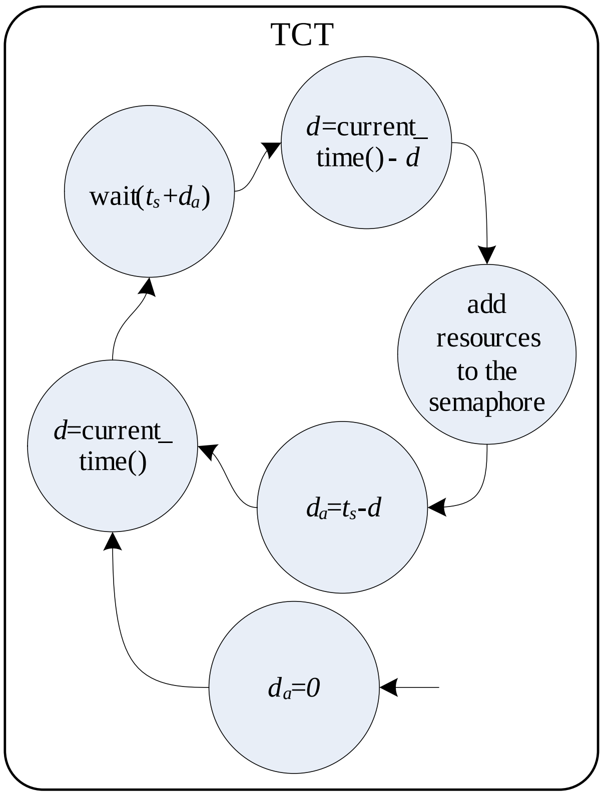


Figure 6. Finite automata associated with TCT.



The simulation of drivers and sensors (using the DAMs and SGMs, respectively) coupled to the dynamic system differentiates the proposed platform, with the simulation process becoming closer to the real case. This benefit is especially important in the development and validation of the embedded control of dynamic systems. Future work will consider the optimization of algorithms for the automatic allocation of DAMs and SGMs to the MCM.

After instantiation of an ODS, the sampling time, t_a, can be expressed by:

$$t_a = t_{DAM} + t_1 + t_{DAT} + t_s + t_2 + t_{SGM} \tag{2}$$

where t_{DAM} is the fastest processing time associated with all of the DAMs utilized by the MCM, t_1 is the time corresponding to the slowest rate of transfer (in bps) between the MCM and all of the DAMs utilized, t_{DAT} is the time taken by the DAT to read the information received by all of the DAMs, t_s is the time required for execution of the ST, t_2 is the slowest transfer time (in bps) between the MCM and the SGMs and t_{SGM} is the fastest processing time associated with all of the SGMs utilized by the MCM. The transmission time, t_1, between a DAM and the MCM can be described by:

$$t_1 = \frac{b_1}{R_1} \tag{3}$$

where b_1 is the resolution (in bits) of the value to be transmitted. Similarly, the transmission time between the MCM and the SGM can be expressed by:

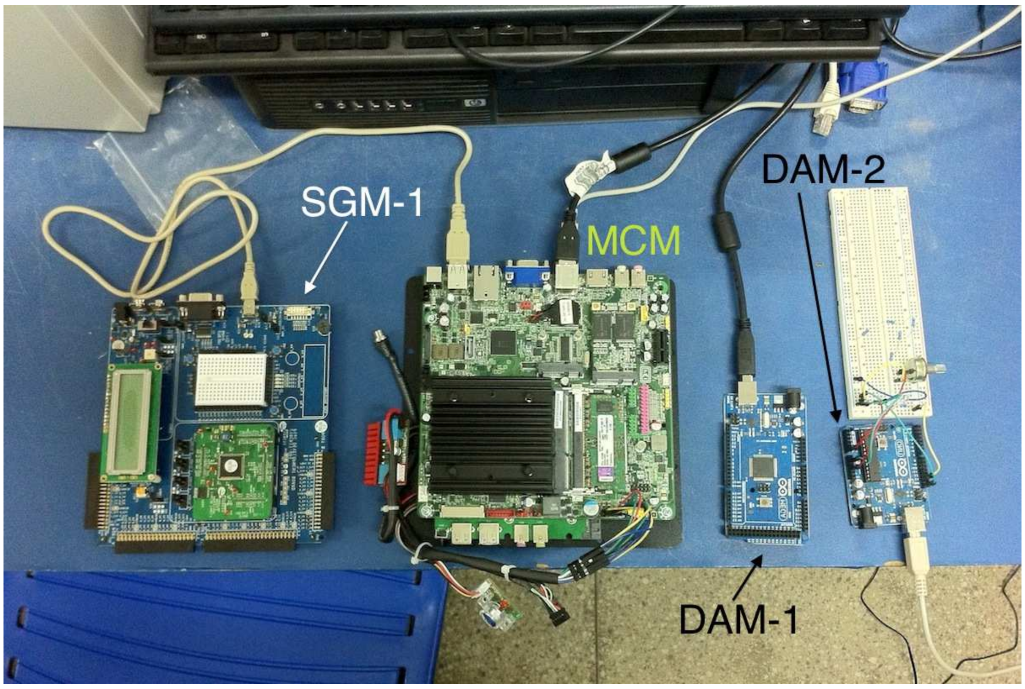
$$t_2 = \frac{b_2}{R_2} \tag{4}$$

where b_2 is the resolution (in bits) of the value transmitted.

4. Description of the Prototype

Validation of the proposed system involved the development of a prototype (illustrated in Figure 7) with $M = 2$ DAMs (DAM-1 and DAM-2) and $N = 1$ SGM (SGM-1). The MCM was implemented using the Intel DN2800MT Mini-ITX platform with the Linux operating system (distribution: Ubuntu 13.10 [35]), and the DAMs and SGMs were implemented in microcontrollers. DAM-1 utilized an Atmel ATmega 2560 processor [36] with the Arduino Mega 2560 development kit [21]; DAM-2 used an Atmel ATmega 328p processor [22] with the Arduino Uno development kit [23]; and the SGM utilized a Cypress PSOC 3 CY8C38 processor [24] with the PSOC CY8CKIT-001 kit [25]. Communication between the two DAMs and the MCM was accomplished using the USART protocol (Universal Synchronous and Asynchronous Serial Receiver and Transmitter) at a rate of $R_1 = 9.6$ kbps, and communication between the SGM and the MCM was by means of the USB (Universal Serial Bus) protocol, at a rate of $R_2 = 1$ Mbps. The main application of the MCM in the prototype was implemented with only one available ODS.

Figure 7. Prototype RTSDS composed of main control module (MCM) data (Intel DN2800MT Mini-ITX), two acquisition modules (DAMs) (Arduino Mega 2560 kit) and a signal generation module (SGM) (PSOC CY8CKIT-001 kit).



After the development of the prototype, the validation process was continued with the simulation of two dynamic systems whose characteristics are presented in the following subsections. The results associated with the simulations and HIL testing with a PID controller are presented in the next section.

4.1. Dynamic System 1: Water in a Tank

This dynamic system is illustrated in Figure 8, and the results are presented in the next section. The real-time simulation involved a tank of water coupled to a pump (see Figure 8). The pump with constant flow, q_m, is coupled to an input valve with continuous control, $V_e(t)$, generating a tank input flow of $q_e(t)$. The tank output valve also possesses a continuous control, $V_o(t)$, and the flow after this valve is given by $q_s(t)$. The level of water in the tank is characterized by the $n(t)$ variable. In this case, the ODS (Figure 9) possesses two inputs, $P = 2$, controlling the input valve, $V_e(t)$, and the output valve, $V_o(t)$, and one output, $H = 1$, which is the water level in the tank, $n(t)$. The system of differential equations associated with the ODS can be expressed by:

$$f(t) = \begin{cases} q_e(t) - V_o(t)q_s(t) = A\frac{dn(t)}{dt} \\ q_e(t) = q_m V_e(t) \\ q_s(t) = a\sqrt{2gn(t)} \end{cases} \tag{5}$$

where $q_m(t)$, A, g and a are the set of adjustable parameters, $\mathcal{M}$, of the ODS, representing the pump flow (cm^2/s), the transverse section of the tank (cm^2), the gravitational acceleration (m/s^2) and the transverse section of the tank output tube (cm^2), respectively.

