Real-Time Syst (2013) 49:267–307
DOI 10.1007/s11241-012-9174-9

Resource and performance trade-offs in real-time embedded control systems

Camilo Lozoya · Pau Martí · Manel Velasco · Josep M. Fuertes · Enrix X. Martin

Published online: 28 December 2012

Abstract Embedded control systems are subject to conflicting demands: end-users ask for devices with better capabilities while strong industrial competition impose tight cost constrains that results in devices with limited resources. Current research in the multidisciplinary embedded systems discipline indicates that by combining real-time and control systems it is theoretically feasible to design resource-constrained embedded control systems capable of trading-off control performance and resource usage.

This paper focuses on the implementation feasibility of recent state-of-the-art resource/performance-aware (RPA) policies that can be applied to a set of control loops that concurrently execute on a microprocessor. The objective of these policies is to improve control performance and/or to minimize resource utilization. The paper first reviews existing state-of-the-art RPA policies. Then it presents a performance evaluation framework (PEF) that permits to assess whether RPA policies can be implemented in practice. The PEF is designed using a modular approach and following the guidelines obtained by a taxonomic analysis performed on the state-of-the-art RPA policies. Finally, a case study is presented where the PEF is applied to a set of representative RPA policies. The case study reveals that the modularity of the PEF

C. Lozoya · P. Martí (✉) · M. Velasco · J.M. Fuertes · E.X. Martin
Automatic Control Department, Technical University of Catalonia, Pau Gargallo 5, 08028 Barcelona, Spain
e-mail: pau.marti@upc.edu

C. Lozoya
e-mail: camilo.lozoya@upc.edu

M. Velasco
e-mail: manel.velasco@upc.edu

J.M. Fuertes
e-mail: josep.m.fuertes@upc.edu

E.X. Martin
e-mail: enric.xavier.martin@upc.edu

allows tailoring the framework to evaluate any specific RPA policy, which indicates that RPA policies can be implemented in practice. But it also reveals that the problem of assessing diverse RPA policies in fair conditions implies facing and solving conflicting demands by even taking decisions that may not favor equal all policies under evaluation. Nevertheless, the comparative analysis permits identifying potential benefits and drawbacks of each policy, as well as extracting design guidelines for future real-time embedded control systems theory and practice.

Keywords Embedded control systems · Resource utilization · Control performance

1 Introduction

Embedded devices are being widely used in many areas, playing a key role in our society. An embedded system is a special-purpose computer system designed to perform dedicated control activities, interacting with the environment. Currently 98 % of computing devices in the world are embedded systems. Conservative estimations indicate a forecast of over 40 billion of available embedded devices worldwide by 2020 (Artemis 2006).

Embedded systems market demands devices with more and better functionality at lower prices. In addition, embedded devices are designed under space, weight and energy constraints, imposed by cost restrictions. As a consequence, embedded applications typically run on small processing units with limited memory and computational power. This increases embedded control systems complexity from both the control and computer science perspectives. As a consequence researchers in the computer and control fields are becoming increasingly aware of the need for an integrated scientific and technological perspective on the role that computers play in control systems and that control can play in computer systems (Sanz and Årzén 2003).

However, by tradition, the design of embedded control systems has been based on the separation-of-concerns principle (Årzén and Cervin 2005). While this principle has proved advantageous from a designer's perspective, it does not necessarily lead to cost-effective implementations because each designer is forced to adopt a conservative viewpoint that may lead to unnecessary over-provisioning in the system implementation and hence to higher system costs and sub-optimal control performance. For example, the simple approach based on the assumption that controllers can be modeled and implemented as periodic real-time tasks with fixed periods and scheduled with standard scheduling policies such as fixed priority (FP) or earliest deadline first (EDF) (Liu and Layland 1973) does not guarantee efficient resource usage and outstanding control performance. First, the selection of fixed rates of execution is not an easy task: low rates imply low resource utilization but also imply low control performance (and viceversa). Second, embedded control systems usually operate in dynamic environments where application demands, computational workload and resource availability experience changes during execution time. Therefore, the enforcement of a fixed rate can be inappropriate.

To overcome these limitations, there has been a recent interest in developing approaches where the concerns of computer science and control systems engineers are treated in a unified manner (see an initial review in Sha et al. 2004). These approaches demand flexible and adaptable scheduling policies and controller designs capable of making an efficient use of the computational resources (Buttazzo 2006). Control tasks periodicity and computation times are the main factors that determine resource utilization and control performance. By varying them, different methodologies can serve towards the same goal.

For a particular task, given a constant period, its resource demands can be regulated by allowing varying computation times, which at the end will result in different control accuracy. This type of methodology uses the anytime control paradigm, e.g. Bhattacharya and Balas (2004), Quagli et al. (2009), or Gupta (2009), and it can be supported at the scheduling level for example by adaptive reservations, as in Fontanelli et al. (2011).

Or alternatively, for a particular control task, given a fixed execution time, periods can be selected and/or changed using different policies to provide different levels of control performance. Within the approaches where the selection of the sampling period is the key, which is the scope of this paper, two new trends can be identified in the literature (Lozoya et al. 2007). The first one, often referred to as "*feedback scheduling*" (FBS), is based on applying efficient sampling period selection techniques that account for processor load and plants dynamics in such a way that the aggregated control performance delivered by the set of control loops is improved.[1] The second trend is based on applying "*event-driven control*" (EDC) techniques to design controllers with non-periodic activation patterns to minimize their resource demands while still guaranteeing stability and acceptable control performance.

Henceforth, both FBS and EDC are referred to as resource/performance-aware (RPA) policies. RPA policies mainly focus on theory, whereas practical aspects are often omitted. Conversely to the initial problem targeted by these policies, that is, to minimize or keep resource requirements to meet the tight cost constraints related with mass production and strong industrial competition, research advances seem to require sophisticated procedures that may impair a cost-effective implementation. Despite of the great variety of different RPA policies that have been proposed in the literature, surveys focusing on implementation aspects covering both FBS and EDC approaches are not available. In addition, no unified framework exists to assess the implementation feasibility of these policies in order to evaluate the potential benefits offered by the theory.

1.1 Paper contributions

The paper contributions can be summarized as follows:

– An exhaustive state-of-the-art of RPA policies is presented in order to review existing results. In addition, a non-exhaustive survey on simulation tools and real-time

[1]The term "*feedback scheduling*", as it will be explained in Sect. 2.1.1, was first introduced to denote policies that use feedback from scheduling to on-line apply different sampling periods and adapt controllers' parameters. Although this applies to a subset of the approaches reviewed later in this category, the paper also includes under this category approaches where the sampling period that applies on-line is selected off-line considering its impact on control performance according to different strategies. Hence, the term "*feedback*" may no longer refer to feedback as it is interpreted in control, but in a more general sense.

platforms is presented to support the specification of a performance evaluation framework (PEF). This is presented in Sect. 2.
– From the state-of-the-art of RPA policies, the paper presents in Sect. 3 the definition and implementation of the PEF that allows assessing RPA policies implementation feasibility. In particular, the definition of the basic services to be supported by the PEF are derived from a taxonomic analysis that reveals key characteristics and tendencies on embedded control systems analysis and design.
– Finally, the application of the PEF to a set of selected and representative RPA policies is described in Sect. 4. It illustrates the applicability of the PEF while assessing benefits and drawbacks of representative tendencies identified in the taxonomy in embedded control systems design.

2 State-of-the-art

2.1 Resource/performance-aware (RPA) policies

The real-time and control communities have provided diverse theoretical results on both control and resource optimization for resource limited computing systems concurrently executing several controllers. Apart from those approaches covering the anytime control paradigm, loosely speaking, the existing results surveyed in this paper suggest to efficiently select controllers' sampling periods such that controllers' execution rates are different from those provided by the standard periodic sampling approach (Åström and Wittenmark 1997). These results, which are divided into FBS and EDC approaches, are reviewed next in terms of advances of the state of the art, but also in terms of which type of evaluation they have been subject to, and also in terms of implementability aspects. It is important to stress that both FBS and EDC techniques have been also developed for networked control systems. However, the next review and the paper contributions only apply to microprocessor-based embedded control systems.

2.1.1 Feedback scheduling (FBS)

Feedback scheduling refers to the problem of sampling period selection for real-time control tasks that compete for limited resources such as processor time. Its goal is to optimize the aggregated control performance achieved by all tasks by using efficiently the scarce resources. The standard FBS approach includes an optimization procedure that dictates how task periods are assigned to each control task in order to optimize the overall control performance. Examples of feedback scheduling results found in the literature are Seto et al. (1996, 1998b), Zhao and Zheng (1999), Eker et al. (2000), Rehbinder and Sanfridson (2000), Henriksson et al. (2002), Palopoli et al. (2002b, 2005), Cervin et al. (2002), Chandra et al. (2003), Martí et al. (2004, 2009a), Henriksson and Cervin (2005), Castañé et al. (2006), Ben Gaid et al. (2006, 2009), Bini and Cervin (2008), Samii et al. (2009a, 2009b), or Cervin et al. (2011).

Initial research on feedback scheduling is found in Seto et al. (1996), where an off-line algorithm was proposed to select tasks sampling periods based on optimizing a cost function that describes the relationship between the sampling rate and the quality of control, subject to an EDF CPU utilization constraint. This concept was extended to RM scheduling in Seto et al. (1998b). Although both approaches were described in detail, only simulations were provided for validation purposes. A further extension to redundant controllers was presented in Chandra et al. (2003), using the same simulated scenarios (temperature control, bubble control, and inverted pendulum) over the Simplex (Seto et al. 1998a) software platform. These type of off-line approaches were extended in Bini and Cervin (2008) by considering also, apart from sampling periods, input-output delays. The approach was validated using simulated workloads and randomly generated delays.

An off-line optimal period selection method for a set of state feedback controllers using the stability radius as a performance criterion was presented by Palopoli et al. (2002b, 2005). Evaluation was performed on simulated inverted pendulums using the Erika real-time kernel Erika (2011).

The approach in Rehbinder and Sanfridson (2000) presented an off-line approach based on cyclic executives (sequences of tasks' jobs rather than the sampling periods). The proposed approach uses the predicted error over a finite time horizon as the information required to define the optimal sequences of tasks. The approach was validated using simulations on inverted pendulums.

Recently, in Samii et al. (2009a, 2009b) it was proposed an off-line approach to scheduling and synthesis of control applications where the output of the algorithm is the schedule that minimizes the cost for a given number of controllers. Simulations were used to validate the approach, and measurements in a PC running Linux was used for complexity analysis.

An important change with respect to previous works has been the use of an on-line resource manager that uses feedback to dynamically adjust control task attributes in order to optimize control performance. This feedback scheduling concept was used initially in Eker et al. (2000) and further developed in Cervin et al. (2002). In Eker et al. (2000) the authors proposed an on-line optimization algorithm for sampling period assignment, that iteratively adjusts the sampling rates based on the CPU load for the case of LQ controllers. An extension to general linear dynamic controllers is found in Cervin et al. (2002). In both approaches the scheduler attempts to keep the processor utilization as close as possible to a utilization set-point by manipulating the sampling periods. These results were implemented in TrueTime (Cervin et al. 2003). Four simulated inverted pendulum were used as the controlled plants. The same idea but targeting model predictive controllers was presented by Henriksson et al. (2002), which was also validated using TrueTime simulations but on a quadruple-tank laboratory process.

The idea of also considering feedback from the current control performance was proposed in Martí et al. (2004). The sampling period reassignment is based on an optimization procedure that considers the current instantaneous cost measurement from each controlled plant. The algorithm was implemented in a Linux 2.4.20 kernel-based platform (named RBED, Brandt et al. 2003) and extensive experiments were performed on simulated inverted pendulums. The same authors presented an extension (Martí et al. 2009a) where the relation between its cost function and standard quadratic cost functions was established.

The approach by Henriksson and Cervin (2005) further formalized this idea for linear quadratic controllers by incorporating the current plant states into a finite-horizon quadratic cost function, which also took the expected future plant noise into account. The extension to the general case of linear controllers was given in Castañé et al. (2006). Both feedback scheduling strategies were simulated with TrueTime using three different second-order plants: ball and beam, DC motor, and harmonic oscillator. A further improvement is offered by Cervin et al. (2011) where the cost function was developed for general linear dynamic controllers. A proof-of-concept implementation was presented based on the Erika real-time kernel and a set of electronic double integrator circuits as controlled plants.

A complementary approach for the optimal integrated control and real-time scheduling of control tasks was presented by Ben Gaid et al. (2006, 2009). It combined non-preemptive optimal cyclic schedules according to the H_2 performance criterion, with an efficient on-line scheduling heuristic in order to improve responsiveness to disturbances. The validation of these approaches were performed using simulated inverted pendulums.

The main drawback of these approaches, identified in several of the listed papers, refers to whether the implementation of the solution presented in each approach is feasible because the offered theory has not been tested in practice.

