The One-Shot Task Model for Robust Real-Time Embedded Control Systems

Article *in* IEEE Transactions on Industrial Informatics · September 2008

DOI: 10.1109/TII.2008.2002702 · Source: IEEE Xplore

Camilo Lozoya
Tecnológico de Monterrey
40 PUBLICATIONS **207** CITATIONS
SEE PROFILE

Manel Velasco
Universitat Politècnica de Catalunya
92 PUBLICATIONS **977** CITATIONS
SEE PROFILE

Pau Martí
Universitat Politècnica de Catalunya
123 PUBLICATIONS **1,572** CITATIONS
SEE PROFILE

The One-Shot Task Model for Robust Real-Time Embedded Control Systems

Camilo Lozoya, Manel Velasco, and Pau Martí, *Member, IEEE*

***Abstract*—Embedded control systems are often implemented in small microprocessors enabled with real-time technology. In this context, control laws are often designed according to discrete-time control systems theory and implemented as hard real-time periodic tasks. Standard discrete-time control theory mandates to periodically sample (input) and actuate (output). Depending on how input/output (I/O) operations are performed within the hard real-time periodic task, different control task models can be distinguished. However, existing task models present important drawbacks. They generate task executions prone to violate the periodic control demands, a problem known as sampling and latency jitter, or they impose synchronized I/O operations at each task job execution that produce a constant but artificially long I/O latency.**

In this paper, the one-shot task model for implementing control systems in embedded multitasking hard real-time platforms is presented. The novel control task model is built upon control theoretical results that indicate that standard control laws can be implemented considering only periodic actuation. Taking advantage of this property, the one-shot task model is shown to remove endemic problems for real-time control systems such as sampling and latency jitters. In addition, it can minimize the harmful effects that long I/O latencies have on control performance. Extensive simulations and real experiments show the feasibility and effectiveness of the novel task model, compared to previous real-time and/or control-based solutions.

***Index Terms*—Aperiodic observer, periodic task model, real-time embedded control systems, sampling and latency jitter.**

I. Introduction

The use of computer controlled systems has increased dramatically in our daily life. Processors and micro-controllers are embedded in most of the devices we use every day, such as mobile phones, cars, dishwashers, etc. Due to cost constraints, many of these devices that run control applications are designed under space, weight, and energy constraints, i.e., with limited resources. As a consequence, these applications typically run on small processing units requiring an efficient use of the available resources [8].

For control applications, recent research providing solutions meeting this trend covers two different problems. The first is related to the allocation of resources to control applications in order to maximize control performance. The second type of problems refers to effective mechanisms such as novel computational models for implementing control algorithms using real-time technology. This paper focuses on the latter.

For embedded control systems implemented in small microprocessors enabled with real-time technology, control algorithms are often implemented as hard real-time periodic tasks [4]. Using a periodic task, consecutive invocations (also called jobs) of a task are released periodically. Therefore, the periodic task abstraction already provides the periodicity required for implementing control algorithms. However, since standard control theory mandates to periodically sample and actuate [3], care must be taken with the I/O operations. Depending on how I/O operations are performed within the hard real-time periodic task, different control task models can be distinguished.

In this paper, it is shown that existing models for implementing control algorithms using periodic tasks present important drawbacks. They generate job executions prone to violate the periodic control demands of the I/O operations, problem known as sampling and latency jitter [4], or, to avoid the jitter problem, they impose synchronized I/O operations at each job execution (e.g., [14]) that produce a constant but artificially long I/O latency within each control loop. In both cases, the effect is that control systems performance degrades.

To overcome these limitations, in this paper the one-shot task model is presented. The novel control task model is built upon control theoretical results that indicate that standard control laws can be implemented considering only periodic actuation. That is, the periodic sampling requirement can be relaxed. Taking advantage of this property, the one-shot task model permits to

1) remove endemic problems for real-time control systems such as sampling and latency jitters; and
2) minimize the harmful effects that artificially imposed longer I/O latencies have on control performance.

To corroborate its correctness and effectiveness, simulations and real experiments have been carried out. They show the benefits that can be obtained by the application of the one-shot task model in terms of operation and performance, compared to existing real-time and/or control-based methods and models currently used a) for implementation of control algorithms using real-time technology or b) for minimizing the degrading effects that jitters have on control performance.

The rest of this paper is structured as follows. Section II introduces the preliminaries and discusses the state of the art to motivate the paper. Section III presents the theoretical approach and Section IV presents the one-shot task model. Sections V–VII provide the results for the simulations and the control experiment. Finally, Section VIII concludes the paper.

Manuscript received November 5, 2007; revised March 13, 2008. First published August 15, 2008; current version published September 10, 2008. This work was supported by part by NoE ARTIST2 IST-FP6-2004-004527, NoE ARTISTDESIGN IST-FP7-2008-214373, and in part by the Spanish Ministerio de Educación y Ciencia under Project ref. CICYT DPI2007-61527. Paper no. TII-07-10-0175.R1.

The authors are with the Automatic Control Department, Technical University of Catalonia, 08028 Barcelona, Spain (e-mail:camilo.lozoya@upc.edu; manel.velasco@upc.edu; pau.marti@upc.edu).

Digital Object Identifier 10.1109/TII.2008.2002702

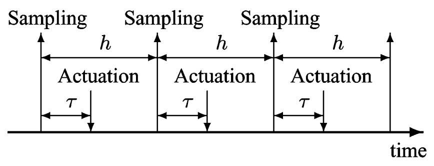

Fig. 1. Control timing demands.

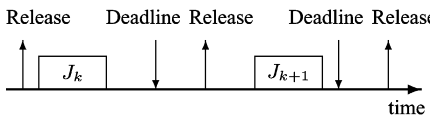

Fig. 2. Hard real-time periodic task.

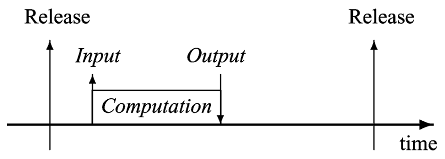

Fig. 3. Naif task model.

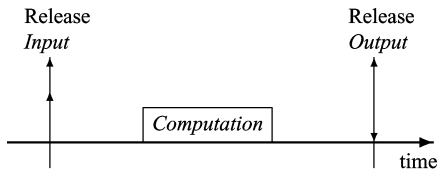

Fig. 4. One-sample task model.

II. Preliminaries

The common approach to computer controlled systems design has two steps. The first step is to obtain a discrete-time model of the plant. The second step is to design a discrete-time control law for the discrete-time plant model. The design approach mandates to periodically sample and actuate.

The key aspect of the design procedure is that the discrete-time model of the plant describes the behavior of the analog plant at the sampling instants. Moreover, the actuation instants are defined in terms of the sampling instants. Therefore, the time reference and synchronism is given by the sampling instants (as illustrated in Fig. 1). Once a sample is taken, the control signal is computed assuming that the next sample will occur after h time units (i.e., one sampling period), and assuming that the actuation will occur after τ time units (i.e., one time delay). Both assumptions refer to future *known* events: next sampling instant or subsequent actuation instant.