The dynamic system was simulated in real time, employing the Euler method for ODE resolution (the *MethodODE* variable), which is a simple, but effective method for the dynamic system represented by Equation (5). In this method, Equation (5) can be rewritten as:

$$n(t) = \begin{cases} z(t) \text{ for } z(t) < n_{max} \\ n_{max} \text{ for } z(t) \geq n_{max} \end{cases} \tag{6}$$

where n_{max} is maximum water level in the tank and:

$$z(t) = n_0 + \int_{t_0}^{t} h_{tl}\left(s, n(s)\right) ds,\ t > t_0 \text{ e } n_0 \geq 0 \tag{7}$$

where n_0 is the initial condition of the tank level and $h_{tl}\left(s, n(s)\right)$ is expressed as:

$$h_{tl}\left(s, n(s)\right) = \frac{1}{A}\left(q_m V_e(s) - V_o(s) a\sqrt{2gn(s)}\right) \tag{8}$$

The Euler method approximates the integral, presented in Equation (7), by the area of a rectangle whose base has length Δt [32,33], in other words,

$$\begin{aligned} z(t + \Delta t) &= n_0 + \int_{t_0}^{t} g\left(s, n(s)\right) ds + \int_{t}^{\Delta t} g\left(s, n(s)\right) ds \\ &\approx z(t) + g\left(t, n(t)\right) \times \Delta t \end{aligned} \tag{9}$$

where Δt is the step of the Euler method [32,33]. For RTSDS, Δt is the run time of the ST, $t_s = \Delta t$. Thus, the resolution method implemented in the prototype can be expressed as:

$$n(t + t_s) = \begin{cases} v(t) \text{ for } v(t) < n_{max} \\ \\ n_{max} \text{ for } v(t) \geq n_{max} \end{cases} \tag{10}$$

where,

$$v(t) = n(t) + \left(\frac{1}{A}\left(q_m V_e(t) - V_o(t)a\sqrt{2gn(t)}\right)\right) \times t_s \tag{11}$$

Figure 8. Dynamic system simulated in real time using the prototype.

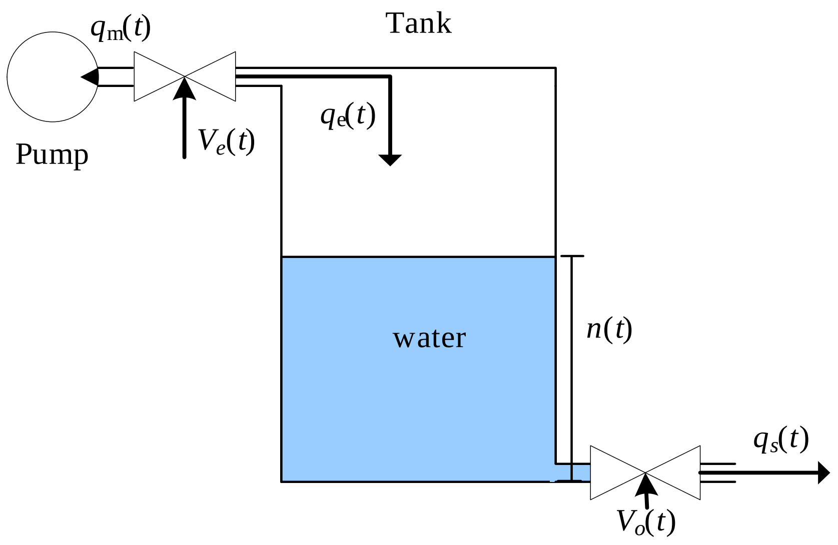
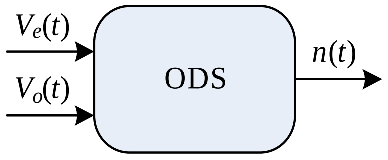


Figure 9. Instantiated object dynamic system (ODS) object composed of two inputs, $P = 2$ (control of the input valve, $V_e(t)$, and control of the output valve, $V_o(t)$), and one output, $H = 1$ (level of water in the tank, $n(t)$).



4.2. Dynamic System 2: Longitudinal Vehicle

The longitudinal vehicle model [37], illustrated in Figure 10, can be described by the expression:

$$M\frac{dx(t)}{dt} = f_t(t) - f_a(t) \tag{12}$$

where M is the mass of the vehicle (Kg), $x(t)$ is the linear velocity of the vehicle (m/s), $f_t(t)$ is the traction force of the vehicle (N) and $f_a(t)$ is the friction force (N). The traction force, $f_t(t)$, is expressed as:

$$f_t(t) = \frac{\tau_m(t)}{r} \tag{13}$$

where r is the radius of the wheel of the vehicle (m) and $\tau_m(t)$ is the torque (Nm) generated by the motor, expressed as:

$$\tau_m(t) = V_e(t)\tau_m^{max} \tag{14}$$

where τ_m^{max} is the maximum torque (Nm) and $V_e(t)$ is a continuous control signal, similar to Dynamic System 1 (water in a tank).

According to [37], the friction force, $f_a(t)$, can be expressed as:

$$f_a(t) = f_d(t) + f_r(t) + f_g(t)\sin(\theta(t)) \tag{15}$$

where $f_d(t)$ is the aerodynamic friction force (N), $f_r(t)$ is the rolling resistance force (N), $f_g(t)$ is the gravitational force (N) and $\theta(t)$ is the inclination angle of the plane on which the vehicle is located. The variable $\theta(t)$ is expressed as:

$$\theta(t) = V_o(t)\theta^{max} \tag{16}$$

where θ^{max} is the maximum inclination angle and $V_o(t)$ is a continuous control signal, similar to Dynamic System 1 (water in a tank).

The aerodynamic friction can be described by:

$$f_d(t) = \frac{1}{2}\rho C_d A_{fr} x^2(t) \tag{17}$$

where ρ is the density of air, C_d is the aerodynamic drag coefficient and A_{fr} is the frontal area of the vehicle (m^2). The rolling resistance force can be described by:

$$f_r(t) = Mg\left(C_0 + C_1 x^2(t)\right) \tag{18}$$

where C_0 and C_1 are the rolling coefficients and g is the acceleration due to gravity (m/s^2). Finally, the gravitational force is given by:

$$f_g(t) = Mg \tag{19}$$

The dynamic system was simulated in real time, employing the Euler method for ODE resolution (the *MethodODE* variable). In this method, Equation (12) can be rewritten as:

$$x(t) = \begin{cases} u(t) \text{ for } u(t) < x_{max} \\ x_{max} \text{ for } u(t) \geq x_{max} \end{cases} \tag{20}$$

where x_{max} is maximum vehicle speed and:

$$u(t) = x_0 + \int_{t_0}^{t} h_{vs}\left(s, x(s)\right) ds,\ t > t_0 \text{ e } x_0 \geq 0 \tag{21}$$

where x_0 is the initial condition of the vehicle speed and $h_{vs}\left(s, x(s)\right)$ is expressed as:

$$h_{vs}\left(s, x(s)\right) = \frac{1}{M}\left(V_e(s)\frac{\tau_m^{max}}{r} - \frac{1}{2}\rho C_d A_{fr} x^2(s) - Mg\left(C_0 + C_1 x^2(s)\right) - Mg\sin(V_o(s)\theta^{max})\right) \tag{22}$$

The Euler method approximates the integral, presented in Equation (21), by the area of a rectangle whose base has length Δt [32,33], in other words,

$$\begin{aligned} u(t+\Delta t) &= x_0 + \int_{t_0}^{t} h_{vs}\left(s, x(s)\right) ds + \int_{t}^{\Delta t} h_{vs}\left(s, x(s)\right) ds \\ &\approx u(t) + h_{vs}\left(t, x(t)\right) \times \Delta t \end{aligned} \tag{23}$$

where Δt is the step of the Euler method [32,33]. For RTSDS, Δt is the run time of the ST, $t_s = \Delta t$. Thus, the resolution method implemented in the prototype can be expressed as:

$$x(t+t_s) = \begin{cases} v(t) \text{ for } c(t) < x_{max} \\ \\ x_{max} \text{ for } c(t) \geq x_{max} \end{cases} \tag{24}$$

where,

$$c(t) = x(t) + \left(\frac{1}{M} \left(V_e(t) \frac{\tau_m^{max}}{r} - \frac{1}{2} \rho C_d A_{fr} x^2(t) - Mg \left(C_0 + C_1 x^2(t) \right) - Mg \sin(V_o(t)\theta^{max}) \right) \right) \times t_s \tag{25}$$

In this case, the ODS (Figure 11) possesses two inputs, $P = 2$, controlling the traction force, $V_e(t)$, as well as the inclination angle, $V_o(t)$, and one output, $H = 1$, which is the speed of the vehicle in the longitudinal direction, $x(t)$.