2.1.2 Event-driven control (EDC)

The execution of event-driven controllers aims at minimizing resource utilization while ensuring stability or bounding the inter-sampling dynamics of the controlled plants. This is achieved by executing controllers without periodic requirements: controllers jobs are only executed when needed. Event-driven controllers adapt the task period directly in response to the application performance (Årzén 1999), which is often expressed as an event condition on the plant state. Since the state is always changing, this approach generates an aperiodic sequence of control task invocations. Examples of results of event-driven control systems in processor-based platforms are Årzén (1999), Heemels et al. (1999), Åström and Bernhardsson (2002), Velasco et al. (2003, 2009b), Tabuada and Wang (2006), Miskowicz (2006), Tabuada (2007), Lemmon et al. (2007), Johannesson et al. (2007), Suh et al. (2007), Anta and Tabuada (2008a, 2008b, 2009, 2010), Wang and Lemmon (2008a, 2008b, 2009a, 2009b), Heemels et al. (2008), Henningsson et al. (2008), Martí et al. (2009b), Mazo et al. (2009), Mazo and Tabuada (2009).

Initial research on EDC provides evidences that event-driven controllers can reduce resource utilization compared to the standard periodic control approach. In Årzén (1999) the integration of an analog event detector to trigger a PID controller is proposed. Simulations were conducted in the control of a simulated double tank process to assess control performance and processor load of the event-based controller compared to a time-trigger controller. Sharing objectives, a control system to synchronize the position of two motors based on asynchronous measurements was presented in Heemels et al. (1999). The proposed design, for this first order system, was implemented and tested with real inductor motors, using a Texas Instrument DSP (TMS-320C40).

An event-based feedback scheduling strategy was suggested by Zhao and Zheng (1999) where the scheduler chooses one plant to be executed at any time based on specific conditions on asymptotic and exponential stability. Simulations showed that the proposed strategy has a better performance in terms of transient response compared to a standard sequential scheduling policy.

In Åström and Bernhardsson (2002) it was shown using simulations that an event-based controller for an integrator plant disturbed by white noise requires on average, only one fifth of the sampling rate of a periodic controller to achieve the same output variance. Reduction in the number of samples was also observed in several approaches: (1) in the data collection strategy proposed by Miskowicz (2006) (where the sampling action is triggered if the signal deviates sufficiently in relation to the most recent sample), (2) in a modified Kalman filter in Suh et al. (2007) (where experiments conducted on sensor boards showed a relatively small estimation performance degradation), (3) in the event-driven approach in Heemels et al. (2008) (where the control update is only triggered when the tracking or stabilization error violates a given threshold), and (4) in the scheduling schema in Johannesson et al. (2007) targeting sporadic controllers for first order systems, whose performance was measured in terms of the state cost and the average number of events per time unit. In Henningsson et al. (2008) an sporadic controller was compared to a periodic controller in a first order lineal system using the same metrics as before.

In the previous approaches, specific hardware was supposed to trigger the execution of control jobs. A radical change can be found in the self-triggered task model presented in Velasco et al. (2003). The method activates control task executions according to control performance and available CPU. The contribution is that the triggering is performed by each task, which determines at each job execution its next release time. Simulations were conducted using two ball and beam plants to analyze the controlled plant dynamics. In Martí et al. (2009b) the authors formulated an optimization problem targeting to minimize the resource utilization of an event-driven controller that achieves the same cost as a periodic controller. Simulations were conducted using a double integrator plant model. In the same line of research, a self-triggered schema where the decision to execute control tasks was determined by an input-to-state stability setting was presented by Tabuada (2007). Simulation of a second order plant was used to verify system stability. This approach was further analyzed in Anta and Tabuada (2008a, 2008b, 2009, 2010) where the self-trigger model was extended for non-linear systems. In a parallel track, in Lemmon et al. (2007) a self-triggered mechanism was presented based on a robust control setting and implemented using the elastic scheduling Buttazzo et al. (1998). Simulations were conducted over three inverted pendulums plants. This work was extended in Wang and Lemmon (2008b, 2008a, 2009a, 2009b), to quantify how robust the control system's performance is with respect to variations in tasks' periods and deadlines. An inverted pendulum model was used as a plant, and the state error was used for performance measurement.

A general procedure leading to self-triggered implementations of feedback controllers was proposed by Mazo et al. (2009). The approach was simulated with a fourth order batch reactor model, and the performance was measured by the size of the inter-execution times. In Mazo and Tabuada (2009) the approach was analyzed to study its robustness with respect to disturbances. In Velasco et al. (2008a) it was provided a general explicit approximated solution to compute activation times for self-triggered controllers. An extension for both FP and EDF schedulability analysis for self-triggered tasks is introduced. The approach was corroborated using a ball and beam plant.

Unfortunately, although work on event-driven control started to appear in the 50's (Ellis 1959), the discipline still lacks a mature system theory, schedulability analysis has not been addressed, and rarely implementation issues are discussed. Regarding implementability, the overhead imposed by EDC schemes becomes a major issue, especially for approaches involving on-line estimation/optimization. Hence, the implementation feasibility analysis, which also considers overheads, is a challenging task, and it will indicate whether EDC theoretical approaches can be implemented in practice.

2.1.3 On the completeness of the state-of-the-art

As mentioned in the introduction, approaches based on the anytime control paradigm and resource reservations have been left out of this paper.

It is also important to mention that other approaches could have been also treated in the state of the art, such as statically designed systems and mode switches. These approaches have a larger application scope than the surveyed RPA policies, and in general they are not designed focusing only on improving control performance and/or reducing resource utilization. Therefore, statically designed systems and mode switches have not been included explicitly in the study. However, statically designed systems and mode switches can be considered to be closely related to RPA policies. On one hand, switched control systems (Liberzon 2003) is an example of an important discipline in control systems theory whose techniques could be applied to the same problem treated by RPA policies. Or looking at the real-time side, mode changes protocols (Real and Crespo 2004) are techniques that could be useful for supporting the implementation of RPA approaches. On the other hand, it is worth noting that several on-line FBS and EDC approaches are based on an off-line design procedure that establishes a set of sampling periods and a set of control gains that will switch at run-time. Therefore, they can already be considered statically designed systems with mode switches. Therefore statically designed systems and mode switches are, at some extend, implicitly represented in the analysis carried out through the paper.

2.2 Effective implementation of control algorithms

For embedded control systems implemented in small microprocessors enabled with real-time technology, control algorithms are implemented as real-time tasks. Simplifying, periodic tasks, which are the common task model for FBS policies, have regular arrival times and are characterized by a period, a deadline and its computation time. Aperiodic tasks, which are the natural task model for EDC policies, have irregular arrival times, and are characterized by a deadline and its computation time.

For both periodic and aperiodic tasks, depending on how the sample (input) and actuation (output) operations are performed within the task body, different control tasks models can be distinguished, such as the "standard", "one-sample", "split", or "one-shot" task models. The standard task model, identified in Årzén et al. (2000) as the common practice implementation of control algorithms using real-time technology, assumes I/O operations at the beginning and end of each task instance execution. Its implementation is simple but introduces sampling and latency jitters. The one-sample task model solves these problems by executing I/O operations at the beginning and end of the task period via hardware interrupts, as in Liu and Layland (1973). The drawback of this approach is that it introduces a longer although predictable I/O delay. The split task model (Cervin et al. 2003) uses also hardware interrupts for I/O operations but splits the task body into parts to minimize the I/O delay. The one-shot task model (Lozoya et al. 2008b) forces only actuation by interrupts while sampling is performed at the beginning of each task instance, thus allowing irregular sampling but adding computational overhead due to the inclusion of observers used to compensate for degrading dynamic effects caused by the I/O delays.

Effective implementation of control algorithms using real-time technology is not straightforward. Several task models exists. Some of them may suit better a particular RPA policy than others, they offer different benefits and drawbacks, and the PEF must be flexible enough to accommodate all of them.

2.3 Evaluation of RPA policies

Partial evaluations for RPA policies can be found in the literature. An evaluation of a feedback scheduling policy using a real-time kernel can be found in Marau et al. (2008). A first attempt to compare feedback scheduling methods can be found in Cervin and Alriksson (2006), and a first attempt to analyze resource demands of a class of event-driven control systems is found in Velasco et al. (2008b, 2009a). In Velasco et al. (2010) it is provided the first comparison between a feedback scheduling policy and an event-driven multitasking control system.

Apart from these partial evaluations, FBS and EDC advances are evaluated and compared with the performance achieved by the traditional static periodic approach of control systems implementation, and considering similar circumstances. Therefore the benefits of each novel approach are only measured considering the static approach but they are not compared to other similar RPA policies. Hence, it becomes very difficult to analyze and evaluate the real benefits of each new approach compared to the state-of-the-art results.

Tables 1 and 2 summarize how state-of-the-art RPA approaches surveyed in Sect. 2.1 were evaluated in terms of control performance and/or resource efficiency. The summary also shows whether simulation or experimentation was applied, the targeted plants used in the analysis, and indicates how control performance is measured and if processor load is considered.

By analyzing both tables, it can be noticed that in most of the cases the proposed RPA policies were only simulated using a computational tool, and only in a few cases real experiments were developed. Another important aspect is the selection of the evaluation metrics. In practically all the FBS approaches the main parameter to

Table 1 FBS evaluation parameters and platform

Method	Plant	Evaluation Parameters		Platform	
		Control Performance	Processor Load	Simulation	Experimental
Seto et al. (1996)	bubble control	Quadratic	%	Yes	No
Seto et al. (1998b)	temperature control, bubble control	Quadratic	No	No	No
Eker et al. (2000)	inverted pendulum	Quadratic	No	Yes	No
Rehbinder and Sanfridson (2000)	inverted pendulum	Quadratic	No	Yes	No
Henriksson et al. (2002)	quadruple tank process	Error	No	Yes	No
Cervin et al. (2002)	inverted pendulum	Quadratic	No	Yes	No
Palopoli et al. (2002b)	scalar system	Robustness	No	Yes	No
Chandra et al. (2003)	temperature control, bubble control, inverted pendulum	Quadratic	No	Yes	Yes
Martí et al. (2004)	inverted pendulum	Error	%	Yes	Yes
Henriksson and Cervin (2005)	integrator	Quadratic	No	Yes	No
Palopoli et al. (2005)	inverted pendulum	Robustness	No	No	Yes
Ben Gaid et al. (2006)	unknown	Error	No	Yes	No
Castañé et al. (2006)	ball and beam, dc motor, harmonic oscillator	Quadratic	No	Yes	No
Bini and Cervin (2008)	scalar plants	Quadratic	No	Yes	No
Ben Gaid et al. (2009)	inverted pendulum, dc motor	Error	Rate	Yes	Yes
Martí et al. (2009a)	inverted pendulum, ball and beam	Quadratic	%	Yes	No
Samii et al. (2009b)	inverted pendulum, ball and beam, dc servos, harmonic oscillator	Quadratic	Runtime	No	Yes
Cervin et al. (2011)	double integrator	Quadratic	Overhead	No	Yes

be measured is control performance and in most of the cases it is measured by using a quadratic cost function. Processor load for FBS is calculated only in few cases where it is measured in terms of percentage of use. And in general, the computational overhead caused by the implementation of the RPA policy is not taken into consideration. For the EDC approaches the main parameter to be measured is processor load, either counting the number of jobs, or with actual measures of the load in terms of percentage of processor utilization or total runtime. The control performance for EDC approaches is just analyzed by observing the transient response and sometimes by measuring the deviation from the desired set-point, i.e. transient error.

Table 2 EDC evaluation parameters and platform

Method	Plant	Evaluation Parameters		Platform	
		Control Performance	Processor Load	Simulation	Experimental
Årzén (1999)	double tank process	No	%	Yes	No
Heemels et al. (1999)	electrical motor	Error	No	No	Yes
Zhao and Zheng (1999)	second-order plant	Transient response	No	Yes	No
Åström and Bernhardsson (2002)	integrator	Variance	No	Yes	No
Velasco et al. (2003)	ball and beam	Transient response	%	Yes	No
Tabuada (2007)	second-order plan	Error	No	Yes	No
Lemmon et al. (2007)	inverted pendulum	Transient response	No	Yes	No
Johannesson et al. (2007)	first-order plant	Quadratic	Jobs	Yes	No
Henningsson et al. (2008)	first-order plant	Quadratic	Jobs	Yes	No
Anta and Tabuada (2008a)	non-linear plant	Error	Jobs	Yes	No
Anta and Tabuada (2008b), Heemels et al. (2008)	jet engine compressor	Error	Jobs	Yes	No
Wang and Lemmon (2008b)	inverted pendulum	Error	Jobs	Yes	No
Wang and Lemmon (2008a)	inverted pendulum	Error	Jobs	Yes	No
Anta and Tabuada (2009)	non-linear plant	No	Runtime	Yes	No
Martí et al. (2009b)	double integrator	Transient response	Jobs	Yes	No
Mazo et al. (2009)	batch reactor model	Stability	Jobs	Yes	No
Mazo and Tabuada (2009)	batch reactor model	Stability	Jobs	Yes	No
Velasco et al. (2009b)	double integrator	Quadratic	%	No	Yes
Wang and Lemmon (2009a)	inverted pendulum	Error	Jobs	Yes	No
Wang and Lemmon (2009b)	inverted pendulum	No	Jobs	Yes	No
Anta and Tabuada (2010)	jet engine compressor	Error	Jobs	Yes	No

Looking at these tables, the PEF must allow the evaluation of several metrics regarding control performance and resource usage, and it is also desirable to permit simulation and experimentation of the diverse RPA policies.