After the controller design stage, the control law is implemented by means of a control algorithm. Although real-time computing is about meeting timing constraints [7], it is not straightforward to meet the periodic control demands with available real-time technology. The hard real-time periodic task is the baseline computational abstraction for implementing control algorithms. In the periodic task, illustrated in Fig. 2, consecutive releases times mark the task period, and jobs execute within each release time and relative deadline. The relative deadline can be assumed to be less or equal to the period. To exemplify the limitations of current practice for real-time implementation of control algorithm, two alternative models are analyzed.

A. Existing Control Task Models

The first model identified in [4] as the common practice implementation of control loops in real-time control systems is refereed as "naif" task model in the rest of this paper. The naif task model assumes control algorithms implemented as hard real-time periodic tasks, with task period equal to the sampling period. Sampling (input) and actuation (output) operations are specified to occur at the beginning and at the end of each job execution, as illustrated in Fig. 3.

The deadline specification is a key aspect in the naif task model. For one task executing in isolation, the timing of the control task execution corresponds to the expected timing (Fig. 1) if the deadline is set equal to the time delay. However, in a multitasking system, this tight specification of the deadline impairs task set schedulability in the general case [20]. Relaxing this specification by setting the deadline greater than the time delay introduces sampling and latency jitters in control job executions. If jitters occur, the next sampling and/or actuation will be performed at times different than the expected ones. The synchronism given by the sampling instants is lost. Therefore, the introduced time uncertainty violates the mandated periodicity, and control performance degradation occurs [28].

The second model for meeting control timing demands with real-time technology is found in the Liu and Layland [18] approach and is also formally given as a computational model for control algorithms in [14]. This model also assumes control algorithms implemented as hard real-time periodic tasks, with task period equals to the sampling period and deadline equals to period. But sampling and actuation are performed by hardware interrupts at the release time and deadline respectively, as illustrated in Fig. 4. Therefore, a constant I/O latency of one sampling period occurs at each job execution. We refer to this type of task model as "one-sample" task model. By using this model, jitters are removed but two main drawbacks are introduced. First, the model imposes an artificial long time delay in the closed-loop system of one-sample, which introduces an unnecessary although predictable control performance degradation. Second, task set schedulability is reduced compared to the naif task model due to the I/O interrupt handlers execution.

B. State of the Art

Apart from the models discussed in the previous section, other solutions to an effective implementation of real-time periodic control tasks can be found in the literature. These can be roughly divided into control-based solutions or real-time-based solutions.

The problem of dealing with scheduling-introduced jitters has been treated from a more pure control perspective. However, the approaches that have been presented (e.g., see [1] for an initial solution to the sampling jitter problem, or see [17] for a partial solution to the latency jitter problem) demand complex control design procedures or pose theoretical assumptions that still need to be verified, tested, or integrated within existing scheduling theory. A partial control solution fully integrated within a real-time framework can be found in [21]. By compensating for jitters through the adjustment of controller parameters, it is shown that both control performance and task set schedulability can be improved. This solution is refereed as "switching" task model in the rest of this paper. The main drawback of this approach is the overhead of switching the controller gains, as well as possible chattering problems that may occur.

Alternative solutions to minimize the likelihood of jitters can be found in the real-time systems literature. Two main approaches can be distinguished. The first approach is based on limiting the time interval where control jobs can execute. This is basically achieved by assigning shorter relative deadlines, as initially proposed in [6]. This approach is also taken in [12], where a task model based on the Giotto model [14] is presented. The second approach is based on splitting the control task into subtasks and schedule them separately. In [11] it is proposed to split control tasks into two parts, one for calculating the control signal and the other for updating the state. Similarly, in [5], a novel task model is presented where sampling, control computation and actuation are split into three subtasks. In both of them tasks are scheduled in such a way that jitters or the delay variance is reduced, respectively. Solution [5] is refereed as "split" task model in the rest of this paper. These real-time-based solutions suffer from three problems. First, the occurrence of jitters is not completely eliminated. Second, artificial I/O latencies are still enforced. Also, when latencies are shortened by specifying shorter relative deadlines, the third problem arises. System schedulability is reduced because jobs are forced to execute within shorter time intervals.

In this paper we present the one-shot task model for implementing control algorithms as hard real-time periodic tasks in such a way that the problems outlined before are solved. The analysis consolidates and extends previous work [23], which was earlier suggested in [2].

III. Theoretical Approach

The key property that permits to develop the one-shot task model states that control algorithms can be implemented considering only periodic actuation [23]. Fig. 5 graphically illustrates the theoretical approach. In Sections III-A–C, this figure will be described in detail. The time reference for the three sub-figures is the same.

A. Timing Analysis of Standard Controllers

Consider the mathematical description of a system given by the state-space model of a linear time-invariant discrete-time system with sampling period h [3]

$$\begin{aligned} x_{k+1} &= \Phi(h)x_k + \Gamma(h)u_k \\ y_k &= Cx_k \end{aligned} \tag{1}$$

where x_k is the plant state, u_k and y_k are the inputs and outputs of the plant, matrix $C \in \mathbb{R}^{p\times n}$ is the output matrix, and matrices $\Phi(t)$ and $\Gamma(t)$ are obtained using

$$\Phi(t) = e^{At}, \quad \Gamma(t) = \int_0^t e^{As}Bds \tag{2}$$

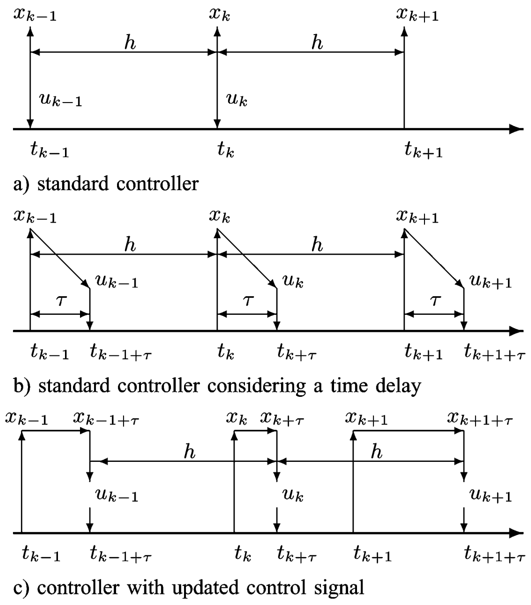

Fig. 5. Timing analysis of controllers.

with $t = h$, where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ are the system and input matrices of the continuous-time form

$$\begin{aligned} \frac{dx(t)}{dt} &= Ax(t) + Bu(t) \\ y(t) &= Cx(t). \end{aligned} \tag{3}$$

If the inputs u_k are given by a control algorithm implemented as a periodic task, k identifies the job within a sequence of jobs. For standard closed-loop operation of (1), the control signal u_k is given by

$$u_k = Lx_k \text{ with } L \in \mathbb{R}^{1\times n} \tag{4}$$

where L is the state feedback gain obtained using standard control design methods from matrices $\Phi(h)$ and $\Gamma(h)$.