Figure 10. Schematic of the longitudinal vehicle model.

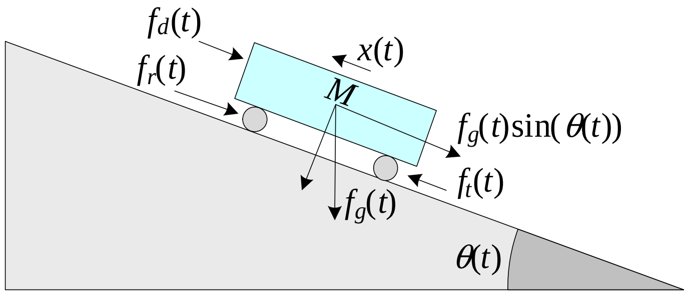
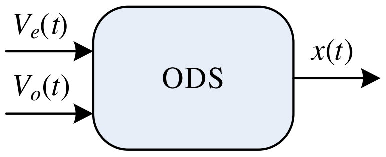


Figure 11. Instantiated ODS object composed of two inputs, $P = 2$ (control of the traction force, $V_e(t)$, and control of the inclination angle, $V_o(t)$), and one output, $H = 1$ (speed of the vehicle in the longitudinal direction, $x(t)$).



4.3. General Parameters

The control signal, $V_e(t)$, can have values of between zero and one in the ODS. In order to represent a more real situation, the external signal, $x_{11}(t)$, associated with $V_e(t)$ consists of a digital signal with a fixed frequency and a pulse width that can be varied between 0% and 100% (pulse-width modulation, PWM) [28]. Hence, the DAM-1, using an embedded ADAM (set $\mathcal{C}$ of codes) in the ATmega 2560, can recognize the PWM pulse width $x_{11}(t)$ in real time and convert it to a value of between zero and one, which is transmitted to the MCM via the USART protocol at a rate of $R_1 = 9.6$ kbps. The resolution of the value associated with $V_e(t)$ in conversion of the PWM signal was eight bits ($b_1 = 8$), in order to simplify the transmission process, resulting in a transmission time of 0.83 ms. The embedded application in the DAM-1 had an execution time of around 110 ms, due to the frequency of 100 Hz utilized in the PWM.

In the case of the control signal, $V_o(t)$, the ODS also operates with values of between zero and one. However, in order to differentiate this case from the preceding case, the external signal, $x_{21}(t)$, associated with $V_o(t)$, has an amplitude that is variable between zero and 2.5 volts. In this way, the DAM-2, using an embedded ADAM (set $\mathcal{C}$ of codes) in another ATmega 2560, recognizes the amplitude of the signal $x_{21}(t)$ in real time and converts it to a value of between zero and one for transmission to the MCM by means of the USART protocol at a rate of $R_1 = 9.6\,\text{kbps}$. The resolution of the value associated with $V_o(t)$ in the analog/digital conversion was eight bits ($b_1 = 8$), resulting in a transmission time of $0.83\,\text{ms}$. The embedded application in the DAM-2 had an execution time of around $20\,\text{ms}$.

Finally, the output of the simulated dynamic system, $n(t)$ (level in the tank) or $x(t)$ (vehicle speed), was associated with an analog signal, $s_{11}(t)$, in SGM-1. In this case, the embedded ASGM (set $\mathcal{D}$ of codes) in the SGM-1 simulates an analog sensor of a type widely used in industry, with output values of between zero and one volt. A resolution of eight bits was used for the value of the level output, $n(t)$, with a transmission rate of 1 Mbps (via USB protocol) between the MCM and SGM-1, resulting in a transmission time of around $8\,\mu\text{s}$. The resolution of the digital/analog converter in SGM-1 was eight bits, and the execution time of the ASGM was approximately $1\,\text{ms}$.

Table 1 summarizes the times estimated for the DAMs and SGMs, based on the variables presented in Equation (2). From these values and the response time of the dynamic system, t_r, it is possible to estimate the value of t_s and calculate the sampling time of the system, t_a.

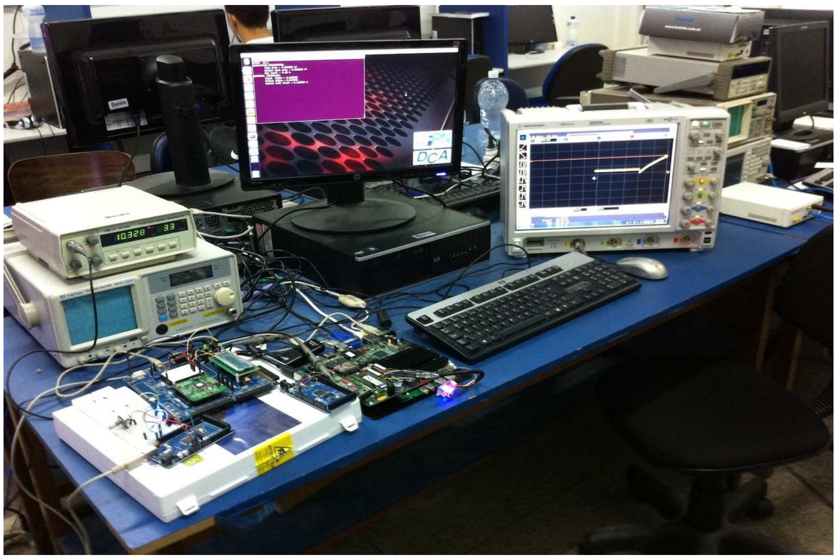
Table 1. Estimated times for the DAMs and SGMs, based on the variables presented in Equation (2).

Times	DAM-1	DAM-2	SGM-1	**MCM**	**Selected**
t_{DAM}	$110\,\text{ms}$	$20\,\text{ms}$	-	-	$110\,\text{ms}$
t_1	$0.83\,\text{ms}$	$0.83\,\text{ms}$	-	-	$0.83\,\text{ms}$
t_{SGM}	-	-	$1\,\text{ms}$	-	$1\,\text{ms}$
t_2	-	-	$8\,\mu\text{s}$	-	$8\,\mu\text{s}$
t_{DAT} ($t^w_{DAT} = 100\,\mu\text{s}$)	-	-	-	$200\,\mu\text{s}$	$200\,\mu\text{s}$

5. Experimental Results

The prototype developed was validated experimentally by varying the set of parameters, $\mathcal{M} = \{q_m(t), A, g, a\}$, of the dynamic system characterized by Equation (5). The results of these tests were compared with those obtained using the MATLAB/Simulink platform [38] in simulations that were not performed in real time. In the case of Dynamic System 1, the values employed for the parameters of the set $\mathcal{M}$ were based on a didactic kit of coupled tanks, supplied by Quanser [39], which is used in disciplines involving control and automation systems. For Dynamic System 2, the values are based on previous field tests of electric vehicles [40,41]. Figure 12 shows the equipment used to perform the experimental tests.

Figure 12. Equipment used to perform the experimental tests.