2.4 Performance evaluation tools

A brief review of some representative simulation tools and real-time platforms is presented here. In order to limit the scope, the focus has been placed on recent tools and platforms suitable for real-time and control co-design.

2.4.1 Simulation tools

The real-time research community has developed a number of tools for schedule simulation, timing analysis and schedule generation, such as STRESS (Audsley et al. 1994) and DRTSS (Storch and Liu 1996). Meanwhile, the control community have used mathematical software for simulations such as Matlab/Simulink created by MathWorks since 1984 or Scilab/Scicos created by the INRIA (Institut National de Recherche en Informatique et Automatique) and the ENPC (Ècole Nationale des Ponts et Chaussées) since 1990.

Recently the following computation tools have been developed for the research of co-analysis and co-simulation of embedded control systems. Ptolemy II (Eker et al. 2003) supports heterogeneous, hierarchical modeling, simulation, and design of concurrent systems, especially embedded systems. In Ptolemy the real-time control system simulation is just one part of a larger framework. The Aida tool-set (Redell et al. 2004) integrates the design and performance analysis of control systems with embedded real-time system design. The tool-set enables specification and analysis of real-time implementations of control applications. Aida focuses more in the model-based design rather than in the co-simulation analysis. RTSIM (Palopoli et al. 2002a) is a tool that is aimed at simulating real-time embedded control systems. The main goal is to facilitate co-simulation of real-time controllers and controlled plants in order to evaluate the timing properties of the architecture in terms of control performance. Torsche (Sucha et al. 2006) is a Matlab-based toolbox including scheduling algorithms that can be used to investigate application performance prior to its implementation. TrueTime (Cervin et al. 2003) is a MATLAB/Simulink-based toolbox that facilitates simulation of the temporal behavior of a multitasking real-time kernel executing control tasks.

Recent surveys on simulation tools for real-time and control systems co-design can be found in Årzén (2005) and Törngren et al. (2006).

2.4.2 Real-time platforms

Many of the RPA policies can be implemented by any existing real-time operating system or kernel supporting scheduling frameworks or scheduling algorithms that permit dynamic task period adjustment at run time and guarantees that no deadline is missed during the adjustment, that is trading off predictability in the performance and efficiency in the resource utilization, as also demanded by other type of modern control applications (Stankovic 1996). Nowadays, there are more than a hundred commercial products that can be categorized as real-time operating systems (Buttazzo 2004). However, most of these kernels are based on fixed priority scheduling.

The rest of this section only focuses on recent real-time kernels developed for research purposes, since this generation of new operating systems include flexible features that allow the analysis and implementation of novel RPA policies. Examples of kernels are Shark kernel (Gai et al. 2001), Marte OS (Rivas and Harbour 2001), PaRTiKle (Peiro et al. 2007) and Erika (Erika 2011).

SHARK (Soft and Hard Real-time Kernel) (Gai et al. 2001) is a dynamic configurable research kernel architecture designed for supporting a simple implementation, integration and comparison of scheduling algorithms. The kernel supports the development and testing of new scheduling algorithms, aperiodic servers and resource management protocols. It is compliant with the POSIX (Portable Operating System Interface) 1003.13 PSE52 specifications (Posix 1996). MARTE (Minimal Real-Time OS for Embedded Applications) (Rivas and Harbour 2001) is a real-time kernel for embedded applications that follows the Minimal Real-Time POSIX.13 subset (Posix 1998). The applications planned for this kernel are industrial embedded systems that are mostly static, with the number of threads and system resources well known at compile time. PaRTiKle (Peiro et al. 2007) is a embedded real time operating system designed to be POSIX compliant. PaRTiKle has been designed to support applications with real-time requirements, providing features such as full preemptability, minimal interrupt latencies, and all the necessary synchronization primitives, scheduling policies, and interrupt handling mechanisms needed for this type of applications. ERIKA (Erika 2011) is a small size, but fully functional kernel distributed by Evidence s.r.l. and supports many features from the OSEK/VDX (Open Systems and the Corresponding Interfaces for Automotive Electronics/Vehicle Distributed eXecutive) standard (Osek 2005). ERIKA has been designed to be an effective educational and research platform for real-time programming in embedded systems.

Platforms for testing embedded control systems can also be found in educational papers such as Årzén et al. (2005) and Martí et al. (2010).

2.5 Summarizing discussion

The state of the art of RPA policies shows a great variety of approaches that even using different techniques always target trading-off resource utilization (mainly in terms of CPU time) and control performance. In addition, the availability of consolidated specific simulation tools (for co-simulation of embedded control systems) and modern real-time kernels (enabled with advanced features that allow implementing diverse FBS and EDC approaches) permits designing a performance evaluation framework to assess the implementation feasibility of RPA policies. It will also allow evaluate state-of-the-art under fair conditions, and identify how different strategies impact on control performance, resource utilization and computational overhead. This will provide insight on the benefits and drawbacks of each RPA policy.

3 Performance evaluation framework (PEF)

Given the great diversity of RPA approaches, the framework must be generic enough to accommodate different policies, and it also must be flexible and accurate enough to facilitate the implementation of these policies while permitting to assess their exact operation in fair/comparable scenarios.

3.1 Taxonomy on RPA policies

This section updates a preliminary taxonomy presented by Lozoya et al. (2007) on the RPA policies. The taxonomy will help on the definition of the PEF. Many of the RPA policies go beyond than just finding the *best* values for control task periods. They provide complete theoretical frameworks tailored to effective concurrent execution of control tasks. In order to perform a taxonomic analysis, several classifying keywords have to be selected. They have to be generic enough to be able to produce a value for all policies. And they have to be specific enough to allow differentiating between policies. It is important to note that different classifying keywords than the ones selected and reviewed next could have been selected as long as they would have help sorting all RPA policies. The updated taxonomy is summarized in Table 3, which shows all RPA policies sorted in chronological order, and characterized by the classifying keywords.

3.1.1 Classifying keywords

The first classifying keywords is *which* criterion is used to select sampling periods (or sequences or schedules). Two main criterion can be identified: optimization approach or bounding the inter-sampling dynamics. In the optimization approaches sampling periods are selected to solve an optimization problem. In the approaches based on bounding the inter-sampling dynamics, sampling periods are selected to keep each closed loop dynamics within predefined thresholds.

The criterion determines *what* real-time paradigm is demanded in the underlying executing platform, that is, whether the solution requires a real-time architecture capable of accommodating a time-triggered (TT) or an event-triggered (ET) task activation paradigm. The next classifying keyword refers to *who* should decide which task to execute (triggering entity). Two options can be distinguished: to have a global resource manager (RM) that decides the *best* periods for the set of control tasks, or to allow the same control tasks deciding when to execute according to predefined conditions.

A key point is also whether the period selection is performed off-line or on-line, that is, *when* the decision is taken, which impacts on overhead and adaptability. On-line algorithms introduce computational overhead, which can be reduced by using look-up tables. However, they have the ability to adapt to changes by varying the demanded resources and/or the control accuracy.

From the reviewed RPA approaches it is also of interest to determine *where* the dynamics accounted for the selection of tasks periods are located. Again, two main options can be identified: to use information from the state of the kernel (mainly in terms of resource utilization but also from scheduling properties) and to use information from the plant dynamics. The latter can be mainly in the form of instantaneous measures (taken using different techniques) or predictions of the plant dynamics in a given finite horizon. However, other measures have been also used such as heuristic approximations or robust control settings related to plant stability.

A final classifying keyword is *how* the decision is given. The solution provided by most RPA approaches is given in the form of sampling periods for the set of control tasks, with different flavors, namely static periods when they are kept constant, varying periods when they change at a given pseudo-periodicity, or time intervals when they constantly vary. However, some methods do not establish sampling periods, rather they provide periodic sequences of ordered control task instances.

3.1.2 Taxonomic analysis

By looking at Table 3 the following conclusions can be drawn. First of all, the first three classifying keywords are closely correlated. FBS approaches usually use the optimization criteria, demand a TT architecture and use some sort of RM for re-setting the sampling periods. On the contrary, EDC approaches usually use the bounding intersampling dynamics criteria, demand an ET architecture, and tasks rate of progress are defined by the same tasks. Only the approach by Zhao and Zheng (1999) combines an ET architecture with a RM. Second, the majority of the RPA policies act on-line. In particular, FBS can act off-line or on-line depending on the specific approach, while all EDC approaches are on-line. Third, while the majority of FBS approaches consider both plant and kernel dynamics for selecting sampling periods, EDC approaches usually only account for plant dynamics. And finally, regarding the solutions, all EDC approaches provide time intervals that constantly vary while FBS approaches provide static or varying periods, depending on whether they act off-line or on-line.

It is worth mentioning that the scheduling strategies that should give support and enforce the diverse type of solutions provided by each method have not been analyzed under any classifying keyword. This obeys to two main reasons. First, while in the majority of the FBS approaches schedulability issues are accounted for and appropriated scheduling policies are adopted or developed, in the majority of EDC approaches the scheduling strategy is missing. Hence, classifying by scheduling support will result in many empty entries in Table 3. Second, as it is indicated in Sect. 2.4.2, the basic requirement for implementing all the RPA policies is to have a real-time kernel including flexible features that allow re-setting tasks periods at run-time. Therefore, the actual scheduling strategy (EDF, FPS, server-based approaches, resource reservations, etc) in not making a big difference in terms of resource and performance beyond the well known properties of each strategy (see for example Buttazzo 2005 for an analysis of EDF vs Rate Monotonic). Therefore, the classification according to scheduling support has not been included. The same applies to the diverse tasks models reviewed in Sect. 2.2. The majority of the RPA policies do not give details on the actual implementation of control tasks. Therefore, classifying by tasks models will not give useful information. In addition, all the approaches can be implemented using several tasks models. Therefore, tasks models are not a classifying feature of the approach itself, although they impact on control performance, as indicated in Lozoya et al. (2008b).

By analyzing the taxonomy and looking at Table 3, the following specifications need to be considered in the definition of the PEF. The framework must support different triggering paradigms (event-driven, time-trigger), optimization algorithms (on-line, off-line), evaluation parameters (control performance, resource utilization), tasks models, and sampling patterns (static periods, varying periods, aperiodic time intervals, job's sequences). This is fulfilled by specifying a simulation platform and

Table 3 Taxonomy of RPA approaches

	Which	What & Who	When	Where	How
Seto et al. (1996)	Optimizat.	TT, RM	Off-line	None	Static periods
Seto et al. (1998b)	Optimizat.	TT, RM	Off-line	None	Static periods
Årzén (1999)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Heemels et al. (1999)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Zhao and Zheng (1999)	Bound dyn.	ET, RM	On-line	Plant (instant.)	Sequences
Eker et al. (2000)	Optimizat.	TT, RM	On-line	Kernel	Varying periods
Rehbinder and Sanfridson (2000)	Optimizat.	TT, RM	Off-line	None	Sequences
Åström and Bernhardsson (2002)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Velasco et al. (2003)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Henriksson et al. (2002)	Optimizat.	TT, RM	On-line	Kernel	Varying periods
Cervin et al. (2002)	Optimizat.	TT, RM	On-line	Kernel	Varying periods
Palopoli et al. (2002b)	Optimizat.	TT, RM	Off-line	Plant stability	Static periods
Chandra et al. (2003)	Optimizat.	TT, RM	Off-line	None	Static periods
Martí et al. (2004)	Optimizat.	TT, RM	On-line	Plant (instant.)	Varying periods
Henriksson and Cervin (2005)	Optimizat.	TT, RM	On-line	Plant (finite hor.)	Varying periods
Palopoli et al. (2005)	Optimizat.	TT, RM	Off-line	Plant stability	Static periods
Ben Gaid et al. (2006)	Optimizat.	TT, RM	On-line	Plant (finite hor.)	Sequences
Castañé et al. (2006)	Optimizat.	TT, RM	On-line	Plant (finite hor.)	Varying periods
Miskowicz (2006)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Tabuada (2007)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Lemmon et al. (2007)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Johannesson et al. (2007)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Suh et al. (2007)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Henningsson et al. (2008)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Anta and Tabuada (2008a)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Anta and Tabuada (2008b)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Bini and Cervin (2008)	Optimizat.	TT, RM	Off-line	None	Static periods
Heemels et al. (2008)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Wang and Lemmon (2008b)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Wang and Lemmon (2008a)	Bound dyn.	ET, Task	On-line	Plant stability	Time intervals
Anta and Tabuada (2009)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Ben Gaid et al. (2009)	Optimizat.	TT, RM	On-line	Plant (finite hor.)	Sequences
Martí et al. (2009b)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Martí et al. (2009a)	Optimizat.	TT, RM	On-line	Plant (instant.)	Varying periods
Mazo et al. (2009)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Mazo and Tabuada (2009)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Samii et al. (2009a)	Optimizat.	TT, RM	Off-line	Schedule	Sequences
Samii et al. (2009b)	Optimizat.	TT, RM	Off-line	Schedule	Sequences
Velasco et al. (2009b)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Wang and Lemmon (2009a)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Wang and Lemmon (2009b)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Anta and Tabuada (2010)	Bound dyn.	ET, Task	On-line	Plant (instant.)	Time intervals
Cervin et al. (2011)	Optimizat.	TT, RM	On-line	Plant (finite hor.)	Varying periods

an experimental platform considering different functional modules that can be tailored to cover all these specifications.