The application of (4) to the plant mandates computing the control signal with *zero* time, as illustrated in Fig. 5(a): the kth control signal u_k is applied to the plant at the same time instant t_k that the kth sample x_k is taken,[1] sample that is required to compute the control signal. This is physically impossible; computing the control signal takes time.

This theoretical model (1) can be augmented to cope with a time delay modeling an I/O latency that appears due to the computation of the control algorithm. The standard model that incorporates a time delay τ, with $\tau \leq h$, is [3]

$$x_{k+1} = \Phi(h)x_k + \Phi(h-\tau)\Gamma(\tau)u_{k-1} + \Gamma(h-\tau)u_k. \tag{5}$$

Note that matrices $\Phi(\cdot)$ and $\Gamma(\cdot)$ in (5) slightly differs from conventional notation. The purpose of this notation is to explicitly indicate dependencies on h and τ.

Model (5) has been taken as the underlying control model for constructing real-time task models for control algorithms.

[1]Sample is used to refer to the full state vector, regardless of whether it has been sampled or observed.

As illustrated in Fig. 5(b), it is based on two synchronization points, the sampling and actuation instants, on a time reference given by the sampling instants. At time t_k the kth sample (x_k) is taken and the computation of the control signal (u_k) can be started. At time $t_{k+\tau}$ the calculated control signal is applied to the plant. The sampling period h is defined from t_k to t_{k+1}, and the time delay τ from t_k to $t_{k+\tau}$. Note that the standard subscript notation in the standard state-space model with time delay (5) may be misleading. In the model, the k-subscript identifies the job within a sequence of jobs, not the time events occur. For example, u_k is the control signal of the k-job, although it is applied at time $t_{k+\tau}$.

For closed-loop operation of (5), the control signal will be

$$u_k = [L_1 \quad L_2] \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} = L_1 x_k + L_2 u_{k-1}$$
$$\text{with } L_1 \in \mathbb{R}^{1\times n}, \quad L_2 \in \mathbb{R}^{1\times 1} \tag{6}$$

where $[L_1 \quad L_2]$ is the state feedback gain obtained using standard design procedures. With this configuration, the control signal u_k is held from $t_{k+\tau}$ to $t_{k+1+\tau}$.

Remark 1: The standard controller designed to cope with a time delay (6) involves computing the control signal (u_k) to be applied to the plant at time $t_{k+\tau}$ using a sample (x_k) taken at t_k, τ time units before [represented by diagonal arrows in Fig. 5(b)].

B. Controller With Updated Control Signal

Rather than applying standard controllers (4) or (6), this paper, as illustrated in Fig. 5(c), proposes to apply a controller with updated control signal to account for the delay. Instead of computing the control signal with an state vector x_k that becomes outdated at the time the control signal is applied, we propose to use an updated (estimated) state vector. Thus, the control signal u_k is computed it terms of the estimated state at time $t_{k+\tau}$, labelled $x_{k+\tau}$. Moreover, $x_{k+\tau}$ can be computed using a sample x_k taken at any time $t_k \in (t_{k-1+\tau}\ t_{k+\tau})$.

Therefore, the controller will first estimate the state

$$x_{k+\tau} = \Phi(\tau_k)x_k + \Gamma(\tau_k)u_{k-1} \tag{7}$$

where matrices $\Phi(\cdot)$ and $\Gamma(\cdot)$ are given by (2) for $t = \tau_k$, with

$$\tau_k = t_{k+\tau} - t_k. \tag{8}$$

Second, the controller will compute the control signal

$$u_k = Lx_{k+\tau} \text{ with } L \in \mathbb{R}^{1\times n} \tag{9}$$

where L is the original controller gain (4) obtained using standard control design methods from matrices $\Phi(h)$ and $\Gamma(h)$.

Remark 2: A controller using (7)–(9) relies on a the time reference given by the actuation instants. The time elapsed between consecutive actuation instants $t_{k+\tau}$ and $t_{k+1+\tau}$ is the sampling period h. Moreover, no delay is present in the closed loop model. Also, samples are not required to be periodic because τ_k in (8) can vary at each closed-loop operation.

Algorithm 1: Controller with updated control signal

```
1 begin
2     $x_k$ := read_input()
3     $t_k$ := get_time()
4     $t_{k+\tau} := t_{k+\tau} + h$
5     $\tau_k := t_{k+\tau} - t_k$
6     $x_{k+\tau} := \Phi(\tau_k)x_k + \Gamma(\tau_k)u_{k-1}$
7     $u_k := Lx_{k+\tau}$
8     $u_{k-1} := u_k$
9 end
```

Fig. 6. Controller pseudo-code.

In previous work [23] it is deduced the equivalence relations between the state space models in closed loop form when using standard controllers [(4) or (6)], or when applying the controller using updated control signals (7)–(9), which are summarized here.

- For irregular sampling with $t_k \in (t_{k-1+\tau}\ t_{k+\tau})$, standard controllers can not be applied. However, the proposed controller given by (7)–(9) can be applied.
- The control law in (9) has the same dimension than the control law in (4). This keeps the controller design problem simpler than the case of the standard model with time delay, where the controller gain in (6) has to also consider the previous control signal.
- The application of controller (6) is more general than (7)–(9). That is, a more complete set of dynamics can be achieved using standard controllers.

C. Design and Implementation Considerations

The application of the proposed controller (7)–(9) requires executing a control algorithm that slightly differs from conventional controllers. Using a simple academic example, in this section we detail the design steps and the control algorithm pseudo-code.

Given the double integrator described by

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \tag{10}$$

a standard discrete-time deadbeat state feedback controller

$$L = [-100 \quad -15] \tag{11}$$

has been designed for a sampling period $h = 0.1$ s. As specified in (7), the state vector estimation requires to apply $\Phi(t)$ and $\Gamma(t)$ for each $t = \tau_k$. Therefore, these matrices have to be precomputed in terms of t as in

$$\Phi(t) = \begin{bmatrix} 0 & t \\ 0 & 0 \end{bmatrix} \text{ and } \Gamma(t) = \begin{bmatrix} \frac{t^2}{2} \\ t \end{bmatrix}. \tag{12}$$

The pseudo-code of the controller with updated control signal is given in Fig. 6. The controller first samples the plant and gets the current time, thus obtaining x_k and t_k (lines 2 and 3). Afterward, lines 4 and 5 are used to compute the time that will elapse from t_k to the actuation instant, thus implementing (8). Recall that actuation instants are given by h (remark 2). The initial value of $t_{k+\tau}$ is zero. Line 6 implements the state prediction at the actuation instant (7). Therefore, it has to substi-

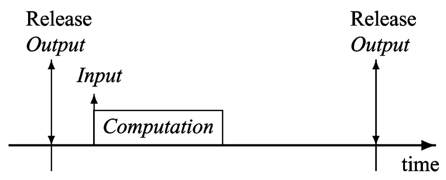

Fig. 7. One-shot task model.