5.1. Results of Dynamic System 1: Water in a Tank

In all of the experiments with Dynamic System 1, the pump flow was held constant at $q_m(t) = 69\,\text{cm}^3/\text{s}$; the acceleration due to gravity was $g = 9.8\,\text{m}/\text{s}^2$; the maximum level of the tank was $n_{max} = 30\,\text{cm}$; and the transverse section of the output tube was $a = 0.1781\,\text{cm}^2$, generating a maximum outlet flow, $q_s(t)$, of $17.81\,\text{cm}^3/\text{s}$. The tests used values of $A = 15.518\,\text{cm}^2$ and $A = 62.072\,\text{cm}^2$ for the area of the transverse section of the tank, so that in the worst case, the response time of the dynamic system, t_r, was in the region of $t_r \approx 100\,\text{s}$. Based on the value of t_r and the values presented above (summarized in Table 1), use of Equation (2) gives an estimated value of t_s of $100\,\text{ms}$, while the calculated value of t_a was $212\,\text{ms}$, taking into account the restriction presented in Equation (1).

Figures 13–17 show the results obtained for the RTSDS platform used to execute the prototype described above, together with the non-real-time results obtained using MATLAB/Simulink. In all cases (with Dynamic System 1), it can be seen that the RTSDS platform provided satisfactory results, with errors that were very small in relation to the reference method. In all of the tests with Dynamic System 1 (water in a tank) (Figures 13–17), the outlet flow volume was fixed at the maximum value, $V_o(t) = 1$, and the inlet flow volume, $V_e(t)$, was adjusted between zero and one, varying the inlet flow rate, $q_e(t)$, relative to the constant flow produced by the pump, $q_m(t)$ (see Equation (5)).

The results for a tank with $A = 15.518\,\text{cm}^2$ are shown in Figures 13 and 14 for $q_e(t) = 0.457$ and $q_e(t) = 0.6$, respectively. Figure 15 presents the results for a programmed sequence of perturbations, with DAM-1 being programmed to begin with $q_e(t) = 0.457$ and then after 100 s to change to $q_e(t) = 0.6$. Figures 16 and 17 present the results for a tank with $A = 62.072\,\text{cm}^2$ and $q_e(t) = 0.3$ and $q_e(t) = 0.45$, respectively.

Figures 18 and 19 present the results obtained for an embedded PID controller [27] in an ATmega 2560 microcontroller (utilizing the Arduino Mega 2560 kit) external to the RTSDS platform. The system was controlled to a reference value (set point) of $15\,\text{cm}$, and the results show the response of the system in terms of the value associated with the level, $n(t)$. This experiment clearly demonstrated the benefits derived from using the RTSDS in the development of embedded control algorithms, where the performance and viability of new algorithms could be tested in real time, without any need to acquire expensive physical equipment. Figure 20 shows the equipment used in testing with the PID.

Figure 13. Results obtained for the RTSDS platform applied to the dynamic system characterized by Equation (5), with $A = 15.518\,\text{cm}^2$ and $q_e(t) = 0.457$.

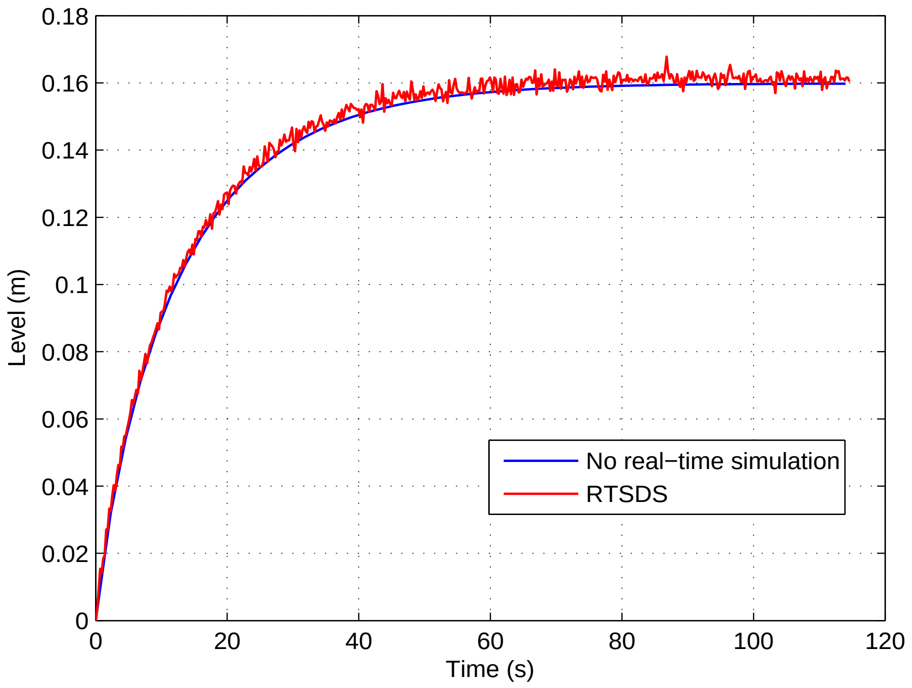


Figure 14. Results obtained for the RTSDS platform applied to the dynamic system characterized by Equation (5), with $A = 15.518\,\text{cm}^2$ and $q_e(t) = 0.6$.

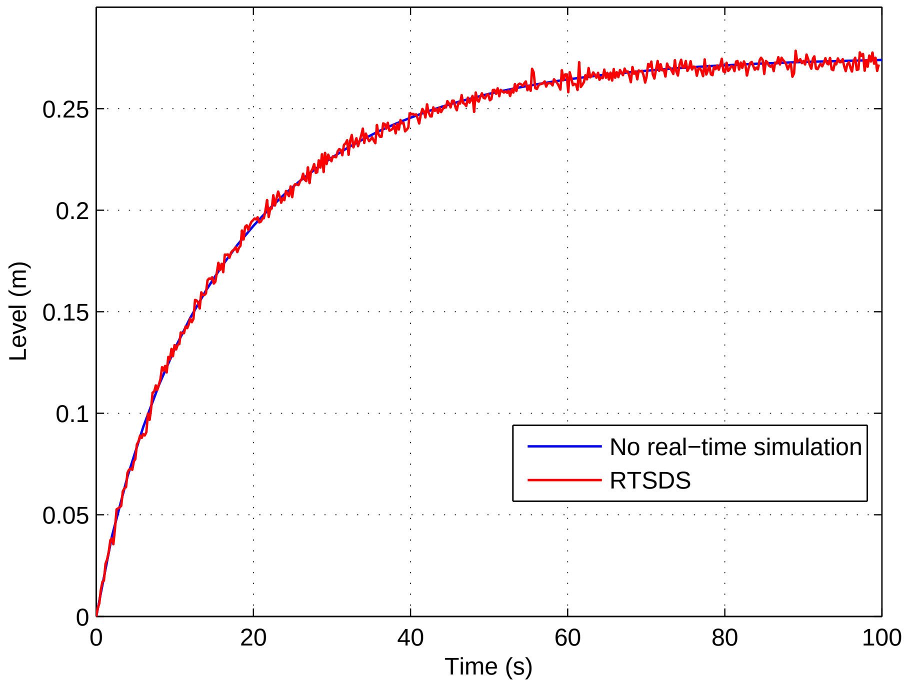


Figure 15. Results obtained for the RTSDS platform applied to the dynamic system characterized by Equation (5), with $A = 15.518\,\text{cm}^2$ and $q_e(t) = 0.457$ initially and then $q_e(t) = 0.6$ after 100 s.