3.2 PEF overview

The PEF is composed by a simulation and an experimental platform. Details of the both platforms can be found in the research report (Lozoya et al. 2012) and the software components of both platforms are available at http://dcs.upc.es/uploads/projects/soft/Platform.zip.

3.2.1 Technology

Regarding the simulation tools and experimental platforms (see Sect. 2.4) available for the PEF, the following requirements have been placed. The simulation platform must allow co-simulation of (1) real-time control tasks executing on top of a real-time kernel and (2) plant's dynamics.

The simulation tool chosen as a basis of the simulation part of the PEF is the TrueTime toolbox integrated with Matlab/Simulink. TrueTime has been shown to be a well accepted simulation tool among the real-time and control community as demonstrated by the large number of publications presenting the simulator and its modifications, as well as for the large number of publications where TrueTime has been the tool for validating diverse theoretical results on control and real-time systems co-design.

For the experimental part of the PEF, the Erika real-time kernel running on top of a Full Flex board equipped with a Microchip dsPIC33 microcontroller has been chosen. Although being a relatively new kernel (released in its first form in 2003), it has an active development and support, and it has been shown to be a good platform for testing state-of-the-art research and educational results on embedded control systems (Martí et al. 2010).

Finally, it is important to stress that the surveyed results in Tables 1 and 2 also showed the diversity of controlled plants that have been used. For the selection of the plants for the simulation and experimental platforms of the PEF, a few factors were considered. First, controller design methods rely on the accuracy of the plant mathematical model. The more accurate the model, the better the observation of the effects of each controller on the plants. Hence, the plant was selected among those for which an accurate mathematical model could easily be derived. Plants such as an inverted pendulum or a direct current motor, which are the *defacto* plants for benchmark problems in control engineering, have a not trivial modeling and the resulting model is often not accurate. Second, it was desired to have a plant not requiring intermediate complex sensors and actuators in order to reduce again the modeling effort. Hence, being able to plug the plant directly into a micro-controller, and sensed directly using the ADC and controlled by a PWM would provide the required simplicity. Third, it was also desired to have a plant that can be easily reproduced, that is, it is easy and cheap to built. Regarding all this requirements, the election was an electronic circuit in the form of a double integrator, whose state-space model is given by

$$\begin{aligned}\dot{x} &= \begin{bmatrix} 0 & -21.2766 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ -21.2766 \end{bmatrix} u \\ y &= \begin{bmatrix} 1 & 0 \end{bmatrix} x \end{aligned} \tag{1}$$

where the state vector is $x = [v_1 \;\; v_2]^T$. The goal of the controller can be a tracking setting such as to make the circuit output voltage, v_1, to track a reference signal by giving the appropriate voltage levels (control signals) u, or a regulator setting. Both states v_1 and v_2 can be read via the ADC port of the micro-controller and u is applied to the plant through the PWM output.

3.2.2 Services of the performance evaluation framework

This section describes the common services that both the simulation part and the experimental part of the PEF have to provide. PEF services refer to the functional modules, each one performing specific activities. They have been conceptually defined for providing a flexible and scalable evaluation platform. Framework services are baseline configuration, resource manager, task controller, optimization method and performance measurement.

The *baseline configuration* module is responsible for providing an interface between the plants and the other functional models while defining the system model. The proposed PEF considers the case of a single processor with multitasking capabilities controlling several continuous plants. However, other models can be supported if a new configuration is defined within this module, e.g. multiple-processors controlling different plants. The module defines the connectivity between the processor and the plants, including analog lines, analog to digital converters, and digital to analog converters. In addition, it defines all the variables of the system model that must characterize each simulation, such as perturbation and/or noise models, different settings depending on the control objective (regulator or tracking), etc.

The *resource manager* module obtains the system dynamics information from the plants and kernel. Using this information, the resource manager executes the previously selected optimization algorithm from a repository of algorithm routines. Off-line optimization algorithms are executed just once during the system initialization process, meanwhile on-line optimization algorithms are executed periodically (mainly for FBS approaches) or aperiodically (for EDC approaches). The optimization module receives the system dynamics information from the resource manager, then the optimization procedure results are sent to the task controller to indicate new settings such as new values for task periods or new controller gains.

The *task controller* module is responsible to create task instances according to a specific task model from a set of available control task models definitions such as standard, one-sample, split, switching and one-shot (recall Sect. 2.2). Task instances support both periodic and aperiodic interruptions. Timing parameters for the task instances can also be modified on-line by the *optimization method* module. This module indicates to the baseline configuration module when the different activities within each closed loop operation must be executed, that is, when sampling, control algorithm computation, and actuation must take place.

The *performance measurement* module is able to obtain information from any of the other modules. Raw information such as algorithm execution time, plants' errors and plants' states are processed by this module in order to perform a complete assessment using proper metrics to evaluate control performance, resource utilization and computational overhead. For simulations, control performance for each plant is measured using a continuous standard quadratic cost function (Åström and Wittenmark 1997)

$$J_{i,control} = \int_0^{t_{eval}} \left[x^T(t)Qx(t) + u^T(t)Ru(t)\right]dt. \tag{2}$$

where Q and R represents the cost weighting matrices, t_{eval} the simulation period, and $x(t)$ and $u(t)$ are the plant state and control input for each i^{th}-plant. Although Q, R, t_{eval}, $x(t)$ and $u(t)$ should be specified with an i-subscript that identifies the plant/control loop, for the sake of clarity, the subscript has been omitted. The control performance achieved by the set of n control tasks is measured by

$$J_{control} = \sum_{i=1}^{n} J_{i,control}. \tag{3}$$

For experiments, control performance is measured using the corresponding discrete-time quadratic cost function

$$J^d_{i,control} = \sum_{k=0}^{t_{eval}} \left[x^T(k)Q_d x(k) + 2x^T(k)N_d u(k) + u^T(k)R_d u(k)\right], \tag{4}$$

where the Q_d, N_d and R_d represent the discrete cost weighting matrices. The derivation of (4) from (2) can be found in Åström and Wittenmark (1997). As before, the i-subscript that should be associated to Q_d, N_d, R_d, t_{eval}, $x(k)$ and $u(k)$ has been omitted for clarity purposes. And the control performance achieved by the set of n control tasks is measured by

$$J^d_{control} = \sum_{i=1}^{n} J^d_{i,control}. \tag{5}$$

Resource utilization is measured as a percentage of use of the processor during each evaluation period (t_{eval}),

$$J_{resource} = \left(\frac{1}{t_{eval}} \sum_{i=1}^{n} E_i\right) * 100, \tag{6}$$

where n is the number of tasks sharing the same processor, and E_i corresponds to the accumulated processor time of a specific control task during each simulation run or each experiment. Computational overhead is included in this parameter.

In order to have a condensed system-level metric that incorporates control performance and resource utilization in a single value, the two previous metrics have

Table 4 Selected methods of FBS and EDC showing key distinctive features

Approach	Tendency	Operation	Dynamics	Exec. Rule
Static approach (Åström and Wittenmark 1997)	standard	Off-line		
Off-line FBS (Seto et al. 1996)	FBS	Off-line		
On-line FBS-Inst.(Martí et al. 2004)	FBS	On-line	kernel/plant	
On-line FBS-FH (Henriksson and Cervin 2005)	FBS	On-line	kernel/plant	
Heuristic Self-triggered (Velasco et al. 2003)	EDC	On-line	kernel/plant	state/utilization
Self-triggered (Lemmon et al. 2007)	EDC	On-line	plant	meas. state
Optimal self-triggered (Martí et al. 2009b)	EDC	On-line	plant	meas. state/opt.

been used to create a new metric, named *performance index* (PI). For simulation, this metric is defined as

$$PI = J_{control} * J_{resource}, \tag{7}$$

while for experiments is defined as

$$PI^d = J^d_{control} * J_{resource}. \tag{8}$$

The PEF modules take different forms in the implementation of the simulation and experimental platforms. The modularity permits to assess different scenarios and accommodate the diverse RPA policies. Details can be found in Lozoya et al. (2012). The modular design of the PEF also implies that the framework is scalable. For example, if another task model appears in the literature, the task controller module can be updated to include it, or if another RPA policy must be reproduced it can be included in the repository of algorithm routines. Similarly, if other metrics must be analyzed such as verification complexity, the measurement module can be updated with the new metric.

3.3 Selected methods for performance evaluation

This section presents which subset of the RPA policies will be evaluated in Sect. 4 using the PEF. The subset of selected methods, listed in Table 4, represent major tendencies identified in the taxonomy as it is discussed next. Note that not all the entries are filled in because some approaches do not consider those item. For example, "execution rule" only applies to EDC approaches, or "Dynamics" only applies to these approaches that operate on-line.

3.3.1 Static approach

This is the only approach that does not belong to the class of FBS nor EDC methods but it is here included for comparative purposes. It implements the traditional approach to real-time implementation of computer controlled systems where each control task is assigned off-line a *specific* sampling period selected according to standard procedures (Åström and Wittenmark 1997).

3.3.2 Off-line FBS (feedback scheduling)

The off-line FBS is represented by the seminal work by Seto et al. (1996) on sampling period selection. Although different on the formulations of the optimization problem in terms of objective functions and restrictions, existing results such as Seto et al. (1998b), Rehbinder and Sanfridson (2000), Palopoli et al. (2002b, 2005), Chandra et al. (2003), Bini and Cervin (2008), Samii et al. (2009a, 2009b) can be included in a subset of works that share in common that they are off-line approaches where sampling periods are derived before run-time and kept constant during execution.

3.3.3 On-line FBS (feedback scheduling)

An advance in the state of the art with respect to off-line FBS is to re-assigning tasks periods at run-time. Initial work presented by Eker et al. (2000) and further developed in Henriksson et al. (2002) and Cervin et al. (2002) on-line adjust sampling periods considering the dynamics of the processor load. A further improvement, introduced by Martí et al. (2004), is to also optimize control performance by on-line adjusting sampling periods according also to the plant dynamics.

Within the existing work considering FBS with on-line period adjustment, two main flavors can be distinguished depending on whether decisions are taken looking at current plant states (instantaneous measure) or looking at predictions of the plant dynamics using for example a finite horizon cost function. The work by Martí et al. (2004) (further refined in Martí et al. 2009a) will be taken as example policy whose decisions rely on an instantaneous metric. The work by Henriksson and Cervin (2005) (further developed in Castañé et al. 2006 and Cervin et al. 2011) will be taken as example of policy whose decisions rely on a finite horizon metric. This type of approach was also adopted in Ben Gaid et al. (2006, 2009).

3.3.4 Heuristic self-triggered (event-driven control)

This method is based on EDC and it is represented by the work by Velasco et al. (2003), where a self-triggered control approach is proposed to heuristically improve control performance considering the plant dynamics but also measures of the processor workload in terms of CPU utilization.

3.3.5 Self-triggered (event-driven control)

This method is based on event-driven control and it is represented by the work by Lemmon et al. (2007), where a self-triggered scheme is presented based on a robust control formulation. In this approach the execution rule is defined only as a function of the system state, and the controller gain is invariant and designed in the continuous time domain. This approach was further developed in Wang and Lemmon (2008a, 2008b, 2009a, 2009b). Similar event-driven control approaches that can be represented under this category are Tabuada and Wang (2006), Tabuada (2007), Anta and Tabuada (2008a, 2008b, 2009, 2010) Mazo et al. (2009), Mazo and Tabuada (2009), Velasco et al. (2009b). Note that some of them do not present the event-driven approach formulated as a self-triggered method. But the key aspect is that they can be transformed to a self-triggered setting, and more important, that sampling intervals are defined as a function of the system state. However, it is also clear that each method will provide different performance numbers in case of being evaluated.