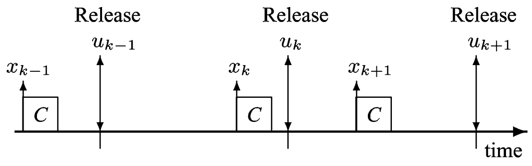

Fig. 8. One-shot task execution.

tute in matrices (12) the current value of τ_k computed in line 5, and compute the estimated vector using the sampled state and the previous control signal. Then, in line 7 the control signal is computed using the gain $L(11)$ that has been obtained at the design stage. The last line of the pseudo-code is used to save the control signal value for the next execution.

The pseudo-code shown in Fig. 6 should be executed "periodically," meaning that it should meet the timing imposed by (7)–(9) and illustrated in Fig. 5(c). It is important to stress that in the pseudo-code the control signal is not directly output to the plant because the real-time kernel will be in charge of enforcing its application at the actuation instants.

Looking at computational overhead, the state vector estimation (line 6) is the most significant modification compared to a standard control algorithm. However, it does not add significant overhead because it implies the same operations than standard observer-based control designs. Future work will cope with this analysis.

IV. One-Shot Task Model

Remark 2 provides the semantics to build a new task model for control tasks, the one-shot task model. The proposed controller (7)–(9) forces job executions to occur within two consecutive actuation instants, separated by fixed h time units. Therefore, each job release takes place at each actuation instant, that is $r_k = kh$. Also, each job deadline is given by the next actuation instant, that is $d_k = r_k + h$. With these timing constraints, the one-shot task model matches the standard Liu and Layland periodic task model with deadline equal to period, but with the following requisites.

- Sampling is performed at each job execution start time.
- Actuation has to be carried out by a synchronized output operation performed by the kernel at the release times.
- The control algorithm implements the pseudo-code of Fig. 6.

Fig. 7 illustrates the one-shot task model (for comparative purposes, see Figs. 3 and 4). Fig. 8 shows the execution of a one-shot task in the scenario illustrated in Fig. 5(c) (where C stands for computation).

A. Task Model Analysis and Properties

The one-shot task model has several appealing properties for controllers. Although some of them have been already stated, all of them are summarized next.

1) Property 1: Compatible with standard scheduling: the one-shot task model does not demand any specific timing constraints other than the ones of the standard hard real-time periodic task model. Therefore, it can be applied within existing scheduling algorithms for periodic tasks.

2) Property 2: Improves schedulability: the only synchronized operation required by each one-shot task is the actuation. Therefore, the number of interrupts handlers for a real-time system executing multiple control loops is cut by half compared to those models using also synchronization operations for sampling. A simple consequence is that task set schedulability is improved (see [16] for an example of analysis of hardware interrupts in dynamic priority task systems).

3) Property 3: Absorbs scheduling jitters: the a priori known time reference for the task model is given by the actuation instants. Moreover, no delay is present in the model (remark 2). Therefore, latency jitter is not a concern. Sampling jitter problems also disappear because (7) absorbs the irregular sampling.

4) Property 4: Does not require switching controllers: compared to other solutions [21] where the gain is updated according to varying timing constraints, in the one-shot task model the gain is constant because constraints do not vary: sampling period is constant and no delay is present.

5) Property 5: Does not force long I/O latencies: the one-shot task model performs sampling at the beginning of each job execution, and actuation is performed at the next task release. Therefore, the I/O latency goes from each job start time to the next release. In the general case, this time interval will be less than one-sample, which is the I/O latency forced by previous approaches [18].

As a final observation, note that Property 3 provides isolation between control tasks, like in the approach presented by [12]. That is, although task instances may be subject to jitters, their operation is not affected by them. Therefore, the performance that will be achieved by tasks using the one-shot task model is similar to the performance that tasks will achieve if executing in isolated processors.

B. Responsiveness to Perturbations

The last property has an immediate benefit in terms of responsiveness of controllers based on the one-shot task model. Short I/O latencies permit controllers to be more efficient when affected by perturbations.

Let us compare a controller that is implemented with a task based on the one-sample model (e.g., [18]) to one with a task based on the one-shot model. Both tasks have the same timing constraints (same period and deadline). An arbitrary schedule gives the sequence of two jobs executions (k and $k+1$ job execution), as shown in Fig. 9. The top part shows the one-sample task jobs and the bottom part shows the one-shot task jobs. Each

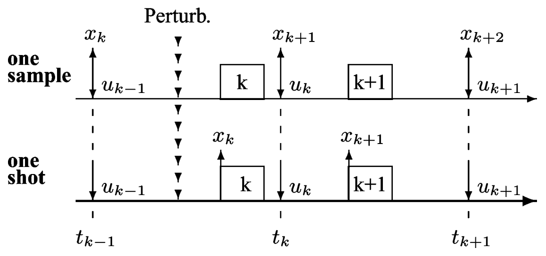

Fig. 9. One-shot task versus one-sample task.

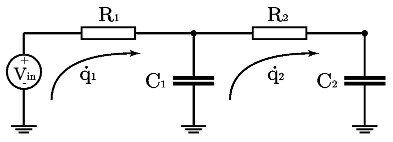

Fig. 10. Electronic circuit scheme.

i-job input and output operations are labeled by x_i and u_i, and illustrated with upside and downside arrows, respectively. Times t_k mark jobs release times.

In Fig. 9 we mark with a line of down arrows the arrival of a perturbation, which occurs before the k-job execution. It is interesting to note that for each k-job, all perturbations arriving at times $t \in (t_{k-1}, t_{k,s}]$ will be detected and started to be corrected by the k-job in the one-shot task but not in the one-sample task ($t_{k,s}$ denotes the k-job execution start time). In the one-sample task, these perturbations will be detected and started to be corrected by the $(k+1)$-job. That is, corrective operations will be send out one sampling period later in the one-sample model.

This benefits the control performance achievable by the one-shot task because it reacts faster to perturbations. If perturbations arrive at times $t \in (t_{k,s}, t_k)$, both tasks will provide the same responsiveness. As a consequence, scheduling policies that favor jobs executions near to their deadlines, e.g., [25]), making longer the interval $(t_{k-1}, t_{k,s})$ will improve control performance if using the one-shot task model for controllers.

V. Simulation Results: Comparison to Existing Solutions

In this section, a first set of simulations show the operation and performance of the one-shot task model compared to existing real-time and/or control-based methods and models currently used in real-time control systems. This evaluation includes the naif task model (already discussed in Section II) and the one-sample task model as example of the application of formal methods to real-time implementation of control loops [14]. In addition, the solution presented by [21], named “switching” task model, as example of control-based solution, and the model presented by [5], named “split” task model, as example of real-time-based solution, are also evaluated.

All simulation results reported in this Section and in Section VI, as well as the experimental results, are based on real-time control of voltage stabilizers.