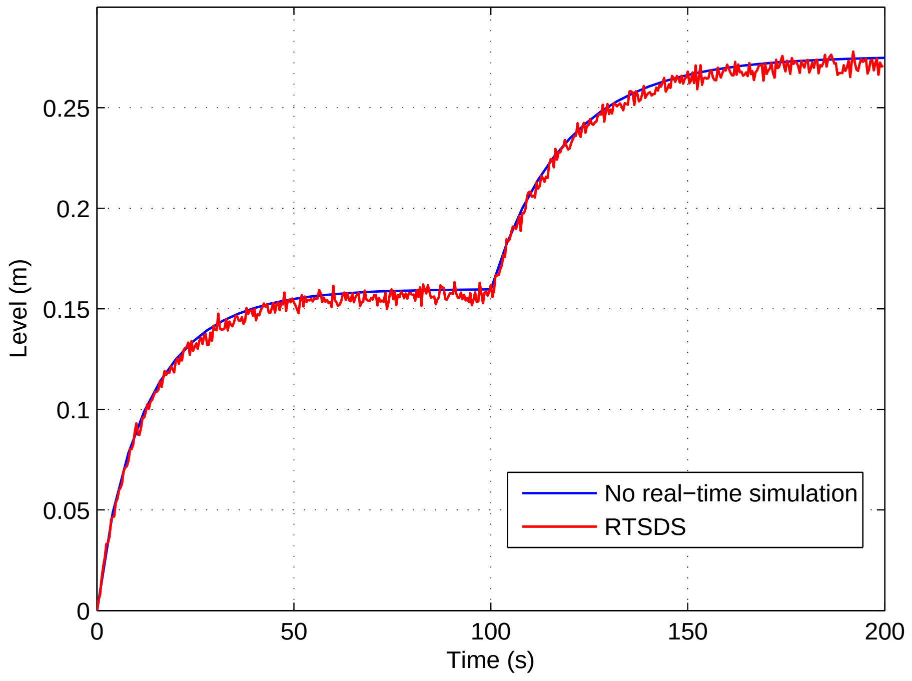


Figure 16. Results obtained for the RTSDS platform applied to the dynamic system characterized by Equation (5), with $A = 62.072\,\text{cm}^2$ and $q_e(t) = 0.3$.

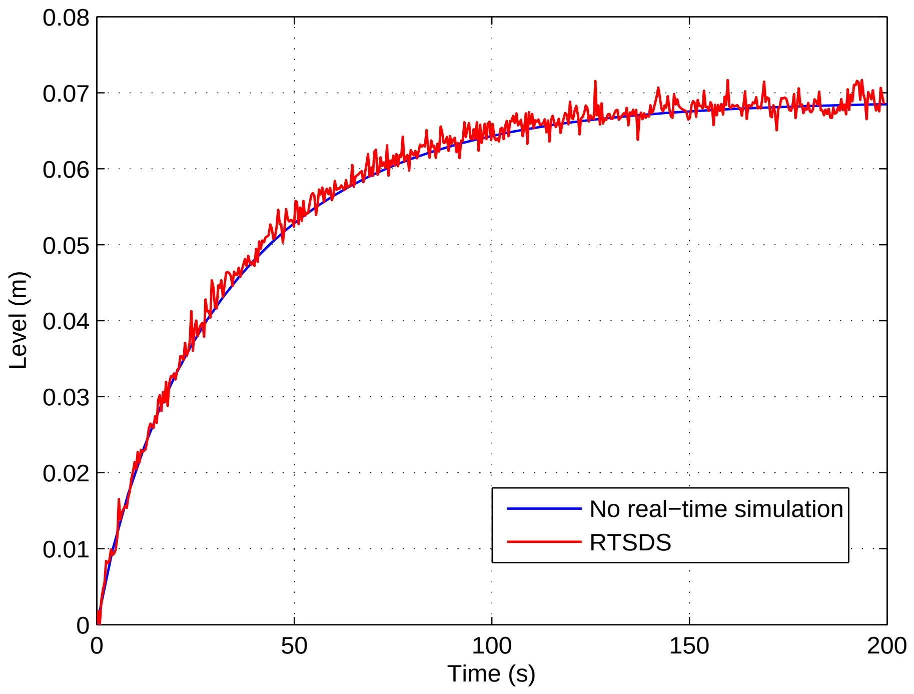


Figure 17. Results obtained for the RTSDS platform applied to the dynamic system characterized by Equation (5), with $A = 62.072\,\text{cm}^2$ and $q_e(t) = 0.45$.

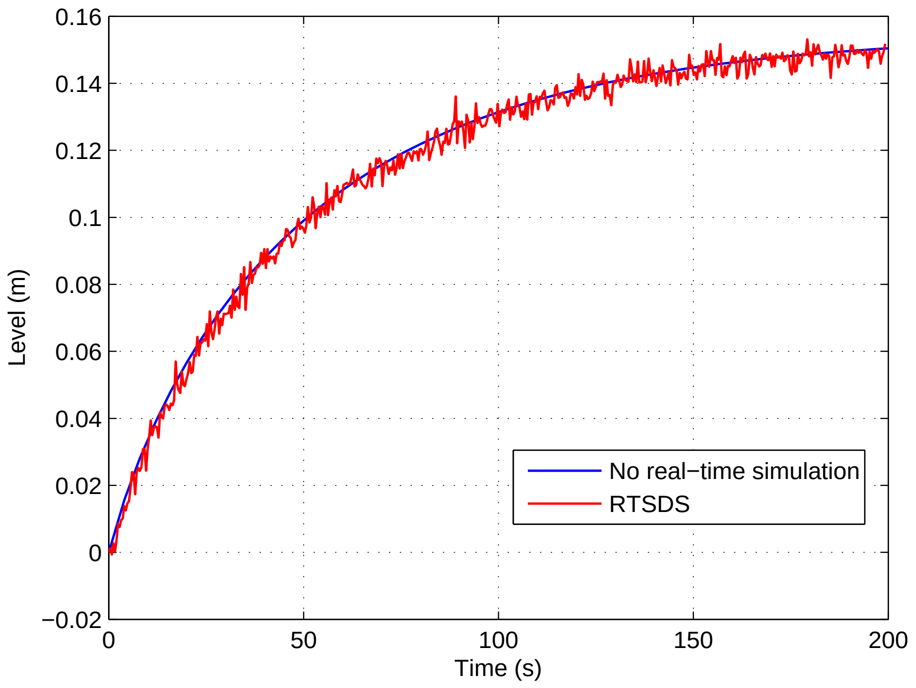


Figure 18. Results obtained for the RTSDS platform utilizing an external PID controller ($k_p = 2, k_i = 0.2$ e $k_d = 0.001$) onboard an ATmega 2560 microcontroller.

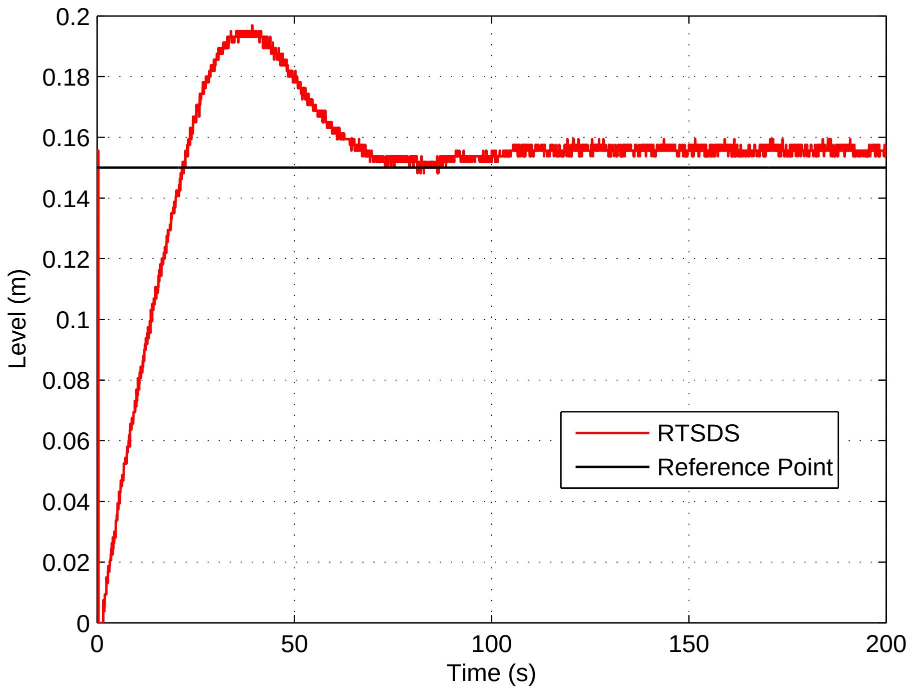


Figure 19. Results obtained for the RTSDS platform utilizing an external PID controller ($k_p = 5, k_i = 0.1$ e $k_d = 0.03$) onboard an ATmega 2560 microcontroller.