3.3.6 Optimal self-triggered (event-driven control)

This method is based on event-driven control, and is an enhancement of Lemmon et al. (2007) that was presented by Martí et al. (2009b). The key aspect is that the different controller settings such as next sampling interval and controller gains are obtained after applying an optimization algorithm that is executed at each controller execution. Optimal parameters are selected according to the plant dynamics and a standard quadratic cost function.

3.3.7 Other EDC approaches

Apart from the previous selected methods for evaluation, there are still different event-driven approaches such as Årzén (1999), Heemels et al. (1999), Zhao and Zheng (1999), Åström and Bernhardsson (2002), Miskowicz (2006), Johannesson et al. (2007), Suh et al. (2007), Heemels et al. (2008), and Henningsson et al. (2008) that could be implemented. However, many of them rely on specific hardware to detect event conditions and their transformation into a self-triggered approach should be carefully analyzed.

4 Implementation and evaluation of selected methods

This section describes the implementation and evaluation of selected FBS and EDC methods using the PEF. The methods have been selected due to their representative characteristics within these two tendencies, as discussed in Sect. 3.3. The objective is to corroborate the flexibility of the PEF to accommodate different RPA policies, as well as to provide valuable analysis through the obtained results to discuss the key elements of each method and differences between them. Both control experiments and simulations are reported.

4.1 On the evaluation of RPA policies

Comparing RPA policies is not an easy task. The design and implementation of the PEF permits specifying particular settings for simulating and performing experiments for all the RPA policies. However, specifying settings in a fair manner for comparison purposes is not trivial, and several decisions have to be taken.

First of all, control performance and resource utilization are the main two evaluation metrics to be evaluated, as defined in Sect. 3.2.2. And in general both metrics are in conflict: a task with low resource demands, which is positive, implies control actions delivering low control performance, which is negative, and viceversa. To provide some sort of reference in the policies comparative analysis, in the evaluation by simulation the resource utilization of each task has been specified to be constant. This means that all methods present a similar CPU utilization (5 % approximately) which is achieved by specifying different period settings while leaving tasks computation times equal and constant to 0.8 ms for all methods. This permits to focus the attention to the control performance evaluation only. However, it is clear that setting all tasks computation times equal and constant is not a realistic setting. But at simulation time, it is impossible to know the exact numbers. Although they could be approximated by looking at the task code complexity, it seems more interesting to fix computation times, and to be able to observe how the variation on the tasks periods affect control performance. A complementary simulation approach could have been to assess the RPA approaches using stochastic computation times. However, since the compared algorithms are intrinsically deterministic, such study would have induced to misleading conclusions. In the experimental evaluation, tasks computation times become very important because they reflect the real resource demands of each method, including the overhead that they introduce. Hence, for the experimental evaluation, the goal has been to tune all methods to provide similar control performance in order to be able to observe the resource demands of each method. In both cases, the unifying performance index PI indicates the goodness of each method both in simulation and in the experiments.

A related problem is to determine how well RPA policies adapt to resource variability. As mentioned in Sect. 3.1.2, the majority of FBS approaches consider both plant and kernel dynamics for selecting sampling periods while EDC approaches usually only account for plant dynamics. Therefore, the evaluation of RPA policies with respect to their ability to adapt to resource availability would have no sense regarding EDC approaches. Therefore, the available portion of CPU for the set of concurrent tasks has been kept constant for all simulations and experiments. A specific subtopic of resource variability would also be to analyze how each policy deals with overload situations. Noting that none of the evaluated policies explicitly specifies the way that they treat these situations, evaluating these scenarios seems to be out of the main goals of the paper. However, the importance of the topic suggests that this could be treated in future work.

Another key point refers to the control goals that have served as a main design force in each method. For example, for the selection of the tasks periods, the FBS approach by Henriksson and Cervin (2005) uses an optimal control setting while the EDC approach by Lemmon et al. (2007) uses a robust control setting. In other workds, both are optimal, but with respect to different goals (cost functions). Therefore, comparing them using a single control performance metric such as the quadratic cost function given in (3) or (5) is not fair because both of them are the type of cost functions used in optimal control. Hence, the FBS approach by Henriksson and Cervin (2005) starts with advantage only due to the selected evaluation metric. To avoid this kind of unfair scenarios, all evaluated RPA policies have been "adapted" to compensate for possible a priori disadvantages, which loosely speaking means trying to make them as optimal as possible with respect to the evaluation metric (further details in the following sections).

Regarding noise, a similar problem could be faced. Some RPA approaches were designed explicitly to cope with noise while others were not. For the selected methods to be evaluated, none of them considers noise as a main design goal. Therefore, in the simulations no noise is injected in the plants, and in the experimental set-up the inherent plant and measurement noise is equal for all the evaluated methods. In the same context, there are RPA policies that treat perturbations explicitly and with different assumptions, while other approaches consider no perturbations. For the selected methods, since perturbations have not been explicitly addressed, the simulation and the experimentation only considers reference set-point changes. However, in the case of evaluating approaches that consider noise and perturbations, the simulation and experimental settings that have been used would not be fair.

In addition, as just mentioned before, reference set-point changes have been specified to each control loop using a square wave reference signal of constant amplitude. The time at which each set-point changes occurs is randomly generated, but enforcing in average that a set-point change has to occur every 3 s. Again, this specification is critical and may put in advantage or disadvantage specific policies. For example those RPA policies that do not update control tasks periods constantly may provide very poor control performance if set-point changes occur too often. However, this penalty could be compensated by the fact that they may incur in less overhead than those policies that constantly adapt tasks periods. Therefore, establishing the frequency of the set-point changes is not an easy task.

It is also important to stress that overhead is not considered in the simulations, where the main goal is to assess control performance. However, in the experiments, overhead is included in the resource performance metric (6) either by noting that (i) the measured control tasks computation times increase when their code complexity increases (which in general is the case of EDC approaches) or (ii) an additional task acting as a RM has been included (which in general in the case of on-line FBS approaches). In order to avoid complexity and to focus on the evaluation, no specific overhead analysis is performed, although the provided experimental data would easily permit to extract overhead performance numbers.

Regarding tasks models, and noting that the majority of the policies do not specify the task model to be used (recall the discussion in Sect. 3.1.2 on this topic), for all simulations and experiments, all the RPA policies use the one-shot task model in the implementation. The one-shot has been selected because it adapts well to all kind of policies (permits periodic and aperiodic sampling), and also because it eliminates the degrading effects that sampling-to-actuation delays could introduce in the evaluation data. As demonstrated by Lozoya et al. (2008a), the degradation caused by jitters (sampling and delay variations) may hide the true performance that can be achieved by the different FBS and EDC methods. In addition, removing the effects of delays permits focusing the control performance evaluation in terms of variations in sampling periods, which is the main evaluation goal.

After having discussed some of the basic decisions taken in the specification of the simulation and experimental settings, next sections will further explain how these settings have been achieved by each evaluated RPA policy. In both platforms, the system model is the same: three double integrators are being controlled by three control tasks, which are executed on the real-time kernel processor (Truetime kernel in simulation, ERIKA kernel in the experimental setup).[2] Each task main goal is to track the square

[2]Details of the prototype and performance demonstrator that includes three plants in the form of double integrator electronic circuits can be found in Velasco et al. (2010).

Table 5 Simulation tasks sampling periods (seconds)

Approach	Type	Task 1	Task 2	Task 3
Static (Åström and Wittenmark 1997)		0.0300	0.0900	0.0900
Off-line FBS (Seto et al. 1996)	FBS	0.0500	0.0500	0.0500
On-line FBS-Inst. (Martí et al. 2004)	FBS	0.0300	0.0900	0.0900
On-line FBS-FH (Henriksson and Cervin 2005)	FBS	0.0300	0.0900	0.0900
Heuristic self-triggered (Velasco et al. 2003)	EDC	0.0812	0.0812	0.0812
Self-triggered (Lemmon et al. 2007)	EDC	0.0555	0.0555	0.0555
Optimal Self-triggered (Martí et al. 2009b)	EDC	0.0576	0.0576	0.0576

wave reference signal (a standard tracking configuration is applied (Åström and Wittenmark 1997), where the reference signal is transformed into a reference state and no need for feedforward signal to eliminate steady-state errors is required because the plant already includes integrators). And each simulation or experimentation run lasts 20 s and EDF has been the scheduling policy used for all RPA approaches (with deadline equal to period). To avoid having set-point changes at the same time instant for the three tasks at the beginning of simulations, each task had an initial offset of 0.1 s, 0.5 s and 1 s, while in the experiments, tasks execute without initial offsets.

4.2 Simulation

4.2.1 Tasks settings for each method

A detailed description of the settings applied to each method, including algorithm details or gain details, is presented next. In addition, Table 5 reports the task periods for FBS methods and average task periods for the EDC that provide a 5 % of CPU load, approximately (as explained in Sect. 4.1). Note that achieving exactly the same CPU load is not possible given all the constraints that have to be fulfilled and the specific operation of each approach.

– Static approach: periods for each task are heuristically selected and the corresponding discrete-time controllers designed before run-time using LQ optimal controller design using the cost function specified in (3) with Q and R being the identity and $t_{eval} = \infty$.
– Off-line FBS (Seto et al. 1996): since the three plants are equal, the off-line optimization procedure mandates to execute each task with the same period obtained using an off-line optimization procedure. Once periods are set, control tasks are scheduled under EDF.
– On-line FBS-Inst. (Martí et al. 2004): the final outcome of the method mandates to consider at run-time only two periods. Tasks (and controller' gains) switch between these two periods whenever the plant with highest error changes. Table 5 shows the sampling period values considering that task 1 has the largest instantaneous error. Each task can apply two discrete-time controller gains designed using LQ control considering the cost function specified as before for the two possible periods.

- On-line FBS-FH (Henriksson and Cervin 2005): this method mandates to switch periods at run-time continuously within the specified range according to the optimization procedure. Switches of tasks periods (and controllers' gains) occur at a given periodicity, called the period of the feedback scheduler $T_{fbs} = 500$ ms. Table 5 shows the sampling period values considering that task 1 has the largest finite-horizon error. Controller gains are an output of the optimization procedure, which give LQ optimal controllers considering the cost function specified as before.
- Heuristic self-triggered (Velasco et al. 2003): in this event-driven method the desired sampling period h_{k+1} for the next task instance execution is heuristically specified as $h_{k+1} = (h_{\max} - h_{\min})\, e^{-K|x_l|} + h_{\min}$, where $|x_k|$ is the norm of the state variables, K determines how abrupt are the changes in the sampling period (for this implementation $K = 4$), and $h_{\min} = 0.030$ s and $h_{\max} = 0.090$ s defines the sampling range. Table 5 shows the expected average sampling period. The controller gain is calculated on-line (each sampling period) using LQ optimal controller design using the cost function specified as before.
- Self-triggered (Lemmon et al. 2007): in this event-driven method, the execution rule that triggers a control task has been defined as a function of the measured state as $e_k(t)^T M e_k(t) = \eta x_k^T M x_k$, where $0 < \eta \leq 1$ specifies the relative size of the boundaries, and M defines the shape of the boundaries. Robust control techniques are used to define the boundary thresholds to ensure stability. Table 5 shows the expected average sampling period. In an event-based system η can be used to adjust the processor load. In this case $\eta = 0.6$, and it was selected to provide a similar processor load as the FBS approaches. Event-driven control methods cannot use the same LQ optimal techniques because no periodic sampling occurs. Hence, in order to provide a fair performance evaluation, the controller gain, which is kept constant, was designed using an off-line numerical algorithm that given η and the boundary shape M, provided the continuous gain that gave the minimum control cost measured in terms of the continuous cost function used as evaluation metric.
- Optimal self-triggered (Martí et al. 2009b): in this event-based method, the execution rule is also defined as a function of the measured state like in the previous approach. The main difference is that (i) the boundary M is a variable to be optimized with respect to the evaluation metric (and obtained off-line using an iterative algorithm for the given η, and according to the plant dynamics and the cost function) and (ii) the controller gain is updated at run time according to the sampling interval that applies and is designed in the discrete-time domain being LQ optimal with respect to the discrete cost function used as evaluation metric. Table 5 shows the expected average sampling period. For this approach it was set $\eta = 0.65$ in order to provide a similar processor load than the case of the FBS approaches.

4.2.2 *Simulation results*

Table 6 presents the control performance, the resource utilization and PI numbers for each simulated FBS and EDC method. Recall that the simulations settings have been selected to give the same resource utilization whenever possible. However, it

Table 6 Control performance and resource utilization simulation results

Approach	Control Performance $J_{control}$	Resource Utilization $J_{resource}$	Performance Index *PI*
Static (Åström and Wittenmark 1997)	1.9905	4.3720	8.7026
Off-line FBS (Seto et al. 1996)	1.4644	4.6720	6.8415
On-line FBS-Inst.(Martí et al. 2004)	1.3286	4.5320	6.0211
On-line FBS-FH (Henriksson and Cervin 2005)	1.3288	4.5680	6.0700
Heuristic self-triggered (Velasco et al. 2003)	1.3636	2.9560	4.0309
Self-triggered (Lemmon et al. 2007)	1.3469	4.3240	5.8238
Optimal Self-triggered (Martí et al. 2009b)	1.3498	4.0840	5.5127

can be observed that the heuristic self-triggered method presents a resource utilization clearly lower than the others. This is due to the fact that with the possible range of sampling periods available to all evaluated methods, the particular settings of the heuristic self-triggered prevent to achieve the same resource utilization. In other words, this approach has a clear tendency to reduce resource utilization.