TABLE I
Task Set Parameters (in ms)

	h	D	C
T1	12	12	6
T2	20	20	6

A. Plants and Control Objective

Voltage stabilizers are electronic circuits in the form of *RCRC* circuits, as illustrated in Fig. 10. For the simulation set-up as well as for the control experiment, the electronic components are $R_1 = R_2 = 330\ \mathrm{K}\Omega$ and $C_1 = C_2 = 100\ \eta\mathrm{F}$. Taking into account the components values, an state-space form is given by

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ -918.27 & -90.90 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 918.27 \end{bmatrix} u(t)$$
$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \quad (13)$$

where $x(t) = [x_1 \quad x_2]$ is the state vector, where x_1 corresponds to the voltage in C_2, and x_2 is $\dot{q}_2/C_2$. Each control low is designed to maintain the output voltage V_{out} of C_2 at the desired set-point, which follows a square wave with 0 V as a low level and 2.4 V as a high level.

B. Workload

For the analysis, let us consider the task set given in Table I, where h is the task period, D is the relative deadline and C is the worst case execution time. Task T_2 is a control task while task T_1 is a non-control real-time periodic task. Under earliest deadline first (EDF [18]) scheduling, it is easy to observe that control task T_2 suffers sampling and latency jitter.

Within this scenario, the control task T_2 will implement a standard pole placement control law to track the square wave set-point. The state feedback controller places the continuous closed-loop poles at $-103.93 + 87.1i$ and $-103.93 - 87.1i$. The corresponding discrete closed-loop poles locations depend on the control task period (i.e., sampling period), that, together with the task latency (i.e., time delay), will then determine the controller gain. Since the delay modeling the computation time has also been considered in some approaches, the third discrete-time closed-loop pole is set to 0 whenever required. Therefore, the specific timing used to design the controller is listed next, according to the five strategies under evaluation.

1) Naif task model: the sampling period and time delay for designing the control law are 20 ms and 6 ms.
2) One-sample task model: the sampling period and time delay for designing the control law are 20 ms and 20 ms. In addition, the control task will execute sampling and actuation at a constant I/O latency of one period, 20 ms.
3) One-shot task model: the sampling period and time delay for designing the control law are 20 ms and 0 ms. Recall remark 2 where it is noted that the sampling period is constant on the basis of equidistant actuation instants, and the time delay is zero since the control signal is computed using the updated state vector.

4) Switching controller gains: the control task will be applying one controller gain out of three different gains depending on the real sampling periods and latencies $\{(h,\tau)\} = \{(14,6),(22,12),(24,6)\}$ ms that can be derived from the EDF schedule assuming that tasks execute on their worst case execution time.
5) Split task model: The operation of the control algorithm is split into three sub-tasks, sampling, control computation and actuation, that are scheduled separately. For this case, the sampling period and time delay for designing the control law are 20 ms and 0 ms.

C. Detailed Performance Analysis

This section gives simulation details of the operation of the one-shot task model in terms of plant outputs and control signals compared to other strategies. The simulation involves evaluating the tracking on a set-point change from 0 V to 2.4 V, occurring at time 10 ms. For the analyzed control strategies, Fig. 11 shows the plant outputs (top) and the control signals (bottom) over 200 ms. The middle sub-figure shows the sequence of jobs of the control task T_2. During the simulation time, ten jobs execute, where high level line means job in execution, and middle level line means job preempted due to the execution of jobs of T_1.

The first strategy, named "reference" and designed only for comparative purposes, is a pure periodic controller, with period 20 ms and no delay, thus simulating an ideal execution in isolation. Therefore, the sequence of jobs shown in the middle sub-figure does not apply to this strategy. For the rest of strategies, the controller implemented in task T_2 uses the one-shot, the one-sample or the naif task model. Note that in this evaluation, the switching and the split task models are not assessed in order to simplify the analysis.

As it can be seen in the top sub-figure, the outputs of the reference controller and the controller using the one-shot task model are equal in terms of dynamics, but shifted in time. Taking into account that the set-point change occurs at time 10 ms, the second job of the reference controller executes at time 20 ms, and therefore it sees the new set-point and starts immediately correcting the tracking error. However, the second job of the controller using the one-shot task model sees the new set-point also at time 20 ms, but it will output the control signal at time 40 ms, at the enforced actuation instant. However, since the first tracking error is the same, both control signals are the same, but shifted in time, as illustrated in the bottom sub-figure. In this case the reference controller is more reactive to the set-point change than the one-shot due to the specific phasing between samples and the set-point change. But in other scenarios, the phasing can favor the one-shot controller. This is further analyzed in Section V-D.

The performance of the one-shot compared to the one-sample and naif task model is different in terms of plant output due to the difference in the control signals. The plant output of the one-sample gives different performance due to the introduced I/O latency of 20 ms. As it can be seen in the bottom sub-figure, the first correcting action, also applied at time 40 ms, is of different magnitude than the one-shot because it is based on an extended controller (6). In addition, subsequent control actions do

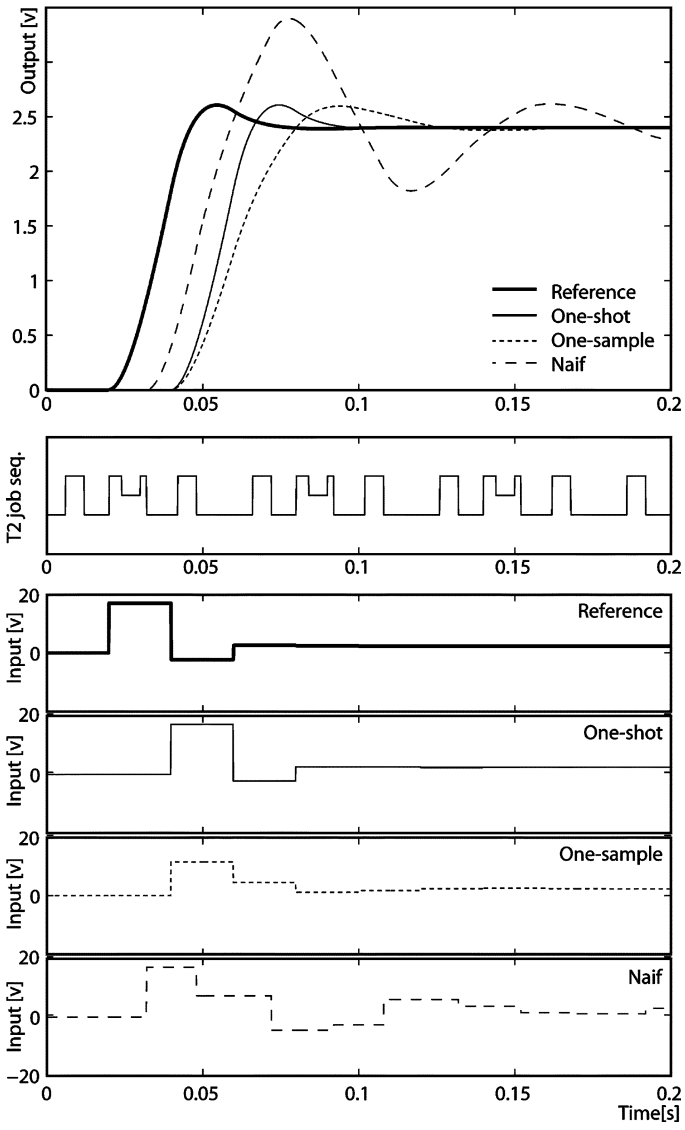

Fig. 11. Detailed view of the operation of several control strategies.

not incorporate the state estimation used in the one-shot. The naif task model gives the worst dynamics because job executions are affected by jitters, which have not been accounted for in the controller.