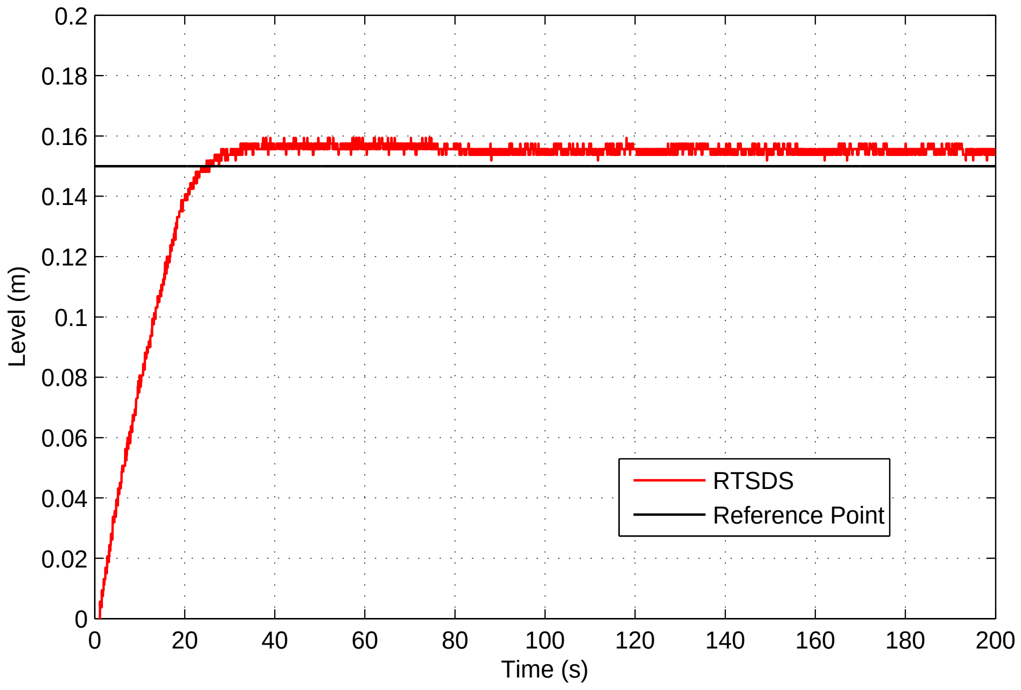
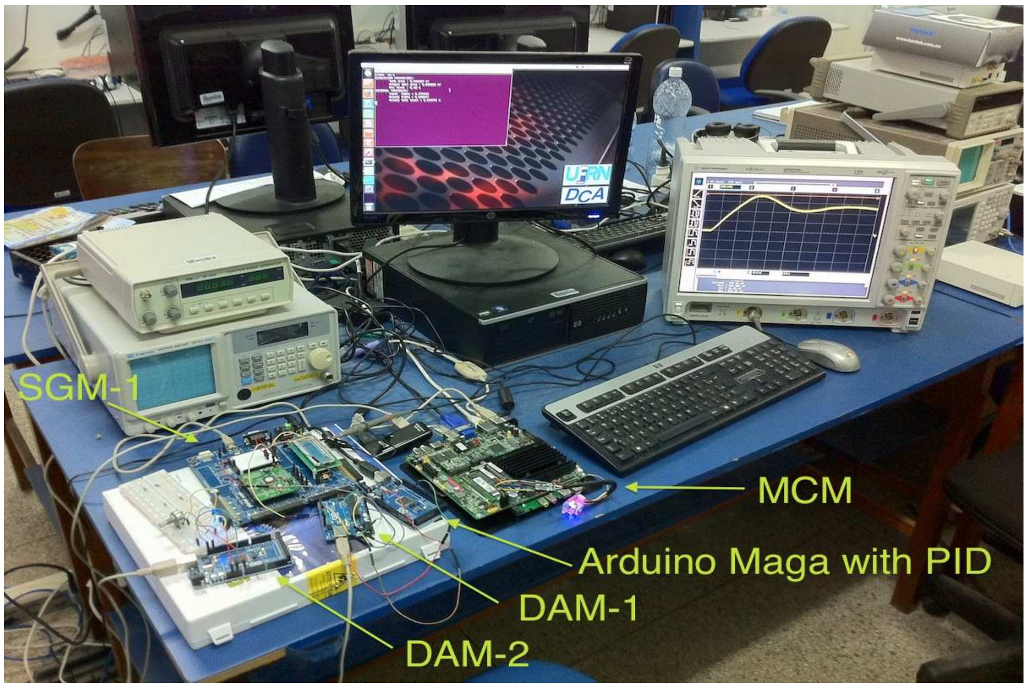


Figure 20. Equipment used in testing with the PID.



5.2. Results of Dynamic System 2: Longitudinal Vehicle

In all of the experiments with Dynamic System 2, the maximum torque was held constant at $\tau_m^{max} = 2000\,\mathrm{Nm}$; the acceleration due to gravity was $g = 9.8\,\mathrm{m/s^2}$; the maximum speed of the vehicle was $x_{max} = 200\,\mathrm{m/s}$; the mass of the vehicle was $M = 450\,\mathrm{Kg}$; the density of air was $C_d = 1.18$; the aerodynamic drag coefficient was $C_d = 0.51$; the frontal area of the vehicle was $A_{fr} = 2.4\,\mathrm{m^2}$; the radius of the wheel was $r = 0.26\,\text{m}$; the maximum inclination angle was $\theta^{max} = 60\,\text{degrees}$; and the rolling coefficients were $C_0 = 0.015$ and $C_1 = 0$.

The t_r was in the region of $t_r \approx 100\,\text{s}$. Based on the value of t_r and the values presented above (summarized in Table 1), use of Equation (2) gives an estimated value of t_s of $100\,\text{ms}$, while the calculated value of t_a was $212\,\text{ms}$, taking into account the restriction presented in Equation (1).

Figures 21 and 22 show the results obtained for the RTSDS platform used to execute the prototype described above, together with the non-real-time results obtained using MATLAB/Simulink. Figure 21 presents the results for a programmed sequence of perturbations, with DAM-1 being programmed to begin with $\tau_m(t) = 500\,\text{Nm}$ ($V_e(t) = 0.25$) and, then, after $2000\,\text{Nm}$ ($V_e(t) = 1$) with $\theta(t) = 0\,\text{degree}$ ($V_o(t) = 0$). Figure 22 presents the results for a programmed sequence of perturbations, with DAM-2 being programmed to begin with $\theta(t) = 0\,\text{degree}$ ($V_o(t) = 0$) and, then, after $30\,\text{degree}$ ($V_o(t) = 0.5$) with $\tau_m(t) = 2000\,\text{Nm}$ ($V_e(t) = 1$). In all cases with Dynamic System 2 (Longitudinal Vehicle), it can be seen that the RTSDS platform provided satisfactory results, with errors that were very small in relation to the reference method.

Figure 23 presents the results obtained for an embedded PID controller [27] in an ATmega 2560 microcontroller (utilizing the Arduino Mega 2560 kit) external to the RTSDS platform. The system was controlled to begin with a reference value (set point) of $55\,\text{m/s}$ and, then, after $100\,\text{m/s}$.

Figure 21. Results obtained for the RTSDS platform applied to the dynamic system characterized by Equation (12), with $\theta(t) = 0\,\text{degree}$ and $\tau_m(t) = 500\,\text{Nm}$ initially and, then, $\tau_m(t) = 2000\,\text{Nm}$ after 30 s.