Analyzing the control performance method by method, it is observed that the static method provides the worst performance, as expected. Then considering only FBS methods, the on-line algorithms provide the best numbers. This result was also expected: a plant not affected by a set-point change may require less attention by the control task (longer sampling period) than the case where it is affected by a set-point change. These numbers confirm this idea and they suggest that on-line adaptation is a desirable feature.

The three EDC methods have a similar control performance. However, even knowing that the tasks resource utilization settings are not realistic, the resource utilization numbers can be viewed as preliminary tendencies to be confirmed in the experiments, where real computation times are considered. Then, considering resource utilization, and looking at PI numbers, EDC methods do not provide the same numbers, and the heuristic self-triggered can be identified by achieving the best performance.

Comparing on-line FBS versus EDC methods, it can be noticed that FBS approaches have slightly better control performance. However, in terms of resources, EDC approaches seem to be more efficient in reducing resource utilization. This tendency will be further analyzed with exact numbers in the experimental results analysis.

It is important to highlight that these results are not intended to identify "winners" and "losers" because not always one method will perform better than other. Depending on the specific simulation set-up, these numbers slightly vary. In addition, FBS and EDC have similar goals but using different strategies. The problem to be solved by FBS approaches includes to fully utilize the available CPU portion dedicated to control tasks, while in EDC formulations, CPU utilization is not even considered in the problem formulation. Therefore, the selection of the simulation settings may also introduce some bias when trying to compare them. In any case, these results can be taken as an indicator of the potential of each particular approach and to provide valid

Table 7 Experimental tasks sampling periods (seconds)

Approach	Type	Task 1	Task 2	Task 3	η
Static (Åström and Wittenmark 1997)		0.0300	0.0900	0.0900	
Off-line FBS (Seto et al. 1996)	FBS	0.0500	0.0500	0.0500	
On-line FBS-Inst. (Martí et al. 2004)	FBS	0.0300	0.0900	0.0900	
On-line FBS-FH (Henriksson and Cervin 2005)	FBS	0.0300	0.0900	0.0900	
Heuristic self-triggered (Velasco et al. 2003)	EDC	0.0840	0.0840	0.0840	
Self-triggered (Lemmon et al. 2007)	EDC	0.0589	0.0589	0.0589	0.45
Optimal Self-triggered (Martí et al. 2009b)	EDC	0.0591	0.0591	0.0591	0.50

information to discuss the benefits and drawbacks of each tendency (FBS and EDC) under different circumstances.

4.3 Experiments

4.3.1 Tasks settings for each method and controller execution patterns

The task periods for FBS methods are the same as the one used during simulations. However, for the EDC approaches some settings change to provide a more accurate experimentation. The η values selected in the experiments were chosen in order to obtain similar control performance compared with the FBS on-line methods. This settings will strengthen the resource utilization differences between FBS and EDC approaches. The specific settings are summarized in Table 7. And the plant response and execution patterns during an interval of 5 s for the FBS and EDC methods are shown in Figs. 1, 2, 3, 4, 5, and 6, and explained next. For each figure, the top sub-figure shows the type of plant dynamics affected by a set-point change, and the bottom sub-figure shows in the x-axis time (in seconds), and the y-axis the sampling period that applies represented by a vertical line (in seconds), whose height indicates the next job release time.

- Static approach: tasks' periods are heuristically selected. No optimization process, either on-line or off-line, is executed. The real measured execution time of each control job, which includes sampling, control algorithm computation implemented using the one-shot task model, and actuation, is 0.214 ms.
- Off-line FBS (Seto et al. 1996): tasks' periods remain constant during the complete experiment as illustrated in Fig. 1. The time spent by the micro-controller for each control job is also 0.214 ms and that task's activation periods are always fixed.
- On-line FBS-Inst. (Martí et al. 2004): apart from the 3 control tasks whose initial periods are shown in Table 7, an additional resource manager task is executed every 500 ms to obtain the current error from the three plants and reassign task periods and controller gains if necessary. The execution time for each control job is still 0.214 ms and for each job of the additional task is 0.039 ms. Note that each job of the additional task only has to identify the plant with highest error and establish task periods and gains accordingly (recall Sect. 4.2.1). Therefore its computation time is much shorter than control jobs computation times. Figure 2 illustrates for

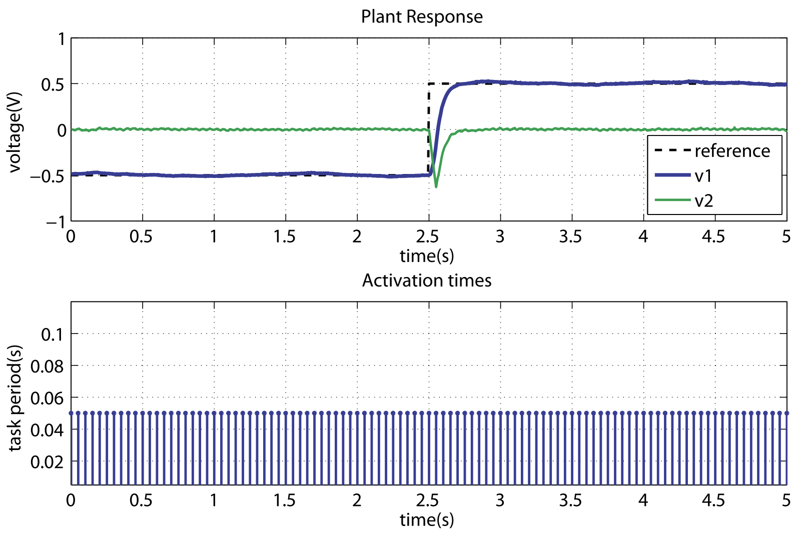


Fig. 1 Off-line FBS plant response and activation times

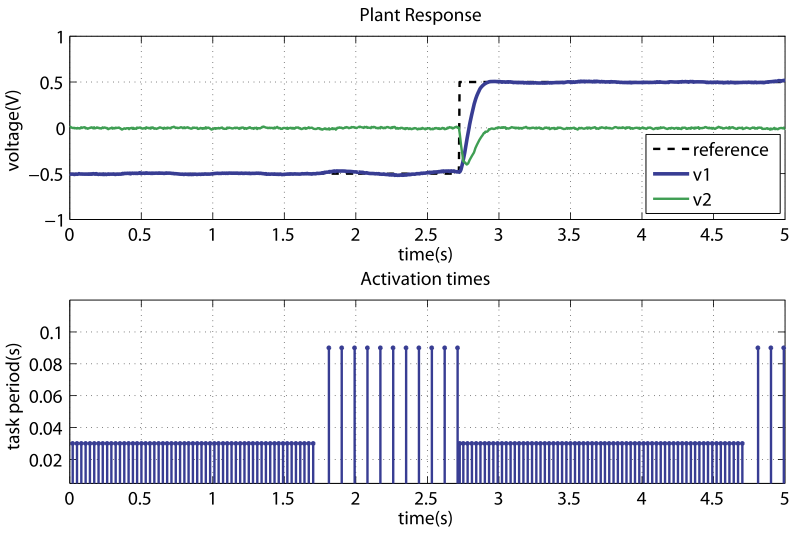


Fig. 2 On-line FBS-Inst. plant response and activation times

one controller that periods switch between two values, and shorter values start at each set-point change (when the transient error is high).

– On-line FBS-FH (Henriksson and Cervin 2005): apart from the 3 control tasks whose initial periods are shown in Table 7, and additional task doing the optimization is also executed periodically every 500 ms. Each job of the additional task obtains the finite-horizon error from the three plants and then selects appropriate periods for each task according to the optimization procedure. The execution time

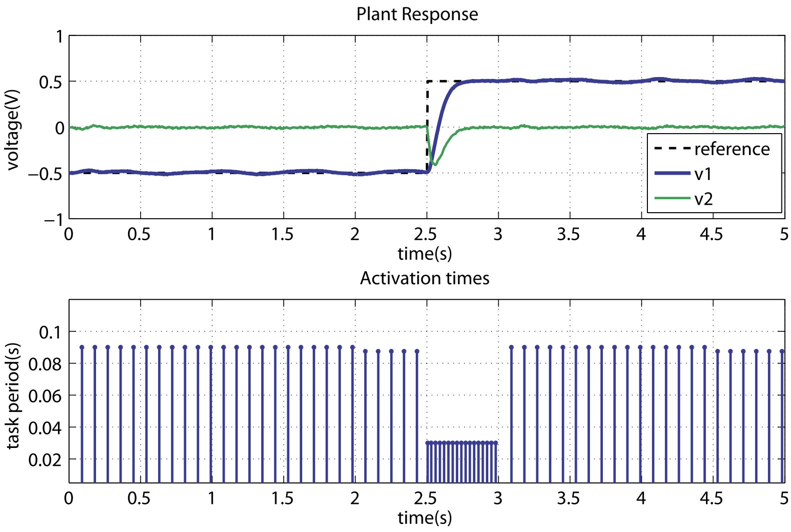


Fig. 3 On-line FBS-FH plant response and activation times

for each control job is the same as before, 0.214 ms, meanwhile the time spent by the micro-controller in each optimization job of the additional task is 1.459 ms. Note that this execution time is very high compared to the control task execution, and also compared to the additional task execution time of the previous approach. The main reason is that in this case the optimization process executed at each job of the additional task has to solve on-line an optimal control problem (recall Sect. 4.2.1). To shorten the long computation time of each job of the additional task, an alternative strategy can be applied. In fact, this time can be reduced to 0.097 ms when look-up tables are used to simplify the on-line computations performed by the additional optimization task. This requires storing in a table task periods and controller gain values. As illustrated in Fig. 3, task periods are smaller when there is a change in the set-point. Then periods change to higher values once the plant reaches the set-point. Although difficult to appreciate in the figure, tasks periods adopt different values (more than two).

– Heuristic self-triggered (Velasco et al. 2003): the three control task are configured as aperiodic, as opposite to the previous FBS methods. Each task defines its own next activation time by using the heuristic explained in Sect. 4.2.1. For each control task, once h_{k+1} is set, its corresponding LQ controller gain is selected. The execution time of each control job is 0.256 ms which is a few milliseconds higher than in the FBS methods. Figure 4 shows that since h_{k+1} depends on the norm of the state variables, for large errors small task periods are obtained. The sampling period range is given by $h_{\min} = 0.030$ s and $h_{\max} = 0.090$ s.
– Self-triggered (Lemmon et al. 2007): considering the three control tasks, η is selected for them to obtain a similar control performance than the FBS on-line methods for a given boundary M and the computed continuous gain. And each one of the three non-periodic control tasks calculates its next activation time from the measured state using the event condition described in Sect. 4.2.1. The next ac-

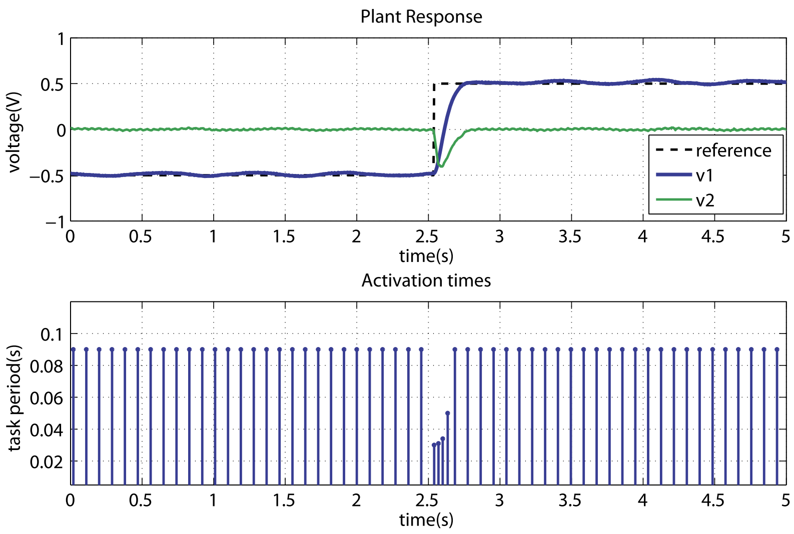


Fig. 4 Heuristic self-triggered plant response and activation times

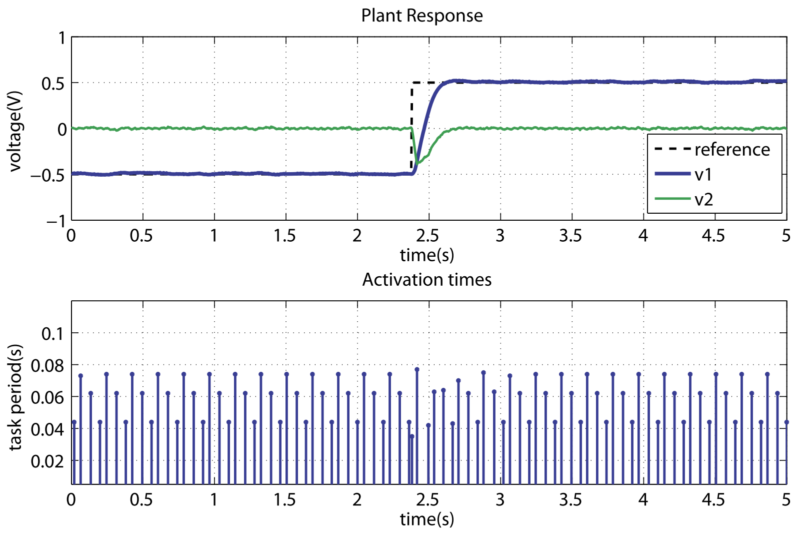


Fig. 5 Self-triggered plant response and activation times

tivation time routine can be implemented by using pre-computed values of the explicit solution given by Velasco et al. (2008a) or by on-line computing the solution presented in Velasco et al. (2010). The pre-computed values solution spends less processor time (0.225 ms) in comparison with the on-line computation solution (0.419 ms). However the pre-computed solution is less flexible since it only supports fixed magnitude set-point changes. As illustrated in Fig. 5, the activation times pattern of this self-triggered approach has an oscillating behavior.