All these effects imply that the desired performance of the tracking achieved by the controller using the one-sample and naif task model is not met. In addition, for all the evaluated task models (see Section V-B), each set-point change produces an increment of cost if the tracking is evaluated using a standard quadratic cost function on the states and control inputs, as it is further analyzed next.

D. Performance Evaluation

Fig. 12 shows the control performance achieved by the five strategies. The y-axis shows the cumulative error measured in terms of the quadratic cost function of the plant state (voltage and electric current) and control signal

$$J = \int_0^{t_f} \left(x^T Q x + u^T R u\right) dt \tag{14}$$

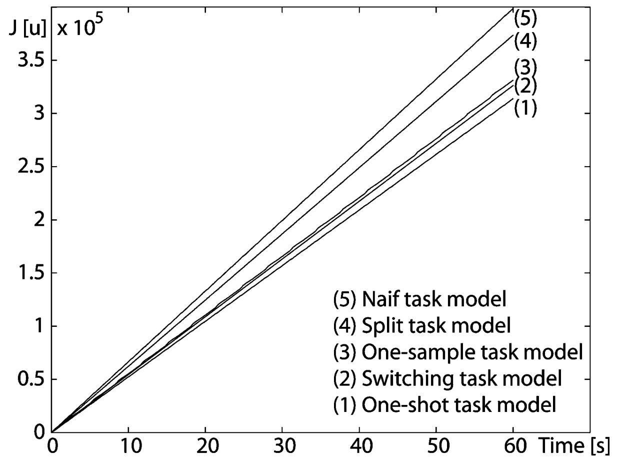

Fig. 12. Performance evaluation in front of existing solutions.

where

$$Q = \begin{bmatrix} 400 & 1 \\ 1 & 0 \end{bmatrix}, \quad R = 1$$

and t_f marks the final time of the evaluation period, 60 s. Since the focus of the controller is to ensure fast tracking of the output variable given the desired voltage output, the cost function heavily penalizes tracking errors. Therefore, the lower the curve, the better the performance. The electronic circuit has been perturbed using the described set-point changes occurring uniformly distributed over the control task period.

As it can be seen in Fig. 12, the one-shot task model achieves the best performance. As expected, the naif task model achieves the worst performance because the I/O operations are subject to jitters. The other approaches, that follow different strategies to overcome the jitter problem, lie in the middle. The exact ordering in terms of performance will vary depending on the number of tasks, plants under control, and scheduling policy. The split task model strategy, although reducing the jitter variance, can not completely remove jitters. The one-sample task model approach eliminates jitters by construction at the expenses of forcing a one-sample delay at each job execution, which introduces performance degradation in the control loop operation. The switching task model strategy, although applying different gains according to the run-time jitters, also introduces some degradation due to the switching. Finally, looking at the one-shot task model, since it eliminates jitters by construction without introducing a one sample delay, it is capable of achieving the best performance.

VI. Simulation Results: Application to Feedback-Scheduling Solutions

This section presents the application of the one-shot task model into recent feedback scheduling (FS) approaches. The goal of FS is to select optimal sampling periods for control tasks such that aggregated control performance in resource-constrained control systems is maximized. To do so, a large number of representative feedback scheduling approaches [9], [10], [13], [22], [26], [27] have been simulated. The objective of this study is twofold: first, it shows the degrading effects that jitters have on control performance, and second, it validates that the one-shot task model is able to absorb them, eliminating the introduced degradation. The key features of the feedback scheduling approaches used in the evaluation are as follows.

- Offline EDF and RM: in [26] and [27] optimal sampling periods are optimally assigned offline to the set of control tasks after considering the *a priori* relation between the control performance and sampling frequencies.
- Online FS: in [13], optimal sampling periods are assigned online considering also workload changes.
- Online instantaneous and finite horizon FS: in [22] and [10] optimal sampling periods are assigned online considering in addition the dynamics of the controlled plants (in terms of the state vector or of a prediction of each plant dynamics over a finite horizon, respectively).
- Online finite horizon cyclic scheduling (CS): this approach provides optimal cyclic sequences of jobs rather than sampling periods. In [9], offline optimal cyclic sequences are derived and updated at run-time according to the predicted error over a finite time horizon.

For the comparative analysis, also the "static approach" to real-time control systems has been also simulated. This approach is not a feedback scheduling approach because it does not include any performance optimization process. For the static approach, the sampling periods values are chosen arbitrarily offline, and EDF is used to dispatch the tasks modeled by the naif task model.

A. Workload and Simulation Details

The simulation process was conducted over a Matlab and Simulink platform, using the TrueTime kernel [15] to implement the real-time multitasking processor.

For each approach, three independent tasks control a set of three identical voltage stabilizers, as described by (13). For the simulation purposes, it is assumed that the three plants starts in a steady state (outputs in the equilibrium point), and during the execution period each task is affected by an external disturbance, represented by a pulse. Pulses are uniformly distributed during the simulation time and the pulse magnitude is randomly generated within a specified range. The control law applied in each approach is designed according to the guidelines given in each solution but with the same performance specifications described in Section V-B in order to keep the output voltage at constant 2.4 V. The sampling period range for all approaches goes from 20 ms to 40 ms. The time delay is 10 ms. The simulation time is 3 s.

Although it is interesting to study the relative performance between the FS approaches, the goal of the one-shot task model is to remove the effect of jitters. Therefore, the following scenarios have been simulated.

- Original approach: This simulation setup corresponds to the original solution. That is, one single processor concurrently executes the three control tasks according to the described feedback scheduling approaches using the task model specified in each approach.

TABLE II
EVALUATION OF FEEDBACK SCHEDULING APPROACHES

Approach	Original	Indep. Proc.	One-Shot
Static approach	109.05	105.82	105.43
Off-line EDF [26]	100.58	96.59	96.07
Off-line RM [27]	121.85	96.59	96.07
On-line FS [13]	99.92	98.74	98.06
On-line instantaneous FS [22]	90.63	64.41	63.26
On-line finite horizon FS [10]	100.61	86.99	83.81
On-line finite horizon CS [9]	82.46	82.46	82.46

- Independent processors: Since many of the evaluations of the previous approaches are subject to the degrading effects of jitters, a new simulation setup has been prepared. In this case, each control task executes on a separated processor, but according to the sampling periods or job sequences given in the first simulation setup. In this way, the results will show the performance of the feedback scheduling approaches in the absence of jitters.
- One-shot: This simulation setup is similar to the first one in the sense that only one processor is considered. The main difference is that the one-shot task model is applied instead of the model given in each particular solution.