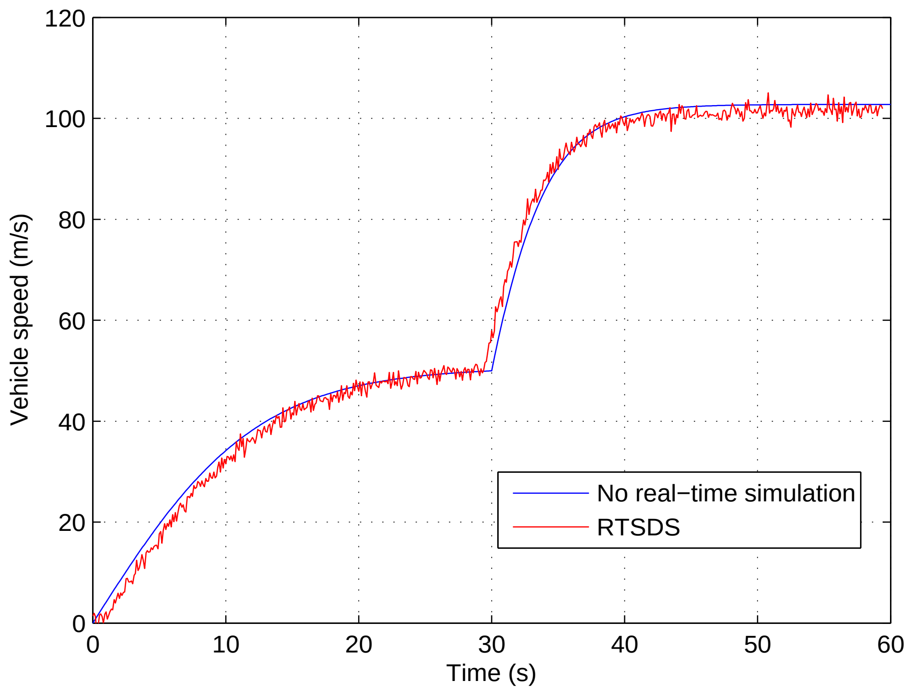


Figure 22. Results obtained for the RTSDS platform applied to the dynamic system characterized by Equation (12), with $\tau_m(t) = 2000\,\text{Nm}$ and $\theta(t) = 0\,\text{degree}$ initially and, then, $\theta(t) = 30\,\text{degree}$ after 30 s.

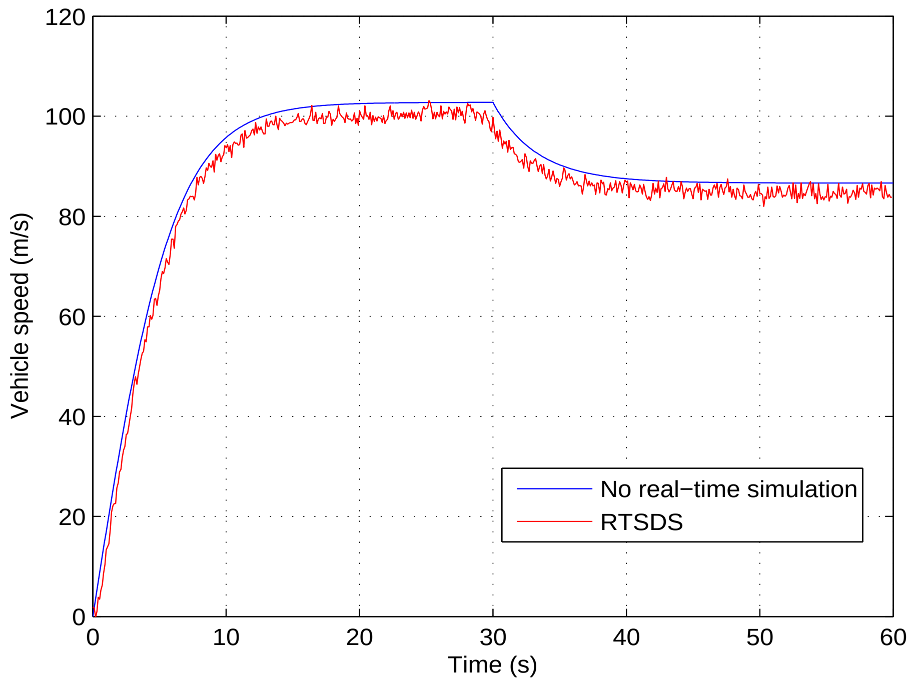
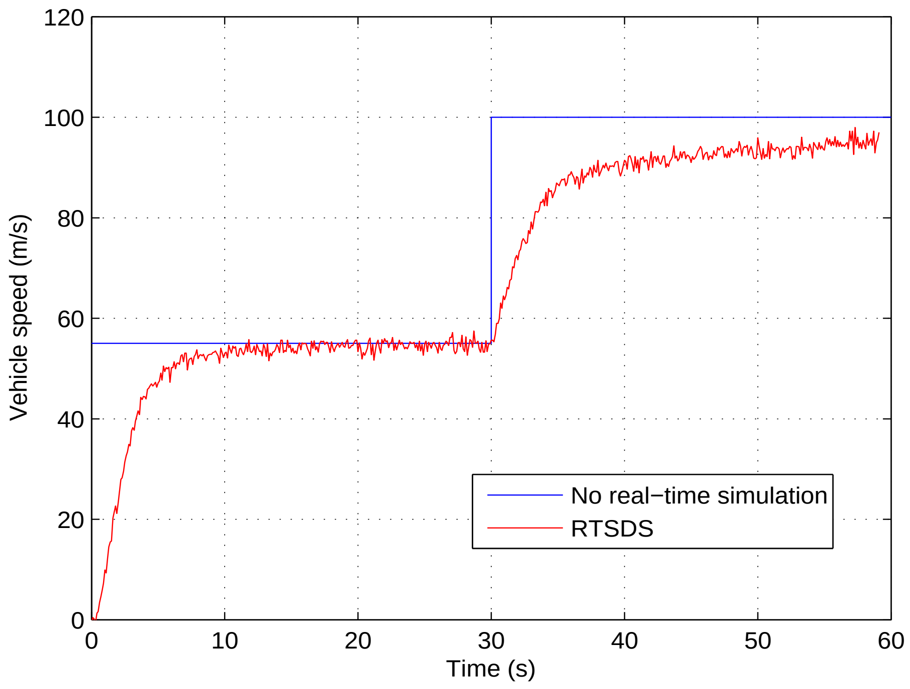


Figure 23. Results obtained for the RTSDS platform utilizing an external PID controller ($k_p = 3, k_i = 0.2$ e $k_d = 0.002$) onboard an ATmega 2560 microcontroller.



6. Conclusions

This work presents a proposal for an embedded solution for the real-time simulation of dynamic systems, called the RTSDS. This platform is composed of a central hardware module (the MCM) and a set of ancillary hardware modules (the DAMs and SGMs), so that the platform is modular and distributed. Experimental trials were performed using a prototype developed using an Intel DN2800MT Mini-ITX kit as the MCM, Arduino Mega 2560 kits as the DAMs and a PSOC CY8CKIT-001 kit as the SGM. The results obtained showed that the RTSDS platform provided good performance, with errors that were very small, compared to simulations that were not performed in real time. The RTSDS could be used to optimize and synchronize embedded control algorithms used in industry and also as a learning tool to support graduate and postgraduate teaching.

Author Contributions

All the authors have contributed in various degrees to ensure the quality of this work. Marcelo A. C. Fernandes conceived the idea and experiments; Isaac D. T. de Souza, Sérgio N. Silva and Rafael M. Teles designed and performed the experiments; Marcelo A. C. Fernandes, Isaac D. T. de Souza, Sérgio N. Silva and Rafael M. Teles analyzed the data; Marcelo A. C. Fernandes wrote the paper.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Dixit, V.; Patil, M.; Chandorkar, M. Real time simulation of power electronic systems on multi-core processors. In Proceedings of the International Conference on Power Electronics and Drive Systems, 2009 (PEDS 2009), Taipei, Taiwan, 2–5 November 2009; pp. 1524–1529.
2. Boukerche, A.; Lu, K. A novel approach to real-time RTI based distributed simulation system. In Proceedings of the 38th Annual Simulation Symposium, San Diego, CA, USA, 4–6 April 2005; pp. 267–274.
3. You, T.; Zhu, Y.A.; Zhang, D.P. Applied Research of Delaminated Real-Time Network Framework Based on RTX in Simulation. In Proceedings of the Second International Conference on Information and Computing Science, 2009 (ICIC '09), Manchester, UK, 21–22 May 2009; Volume 1, pp. 389–392.
4. Vityaz, O.; Zimmermann, G. Real-time simulation using graceful degradation of accuracy. *Energy Build.* **2005**, *37*, 795–806.
5. Anagnostopoulos, D. A methodological approach for model validation in faster than real-time simulation. *Simul. Model. Pract. Theory* **2002**, *10*, 121–139.
6. Champagne, R.; Dessaint, L.A.; Fortin-Blanchette, H. Real-time simulation of electric drives. *Math. Comput. Simul.* **2003**, *63*, 173–181.
7. Ćosić, K.; Kopriva, I.; Šikić, T. A methodology for digital real time simulation of dynamic systems using modern DSPs. *Simul. Pract. Theory* **1997**, *5*, 137–151.
8. Matar, M.; Iravani, R. FPGA implementation of the power electronic converter model for real-time simulation of electromagnetic transients. *IEEE Trans. Power Deliv.* **2010**, *25*, 852–860.