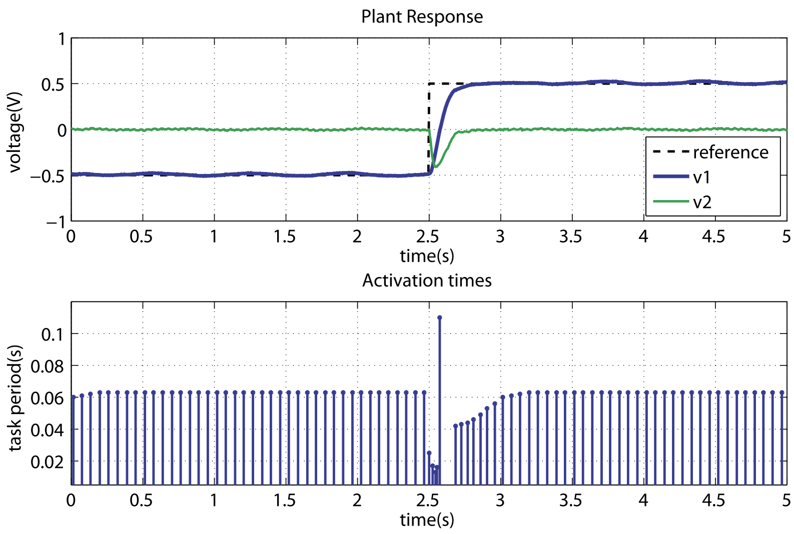


Fig. 6 Optimal self-triggered plant response and activation times

– Optimal self-triggered (Martí et al. 2009b): as in the previous method η is selected for the three tasks to obtain a similar control performance than the on-line FBS methods and the boundary shape M is off-line selected to be optimal in the LQ sense. As in the Self-triggered approach, the next activation time routine (that also returns the discrete LQ gain) can be implemented by using pre-computed values or by computing on-line the solution presented in Velasco et al. (2010). The first one has an execution time of 0.225 ms for each job while the second spends 0.419 ms. Figure 6 shows an activation times pattern that starts with small periods when a set-point change occurs. Then one large period is introduced, and later on the periods are lineally increasing. The long period may be explained as follows: the controller identifies that the state is moving towards the correct set-point and that it will take "time" to reach the set-point. Therefore, the next task activation time should occur quite late compare to other activations.

4.3.2 Experimental results

This section summarizes the experimental results obtained from the evaluation of the different methods. Table 8 presents the control performance and the resource utilization results for each FBS and EDC method.

By comparing simulation results (Table 6) with the experiment results (Table 8), it can be noticed that control performance results are similar (same scale). However resource utilization for simulation is higher compared with the experiment results (more than two times higher in most of the cases). This is because in simulation a constant and not realistic enough execution time value was assumed (0.8 ms) meanwhile for experiments the real execution time spent by the micro-controller is measured for both the set of 3 control tasks and the additional task for the case of FBS approaches. Therefore, resource utilization results for the experiments reflect the real load in the

Table 8 Control performance and resource utilization experimental results

Approach	Calculation	Control Performance $J^d_{control}$	Resource Utilization $J_{resource}$	Performance Index PI^d
Static (Åström and Wittenmark 1997)		2.3090	1.1888	2.7449
Off-line FBS (Seto et al. 1996)		1.8331	1.2840	2.3537
On-line FBS-Inst. (Martí et al. 2004)	On-line	1.4699	1.1951	1.7567
On-line FBS-FH (Henriksson and Cervin 2005)	On-line	1.4919	1.6254	2.4249
On-line FBS-FH (Henriksson and Cervin 2005)	Look-up table	1.4919	1.2168	1.8153
Heuristic self-triggered (Velasco et al. 2003)	On-line	1.5931	0.9153	1.4583
Self-triggered (Lemmon et al. 2007)	On-line	1.6271	2.1858	3.5563
Self-triggered (Lemmon et al. 2007)	Pre-computed	1.6271	1.1464	1.8652
Optimal Self-triggered (Martí et al. 2009b)	On-line	1.5558	2.1772	3.3872
Optimal Self-triggered (Martí et al. 2009b)	Pre-computed	1.5558	1.1419	1.7765

micro-controller. And then, the PI permits to obtain a global view of the achieved performance of each method.

Analyzing the control performance method by method, it is observed that the static method again provides the worst control performance. The on-line FBS methods provides the best control performance, while the EDC methods lie in the middle between the on-line FBS and the off-line FBS performances. On-line FBS obtain the best control performance because they are designed to fully exploit the available resources to optimize control performance. In addition, the additional task is aware of the rate of progress of the three control tasks and therefore it can be more accurate at detecting needs and providing resources when required.

Now let's analyze the resource utilization results. By considering the Static approach as a reference (1.1888), it can be noticed that the On-line FBS-Inst. has a similar resource utilization (1.1951), meaning that the execution of the on-line optimization algorithm is efficient in terms of processor time. However in the On-line FBS-FH the on-line algorithm increases the utilization considerably due the complexity of the algorithm (1.6254). To solve this problem a look-up table can be used to reduce the processor load (1.2168). Using this strategy, all the FBS approaches have a similar processor load compared to the static case while delivering better control performance than the static approach. In particular, both on-line FBS methods deliver then the best control performance.

Although both on-line FBS approaches have a resource manager task that executes every 500 ms, the switching behavior is quite different, as it can be seen in Figs. 2 and 3. First of all, in the on-line FBS-Inst, control tasks can only switch between two periods while in the on-line FBS-FH the range of periods is richer. This can be observed in Fig. 3, between times 2 s and 2.5, where the period is not exactly 0.09 s, as it is in the on-line FBS-Inst approach. Secondly, the switching is different because of the phasing of the set-point changes acting in the three tasks. During the time interval shown in Fig. 2, the task is mainly executing with a period of 0.03 s. It started with a period of 0.03 s while the others had 0.09 s. At 1.7 s approximately, another task was affected by a set-point changed and required to switch to 0.03 s. The effect was that the displayed task had to switch to 0.09 s until a set-point occurring at 2.7 s approximately forced again another switching of periods. This sequence of set-point changes is different for the case shown in Fig. 3, which explains the difference in period switches.

Analyzing EDC approaches, the following observations can be made. The Heuristic self-triggered approach has the best overall resource utilization (0.9153) since less task jobs are triggered compared with the FBS approaches, and because the computation of the next activation time is relatively simple compared to the self-triggered approaches, which confirms the preliminary conclusions identified in the simulations. The Self-triggered and the Optimal Self-triggered have higher processor load if the next activation time value is computed on-line (2.1858 and 2.1772 respectively). Therefore pre-computed values have to be used in order to improve the processor load (1.1464 and 1.1419 respectively) obtaining better results compared to the Static approach. And in this case, apart from demanding less CPU than the static approach, are capable of providing better control performance.

By looking at the PI index, and paying attention to those approaches that have a resource utilization lower than the static (thanks to the use of off-line strategies in the implementation), it can be seen that the worst approach is the static and the best is the heuristic self-triggered. The rest lie in the middle. However it is important to stress that those approaches that incur in considerable overhead perform worse than the static. This identifies a clear problem. Future on-line approaches have to offer light-weight algorithms or off-line strategies to alleviate the on-line operation. Otherwise, they implementation will not be feasible.

4.4 Discussion

The presented simulation and experimental results corroborate that the presented performance evaluation framework is flexible enough to accommodate different approaches and it permits to evaluate their implementation feasibility. The modularity of the PEF allows shaping simulations and experiments with the specific settings of each policy. Hence, each method can be reproduced in a common framework, which permits to evaluate using the same metrics its control performance and resource utilization. And the modularity of the PEF also allows to tune and/or adapt the specific settings of each method in such a way that comparisons can be fairly made. For example, even not all the original methods use the same task model or treat delays explicitly, a single task model that treats delays in a specific way can be used in all the methods to start with an evaluation scenario that is equal to all of them in terms of delays.

In addition, the evaluation of the different RPA methods reveals the following key aspects:

- FBS and EDC have the ability to improve control performance with respect to the static approach, which is the standard approach for real-time implementation of control loops. Therefore, advances in the state-of-the-art of FBS and EDC must be promoted because they meet the demands imposed by modern embedded control systems.

- On-line FBS methods outperform the off-line FBS one if the additional task in charge of solving the optimization procedure has a lower processor demand. Hence, for future FBS approaches, it is desirable to avoid policies that impose a high computational demand, or to define implementation strategies that simplify the implementation. In any case, redistributing processor time at run-time among control tasks according to changes in the plants and CPU workload is the track to follow. Off-line approaches are not reactive to these changes, and computing resources can be wasted or not fully exploited.
- In general, on-line FBS methods are capable of providing better control performance compared with the EDC approaches. A major reason for this result is that FBS approaches apply a theoretical approach that considers all control loops at the same time, in a coordinated fashion. However, in EDC approaches, each control task acts individually, it executes when required, regardless of the other tasks needs. This may create conflicts between tasks, and in the end it lowers the control performance improvement that could be achieved in a more coordinated approach, as in the case of FBS.
- In general, EDC methods in self-triggered form have a lower processor load compared to FBS approaches if look-up table strategies are adopted. In other words, the number of executed control jobs in EDC approaches is lower than in the FBS cases. A major drawback for EDC in self-triggered form is that the computation of the next activation time at each job execution can be too expensive in terms of resource utilization. Hence simple triggering conditions should be adopted. It is important to stress that, if rather than adopting a self-triggered implementation, specific hardware is used for triggering control jobs, EDC methods are the key for (processor-based) systems demanding a very low resource utilization.
- EDC methods are a promising approach for networked control systems where the resource limitation is the communication bandwidth. In the networked scenario, bandwidth consumption only maps to the number of executed jobs and it is not affected by the computation of the next activation time if self-triggered form is implemented (which is the critical point in processor-based control systems).
- The control performance and resource utilization gains of the evaluated approaches are qualitative in the sense that they do not have any practical significance. They establish tendencies and outline differences between methods. Then, given a particular real problem, these numbers will become quantitative, and the benefits/drawbacks should then be analyzed carefully. However, it is out of the scope of this paper to establish which are the implications of the performance gains for real applications.

5 Conclusions

This paper has analyzed the control performance and resource utilization trade-offs of current state-of-the-art approaches in the area of embedded control systems. An exhaustive literature survey has been performed, identifying two key tendencies: feedback scheduling and event-driven control. The survey has permitted to construct a taxonomy that allows classifying each method. It has also established which basic services are required for developing a performance evaluation framework able to accommodate the diversity of existing approaches and able to assess whether these approaches can be implemented in practice. The application of this framework to representative approaches has shown that on-line management of computing resources is the key for having embedded control systems capable of adapting to varying demands while delivering specific control performance with tunable resource utilization.