B. Performance Evaluation

The performance of each approach on each simulation setup has been measured in terms of the cumulative error of the tracking in the three plants, using the same quadratic cost function specified in (14). Table II summarizes all simulation results. Rows refer to the feedback scheduling approaches under evaluation, and columns refer to the three simulation setups. Numbers are the accumulated error, that is, the smaller the number, the better the performance.

In the first column we can observe unexpected results. That is, the static approach should give the worst performance (the highest number) because the rest of approaches theoretically should perform better than the static because they perform some kind of optimization. However this is not the case. Jitters hide the true performance of the feedback scheduling approaches. However, if we look at the second column, since tasks do not suffer jitters because they execute in independent processors, the results are consistent with the theory. That is, the static approach is the worst, and the others perform better.

The third column shows the performance of all the approaches when tasks are executed using the one-shot task model. As it can be seen, the degrading effects that jitters have on performance are removed. Moreover, similar performance numbers to the case where tasks execute in isolation is achieved. This corroborates one of the main properties of the one-shot task model: it absorbs scheduling-introduced jitters.

VII. Control Experiment

Finally, a control experiment involving the implementation of the one-shot task model in a real-time kernel is presented. The experiment shows the feasibility of implementing an embedded control application using the one-shot task model compared to existing approaches. Similar to the simulation carried out in Section V, the workload consists in a control task controlling a voltage stabilizer sharing the microprocessor with another hard real-time periodic task.

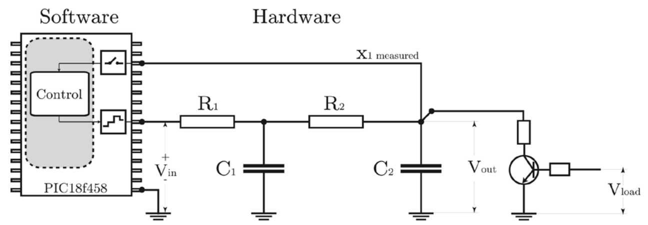

Fig. 13. Physical system.

The strategies that are evaluated are the naif, the one-sample and the one-shot task model, as well as the case where the control task executes in isolation, named "reference." The first three strategies are designed as described in Section V. The switching task model and the split task model are omitted in the implementation because their performance lie in the middle and because they are not as common as the others. The true comparison should be between the one-sample and the one-shot. Finally, the reference is introduced to assess the case where the control task is not subject to jitters. In each model, the control task implements the pseudo-code given in Fig. 6 and follows the specifications given in Section V-B in order to track the square wave. In addition to the controller gain, a deadbeat reduced observer for estimating the second state variable has been designed. This differs from the simulation described in Section V where it was assumed to have available all state vector.

A. Hardware and Software Details

The hardware platform used in the experiments is based on the Microchip PIC18FXX8 micro-controllers family[2] and the RTKPIC18 real-time kernel [24]. In particular, for this experiment, the kernel has been ported to a PIC18F458. Fig. 13 illustrates the control task and the voltage stabilizer circuit in the real experiment. The control signal generated by the control task is sent to the circuit, as illustrated by V_{in}. The sampled controlled variable is V_{out}, as illustrated by $x_{1measured}$. The circuit can be perturbed by a load tension, as illustrated by V_{load}.

Details on the modifications made in the kernel [24] related to the task structure, task control block, and I/O operations for implementing the task models can be found in [19]. The code for implementing each type of task model, also explained [19], confirms that embedded control applications using the one-shot task model does not require specific kernel support compared to the case of using the one-sample or the naif task model.

B. Performance Evaluation

Similar to Sections V and VI, the performance evaluation is measured in terms of the cumulative error using the cost (14) during the simulation time. Therefore, the lower the curve, the better the performance. Fig. 14 show the results of the four

[2]http://www.microchip.com/

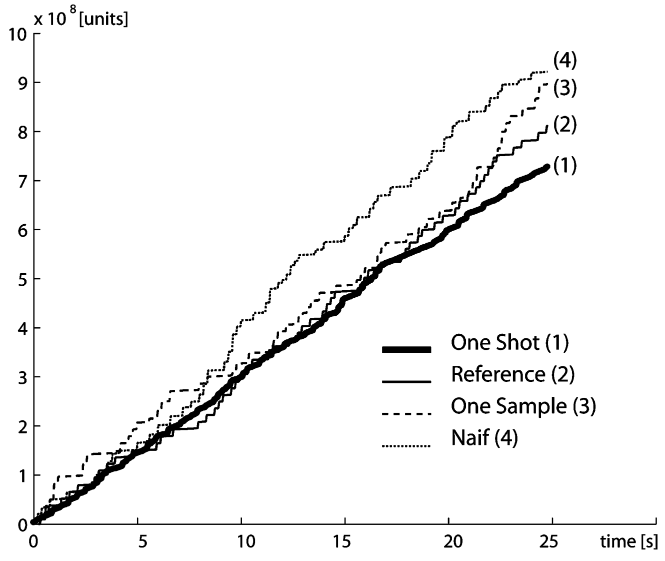

Fig. 14. Performance evaluation of the control experiment.

strategies. As expected, the naif task model gives the worst performance. The best performance is achieved by the one-shot approach, with an increment in performance of 20% with respect to the naif approach.

It is interesting to note two important facts. First, the one-sample approach curve lies between the naif and the one-shot. Therefore, this proves that the most current approach to avoid the jitter problem is outperformed by the approach presented in this paper. But even more interesting is to compare the one-shot curve with respect the reference curve. The reference curve is the performance of the control task when it is executed in isolation, that is, without jitters. As it can be seen in the figure, the one-shot even outperforms this scenario. This is due to the fact that controllers implemented using the one-shot task model are more responsiveness to perturbations, as explained in Section IV-B. The reference curve corresponds to the controller executing without jitters, that is, sampling at each release time and actuating right after the control algorithm computation (which is less than the period). Therefore, all perturbations affecting the circuit between the end of the control algorithm computation and the release time are detected at the next release time. However, in the one-shot, since samples are taken later than the release time, more perturbations can be detected. Note that the real benefit of this property depends on the perturbation arrival times as well as on the workload. This property has been also observed in previous simulations.

VIII. Conclusions

We have presented the one-shot task model for real-time control systems. This task model is synchronized at the actuation instants rather than at the sampling instants. This has been shown to provide interesting properties. From the scheduling point of view, the new task model can be seamlessly integrated into existing scheduling theory and practice, while minimizing the hardware interrupts required by previous solutions, which in turn improves task set schedulability. From a control perspective, the one-shot task model absorbs jitters because it allows irregular sampling, and improves reactiveness in front of perturbations, which even permits to achieve better performance than the case where controllers are executed in isolated processors. Simulations and the reported experimental results corroborate the promised benefits and show the implementation feasibility of the one-shot task model.