9. Monga, M.; Karkee, M.; Tondehal, L.K.; Steward, B.; Kelkar, A.; Zambreno, J. Real-Time simulation of dynamic vehicle models using a high-performance reconfigurable platform. In Proceedings of the 2012 International Conference on Computational Science (ICCS), Omaha, NE, USA, 4–6 June 2012.
10. Menghal, P.M.; Laxmi, A. Real time simulation: A novel approach in engineering education. In Proceedings of the 2011 3rd International Conference on Electronics Computer Technology (ICECT), Kanyakumari, India, 8–10 April 2011; Volume 1, pp. 215–219.
11. Dufour, C.; Andrade, C.; Bélanger, J. Real-Time simulation technologies in education: A link to modern engineering methods and practices. In Proceedings of the 2010 11th International Conference on Engineering and Technology Education (INTERTECH), Baía, Brazil, 7–10 March 2010; pp. 1–5.
12. Bélanger, J.; Venne, P.; Paquin, J.N. *The What, Where and Why of Real-Time Simulation*; Planet-RT; Opal-RT Technologies Inc.: Montreal, QC, Canada, 2010.
13. Venugopal, R.; Wang, W.; Belanger, J. Advances in real-time simulation for power distribution systems. In Proceedings of the 2011 International Conference on Energy, Automation, and Signal (ICEAS), Bhubaneswar, India, 28–30 December 2011; pp. 1–6.
14. Fathy, H.K.; Filipi, Z.S.; Hagena, J.; Stein, J.L. Review of hardware-in-the-loop simulation and its prospects in the automotive area. *Proc. SPIE* **2006**, *6228*, 62280E:1–62280E:20.
15. Kim, W.; Lee, B.; Kim, K.; Yang, T.; Kim, S. A real-time HWIL simulation control system architecture for implementing evaluation environment of complex embedded systems. In Proceedings of the 2011 13th International Conference on Advanced Communication Technology (ICACT), Phoenix Park, Korea, 13–16 February 2011; pp. 254–259.
16. Li, H.; Steurer, M.; Shi, K.L.; Woodruff, S.; Zhang, D. Development of a unified design, test, and research platform for wind energy systems based on hardware-in-the-loop real-time simulation. *IEEE Trans. Ind. Electron.* **2006**, *53*, 1144–1151.
17. Paquin, J.N.; Li, W.; Belanger, J.; Schoen, L.; Peres, I.; Olariu, C.; Kohmann, H. A modern and open real-time digital simulator of All-Electric Ships with a multi-platform co-simulation approach. In Proceedings of the IEEE Electric Ship Technologies Symposium, 2009 (ESTS 2009), Baltimore, MD, USA, 20–22 April 2009; pp. 28–35.
18. HYPERSIM Power System Real-Time Digital Simulator, OPAL-RT. Available online: http://www.opal-rt.com (accessed on 13 October 2014).
19. Paré, D.; Turmel, G.; Soumagne, J.C.; Do, V.Q.; Casoria, S.; Bissonnette, M.; Marcoux, B.; McNabb, D. Validation Tests of the Hypersim Digital Real Time Simulator with a Large AC-DC Network. Available online: http://ipstconf.org/papers/Proc_IPST2003/03IPST13-05.pdf (accessed on 14 October 2014).
20. dSPACE Systems, dSPACE. Avaliable online: http://www.dspace.com/en/inc/home.cfm (accessed on 13 October 2014).
21. Arduino-Arduino Mega 2560. Avaliable online: http://arduino.cc/en/Main/ArduinoBoardMega2560 (accessed on 13 October 2014).
22. Atmle-AVR-ATmega 328p. Avaliable online: http://www.atmel.com/devices/ATMEGA328P.aspx (accessed on 13 October 2014).

23. Arduino-Arduino Uno. Avaliable online: http://arduino.cc/en/Main/ArduinoBoardUno (accessed on 13 October 2014).
24. Cypress-PSoC 3: CY8C38 Family Data Sheet. Avaliable online: http://www.cypress.com/?rID=35178 (accessed on 13 October 2014).
25. Cypress-CY8CKIT-001 PSoC Development Kit. Avaliable online: http://www.cypress.com/?rID=37464 (accessed on 13 October 2014).
26. Popovici, K.; Mosterman, P. *Real-Time Simulation Technologies: Principles, Methodologies, and Applications*; Computational Analysis, Synthesis, and Design of Dynamic Systems; CRC Press: Boca Raton, FL, USA, 2012.
27. Ogata, K. *Modern Control Engineering*, 4th ed.; Prentice Hall PTR: Upper Saddle River, NJ, USA, 2001.
28. Trzynadlowski, A. *Introduction to Modern Power Electronics*; Wiley: Hoboken, NJ, USA, 2010.
29. Garrett, P. *High Performance Instrumentation and Automation*; Taylor & Francis: London, UK, 2010.
30. RTSDS Site, RTSDS. Avaliable online: http://www.dca.ufrn.br/~isaacdiego/RTSDS/index.html (accessed on 13 October 2014).
31. Intel-Intel Desktop Board DN2800MT. Avaliable online: http://www.intel.com/content/www/us/en/motherboards/desktop-motherboards/desktop-board-dn2800mt.html (accessed on 13 October 2014).
32. Butcher, J. *Numerical Methods for Ordinary Differential Equations*; Wiley: Hoboken, NJ, USA, 2008.
33. Griffiths, D.; Higham, D. *Numerical Methods for Ordinary Differential Equations: Initial Value Problems*; Springer Undergraduate Mathematics Series; Springer: Berlin, Germany, 2010.
34. Ben-Ari, M. *Principles of Concurrent and Distributed Programming*; Addison-Wesley: Upper Saddle River, NJ, USA, 2006.
35. Ubuntu 13.10. Avaliable online: http://www.ubuntu.com (accessed on13 October 2014).
36. Atmel-AVR-ATmega 2560. Avaliable online: http://www.atmel.com/devices/atmega2560.aspx (accessed on 13 October 2014).
37. Pacejka, H.B. *Tire and Vehicle Dynamics*; Society of Automotive Engineers and Butterworth-Heinemann: Oxford, UK, 2002.
38. Matlab/Simulink. Avaliable online: http://www.mathworks.com (accessed on 13 October 2014).
39. Quanser-Coupled Water Tanks. Avaliable online: http://www.quanser.com/Products/coupled_tanks (accessed on 13 October 2014).
40. Trovao, J.; Pereirinha, P.; Jorge, H. Simulation model and road tests comparative results of a small urban electric vehicle. In Proceedings of the 35th Annual Conference of IEEE Industrial Electronics, 2009 (IECON '09), Porto, Portugal, 3–5 November 2009; pp. 836–841.
41. Elbakush, E.; Sharaf, A.; Altas, I. An efficient tri-loop controller for photovoltaic powered four-wheel electric vehicle. In Proceedings of the 4th International Conference on Innovations in Information Technology, 2007 (IIT '07), Dubai, UAE, 18–20 November 2007; pp. 421–425.