References

[1] P. Albertos and J. Salt, “Digital regulators redesign with irregular sampling,” in *Proc. IFAC World Congr.*, 1990, vol. 8, pp. 157–161.
[2] P. Albertos and A. Crespo, “Real-time control of non-uniformly sampled systems,” *Control Eng. Pract.*, vol. 7, no. 4, pp. 445–458, 1999.
[3] K. J.Åström and B. Wittenmark, *Computer-Controlled Systems*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1997.
[4] K.-E.Årzén, A. Cervin, J. Eker, and L. Sha, “An introduction to control and scheduling co-design,” in *Proc. 39th IEEE Conf. Decision and Control*, 2000.
[5] P. Balbastre, I. Ripoll, J. Vidal, and A. Crespo, “A task model to reduce control delays,” *J. Real-Time Syst.*, vol. 27, no. 3, pp. 215–236, 2004.
[6] S. Baruah, G. Buttazzo, S. Gorinsky, and G. Lipari, “Scheduling periodic task systems to minimize output jitter,” in *Proc. 6th Int. Conf. Real-Time Computing Systems and Applications*, 1999.
[7] G. Buttazzo, *Hard Real-Time Computing Systems. Predictable Scheduling Algorithms and Applications*. New York: Springer, 1997.
[8] G. Buttazzo, “Research trends in real-time computing for embedded systems,” *ACM Sigbed Rev.*, vol. 3, no. 3, 2006.
[9] M. M. B. Gaid, A. Ela, and Y. Hamam, “Optimal integrated control and scheduling of systems with communication constraints,” in *Proc. 44th IEEE Conf. Decision and Control and Eur. Control Conf.*, Dec. 2005.
[10] R. Castane, P. Martí, M. Velasco, A. Cervin, and D. Henriksson, “Resource management for control tasks based on the transient dynamics of closed-loop systems,” in *Proc. 18th Euromicro Conf. Real-Time Systems*, 2006.
[11] A. Cervin, “Improved scheduling of control tasks,” in *Proc. 11th Euromicro Conf. Real-Time Systems*, 2001.
[12] A. Cervin and J. Eker, “The control server: A computational model for real-time control tasks,” in *Proc. 15th Euromicro Conf. Real-Time Systems*, 2003.
[13] J. Eker, P. Hagander, and K. E. Arzen, “A feedback scheduler for real-time controller tasks,” *Control Eng. Pract.*, vol. 8, no. 12, pp. 1369–1378, 2000.
[14] T. A. Henzinger, B. Horowitz, and C. M. Kirsch, “Giotto: a time-triggered language for embedded programming,” in *Proc. 1st Int. Workshop Embedded Software*, 2001, vol. 2211, LNCS, pp. 166–184.
[15] D. Henriksson, A. Cervin, and K.-E. Årzén, “Truetime: Simulation of control loops under shared computer resources,” in *Proc. 15th IFAC World Congr. Automatic Control*, Jul. 2002.
[16] K. Jeffay and D. L. Stone, “Accounting for interrupt handling costs in dynamic priority tasks systems,” in *Proc. 14th IEEE Real-Time System Symp.*, 1993.
[17] B. Lincoln, “Jitter compensation in digital control systems,” in *Proc. 2002 Amer. Control Conf.*, May 2002.
[18] C. L. Liu and J. W. Layland, “Scheduling algorithms for multiprogramming in a hard real-Time environment,” *J. ACM*, vol. 20, no. 1, pp. 40–61, 1973.
[19] C. Lozoya, M. Velasco, and P. Martí, The One-Shot Task Model for Real-Time Embedded Control Systems, 2007. [Online]. Available: http://www.upcnet.es/~pmc16/07RRshot.pdf.
[20] P. Martí, J. M. Fuertes, R. Villà, and G. Fohler, “On real-time control tasks schedulability,” in *Proc. Eur. Control Conf.*, 2001.
[21] P. Martí, G. Fohler, K. Ramamritham, and J. M. Fuertes, “Jitter compensation for real-time control systems,” in *Proc. 22rd IEEE Real-Time System Symp.*, 2001.
[22] P. Martí, C. Lin, S. A. Brandt, M. Velasco, and J. M. Fuertes, “Optimal state feedback based resource allocation for resource-constrained control tasks,” in *Proc. 25th IEEE Int. Real-Time Systems Symp.*, 2004.
[23] P. Martí and M. Velasco, “Toward flexible scheduling of real-time control tasks: Reviewing basic control models,” in *Proc. 10th Int. Conf. Hybrid Systems, Computation and Control*, 2007, LNCS.
[24] R. Marau, P. Leite, L. Almeida, M. Velasco, P. Martí, and J. M. Fuertes, *Implementing Flexible Embedded Control on a Simple Real-Time Multitasking Kernel,* Research Rep. ESAII-RR-07-10, Apr. 2007, Autom. Control Dept., Tech. Univ. Catalonia.

[25] S.-H. Oh and S.-M. Yang, "A modified least-laxity-first scheduling algorithm for real-time tasks," in *Proc. 5th Int. Conf. Real-Time Computing Systems and Applications*, 1998.

[26] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin, "On task schedulability in real-time control systems," in *Proc. IEEE Real-Time Systems Symp.*, 1996, pp. 13–21.

[27] D. Seto, J. P. Lehoczky, and L. Sha, "Task period selection and schedulability in real-time systems," in *Proc. IEEE Real-Time Systems Symp.*, 1998.

[28] B. Wittenmark, J. Nilsson, and M. Törngren, "Timing problems in real-time control systems," in *Proc. Amer. Control Conf.*, 1995.

Camilo Lozoya received the degree in electronics engineering in 1993 from the Chihuahua Institute of Technology, Chihuahua, Mexico. Currently, he is pursuing the Ph.D. degree in automatic control from the Technical University of Catalonia, Barcelona, Spain.

Since 2004, he has been a Professor in the Engineering School of the Monterrey Institute of Technology—Chihuahua campus. Previously, he worked as a project manager in the information technology department of the Motorola Chihuahua factory from 1996 to 2004. His research interests include embedded systems, distributed control systems, and real-time systems.

Manel Velasco graduated in maritime engineering in 1999 and received the Ph.D. degree in automatic control in 2006, both from the Technical University of Catalonia, Barcelona, Spain.

Since 2002, he has been an Assistant Professor in the Department of Automatic Control at the Technical University of Catalonia. He has been involved in research on artificial intelligence from 1999 to 2002 and, since 2000, on the impact of real-time systems on control systems. His research interests include artificial intelligence, real-time control systems, and collaborative control systems, especially on redundant controllers and multiple controllers with self-interacting systems.

Pau Martí (M'02) received the degree in computer science and the Ph.D. degree in automatic control from the Technical University of Catalonia, Barcelona, Spain, in 1996 and 2002, respectively.

Since 1996, he has been an Assistant Professor in the Department of Automatic Control at the Technical University of Catalonia. During his Ph.D., he was a visiting student at Malardalen University, Malardalen, Sweden. From 2003 to 2004, he held a research fellow appointment in the Computer Science Department at the University of California at Santa Cruz. His research interests include embedded systems, control systems, real-time systems, and communication systems.