References

Anta A, Tabuada P (2008a) Self-triggered stabilization of homogeneous control systems. In: Proceedings of the American control conference

Anta A, Tabuada P (2008b) Space-time scaling laws for self-triggered control. In: 47th IEEE conference on decision and control. CDC 2008, pp 4420–4425

Anta A, Tabuada P (2009) Isochronous manifolds in self-triggered control. In: Proceedings of the 48th IEEE conference on decision and control held jointly with the 28th Chinese control conference. CDC/CCC 2009, pp 3194–3199

Anta A, Tabuada P (2010) To sample or not to sample: self-triggered control for nonlinear systems. IEEE Trans Autom Control 55(9):2030–2042

Artemis (2006) Strategic research agenda. Tech rep, Advanced research and technology for embedded intelligence and systems. http://www.artemisia-association.org/downloads/SRA_MARS_2006.pdf

Årzén KE (1999) A simple event-based PID controller. In: Preprints 14th world congress of IFAC, Beijing, PR China

Årzén KE (2005) Timing analysis and simulation tools for real-time control. In: Pettersson P, Yi W (eds) Formal modeling and analysis of timed systems. Lecture notes in computer science, vol 3829. Springer, Berlin, pp 142–143

Årzén KE, Cervin A (2005) Control and embedded computing: Survey of research directions. In: Proc 16th IFAC world congress, Prague, Czech Republic

Årzén KE, Cervin A Eker J, Sha L (2000) An introduction to control and scheduling co-design. In: Proceedings of the 39th IEEE conference on decision and control, vol 5, pp 4865–4870

Årzén KE, Blomdell A, Wittenmark B (2005) Laboratories and real-time computing. IEEE Control Syst Mag 25(1):30–34

Åström K, Bernhardsson B (2002) Comparison of Riemann and Lebesgue sampling for first order stochastic systems. In: Proceedings of the 41st IEEE conference on decision and control, vol 2, pp 2011–2016

Åström K, Wittenmark B (1997) Computer-controlled systems: theory and design. Prentice Hall, New York

Audsley NC, Burns A, Richardson MF, Wellings AJ (1994) Stress: a simulator for hard real-time systems. Softw Pract Exp 24(6):543–564

Ben Gaid M, Cela A, Hamam Y, Ionete C (2006) Optimal scheduling of control tasks with state feedback resource allocation. In: American control conference

Ben Gaid M, Cela A, Hamam Y (2009) Optimal real-time scheduling of control tasks with state feedback resource allocation. IEEE Trans Control Syst Technol 17(2):309–326

Bhattacharya R, Balas G (2004) Anytime control algorithm: model reduction approach. J Guid Control Dyn 27:767–776

Bini E, Cervin A (2008) Delay-aware period assignment in control systems. In: Proceedings of the 2008 real-time systems symposium, pp 291–300

Brandt SA, Banachowski S, Lin C, Bisson T (2003) Dynamic integrated scheduling of hard real-ti soft real-time and non-real-time processes. In: Proceedings of the 24th IEEE international real-time systems symposium

Buttazzo GC (2004) Hard real-time computing systems: predictable scheduling algorithms and applications. Real-time systems. Springer, Berlin

Buttazzo G (2005) Rate monotonic vs edf: judgment day. J Real-Time Syst 29:5–26

Buttazzo G (2006) Research trends in real-time computing for embedded systems. SIGBED Rev 3:1–10

Buttazzo GC, Lipari G, Abeni L (1998) Elastic task model for adaptive rate control. In: Proceedings of the IEEE real-time systems symposium

Castañé R, Martí P, Velasco M, Cervin A (2006) Resource management for control tasks based on the transient dynamics of closed-loop systems. In: Proceedings of the 18th Euromicro conference on real-time systems, pp 171–182

Cervin A, Alriksson P (2006) Optimal on-line scheduling of multiple control tasks: A case study. In: Proceedings of the 18th Euromicro conference on real-time systems, Dresden, Germany

Cervin A, Eker J, Bernhardsson B, Årzén KE (2002) Feedback-feedforward scheduling of control tasks. Real-Time Syst 23:25–53

Cervin A, Henriksson D, Lincoln B, Eker J, Årzén KE (2003) How does control timing affect performance? Analysis and simulation of timing using Jitterbug and TrueTime. IEEE Control Syst Mag 23(3):16–30

Cervin A, Velasco M, Marti P, Camacho A (2011) Optimal online sampling period assignment: theory and experiments. IEEE Trans Control Syst Technol 19(4):902–910

Chandra R, Liu X, Sha L (2003) On the scheduling of flexible and reliable real-time control systems. Real-Time Syst 24:153–169

Eker J, Hagander P, Årzén KE (2000) A feedback scheduler for real-time controller tasks. Control Eng Pract 8(12):1369–1378

Eker J, Janneck J, Lee E, Liu J, Liu X, Ludvig J, Neuendorffer S, Sachs S, Xiong Y (2003) Taming heterogeneity—the ptolemy approach. Proc IEEE 91(1):127–144

Ellis P (1959) Extension of phase plane analysis to quantized systems. IRE Trans Autom Control 4(2):43–54

Erika (2011) Erika enterprise reference manual. Tech rep, evidence Srl. http://erika.tuxfamily.org/download/manuals/pdf/ee_refman_1_4_4.pdf

Fontanelli D, Palopoli L, Greco L (2011) Deterministic and stochastic qos provision for real-time control systems. In: 17th IEEE real-time and embedded technology and applications symposium, pp 103–112

Gai P, Abeni L, Giorgi M, Buttazzo G (2001) A new kernel approach for modular real-time systems development. In: Proceedings of the 13th Euromicro conference on real-time systems

Gupta V (2009) On an anytime algorithm for control. In: Proceedings of the 48th IEEE Conference on decision and control, held jointly with the 28th Chinese control conference. CDC/CCC 2009, pp 6218–6223

Heemels WPMH, Gorter RJA, van Zijl A, van den Bosch PPJ, Weiland S, Hendrix WHA, Vonder MR (1999) Asynchronous measurement and control: a case study on motor synchronization. Control Eng Pract 7(12):1467–1482

Heemels WPMH, Sandee JH, Bosch P (2008) Analysis of event-driven controllers for linear systems. Int J Control 84(1)

Henningsson T, Johannesson E, Cervin A (2008) Sporadic event-based control of first-order linear stochastic systems. Automatica 44(11):2890–2895

Henriksson D, Cervin A (2005) Optimal on-line sampling period assignment for real-time control tasks based on plant state information. In: 44th IEEE Conference on decision and control, and European control conference. CDC-ECC'05, pp 4469–4474

Henriksson D, Cervin A, Åkesson J, Årzén KE (2002) Feedback scheduling of model predictive controllers. In: Proceedings of the eighth IEEE real-time and embedded technology and applications symposium (RTAS'02).

Johannesson E, Henningsson T, Cervin A (2007) Sporadic control of first-order linear stochastic systems. In: Proc 10th international conference on hybrid systems: computation and control, Pisa, Italy. Springer, Berlin

Lemmon M, Chantem T, Hu X, Zyskowski M (2007) On self-triggered full-information h-infinity controllers. In: Bemporad A, Bicchi A, Buttazzo G (eds) Hybrid systems: computation and control. Lecture notes in computer science, vol 4416. Springer, Berlin, pp 371–384

Liberzon D (2003) Switching in systems and control. Birkhäuser, Basel

Liu CL, Layland JW (1973) Scheduling algorithms for multiprogramming in a hard-real-time environment. J ACM 20:46–61

Lozoya C, Velasco M, Martí P (2007) A 10-year taxonomy on prior work on sampling period selection for resource-constrained real-time control systems. In: Work in progress 19th Euromicro conference on real-time systems

Lozoya C, Martí P, Velasco M, Fuertes JM (2008a) Control performance evaluation of selected methods of feedback scheduling of realtime control tasks. In: 17th IFAC world congress

Lozoya C, Velasco M, Martí P (2008b) The one-shot task model for robust real-time embedded control systems. IEEE Trans Ind Inform 4(3):164–174

Lozoya C, Martí P, Velasco M, Fuertes J (2012) Performance evaluation framework (pef) for real-time embedded control systems. Tech rep, research report ESAII-RR-12-02, Automatic Control Department, Technical University of Catalonia

Marau R, Leite P, Velasco M, Martí P, Almeida L, Pedreiras P, Fuertes J (2008) Performing flexible control on low-cost microcontrollers using a minimal real-time kernel. IEEE Trans Ind Inform 4(2):125–133

Martí P, Lin C, Brandt SA, Velasco M, Fuertes JM (2004) Optimal state feedback based resource allocation for resource-constrained control tasks. In: Proceedings of the 25th IEEE international real-time systems symposium, pp 161–172

Martí P, Lin C, Brandt SA, Velasco M, Fuertes JM (2009a) Draco: efficient resource management for resource-constrained control tasks. IEEE Trans Comput 58:90–105

Martí P, Velasco M, Bini E (2009b) The optimal boundary and regulator design problem for event-driven controllers. In: Majumdar R, Tabuada P (eds) Hybrid systems: computation and control. Lecture notes in computer science, vol 5469. Springer, Berlin, pp 441–444

Martí P, Velasco M, Fuertes J, Camacho A, Buttazzo G (2010) Design of an embedded control system laboratory experiment. IEEE Trans Ind Electron 57(10):3297–3307

Mazo M, Tabuada P (2009) Input-to-state stability of self-triggered control systems. In: Proceedings of the 48th IEEE conference on decision and control, held jointly with the 28th Chinese control conference. CDC/CCC 2009, pp 928–933

Mazo M, Ant A, Tabuada P (2009) On self-triggered control for linear systems: guarantees and complexity. In: 10th European control conference

Miskowicz M (2006) Send-on-delta concept: an event-based data reporting strategy. Sensors 6:49–63

Osek (2005) Osek/vdx: open systems and the corresponding interfaces for automotive electronics. Tech rep, ISO 17356. http://portal.osek-vdx.org/files/pdf/specs/os223.pdf

Palopoli L, Lipari G, Lamastra G, Abeni L, Bolognini G, Ancilotti P (2002a) An object-oriented tool for simulating distributed real-time control systems. Softw Pract Exp 32:907–932

Palopoli L, Pinello C, Sangiovanni Vincentelli A, Elghaoui L, Bicchi A (2002b) Synthesis of robust control systems under resource constraints. In: Tomlin C, Greenstreet M (eds) Hybrid systems: computation and control. Lecture notes in computer science, vol 2289. Springer, Berlin, pp 197–225

Palopoli L, Pinello C, Bicchi A, Sangiovanni-Vincentelli A (2005) Maximizing the stability radius of a set of systems under real-time scheduling constraints. IEEE Trans Autom Control 50(11):1790–1795

Peiro S, Masmano M, Ripoll I, Crespo A (2007) PaRTiKle OS, a replacement for the core of RTLinux-GPL. In: 9th real time Linux workshop

Posix (1996) Portable operating system interface (posix) standard. Tech rep, International Standard ISO/IEC 9945-1: (E) IEEE Std 1003.1

Posix (1998) Information technology—standardized application environment profile—posix realtime application support (aep). Tech rep, POSIX.13. IEEE Std. 1003.13-1998

Quagli A, Fontanelli D, Greco L, Palopoli L, Bicchi A (2009) Designing real-time embedded controllers using the anytime computing paradigm. In: IEEE conference on emerging technologies & factory automation. ETFA 2009, pp 1–8

Real J, Crespo A (2004) Mode change protocols for real-time systems: a survey and a new proposal. J Real-Time Syst 26:161–197

Redell O, El-khoury J, Törngren M (2004) The aida toolset for design and implementation analysis of distributed real-time control systems. Microprocess Microsyst 28(4):163–182

Rehbinder H, Sanfridson M (2000) Integration of off-line scheduling and optimal control. In: Proceedings of the 12th Euromicro conference on real-time systems, pp 137–143

Rivas MA, Harbour MG (2001) Marte os: an ada kernel for real-time embedded applications. In: Proceedings of the international conference on reliable software technologies, Ada-Europe-2001

Samii S, Cervin A, Eles P, Peng Z (2009a) Integrated scheduling and synthesis of control applications on distributed embedded systems. In: Proceedings of the conference on design, automation and test in Europe, pp 57–62

Samii S, Eles P, Peng Z, Cervin A (2009b) Quality-driven synthesis of embedded multi-mode control systems. In: Proceedings of the 46th annual design automation conference, pp 864–869

Sanz R, Årzén KE (2003) Trends in software and control. IEEE Control Syst 23(3):12–15

Seto D, Lehoczky J, Sha L, Shin K (1996) On task schedulability in real-time control systems. In: 17th IEEE real-time systems symposium, pp 13–21

Seto D, Krogh B, Sha L, Chutinan A (1998a) Dynamic control system upgrade using the simplex architecture. IEEE Control Syst 18(4):72–80

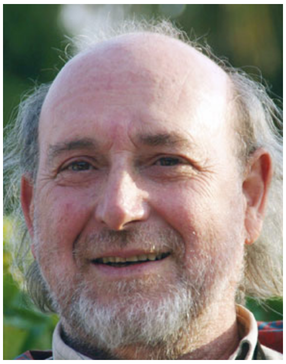
Seto D, Lehoczky JP, Sha L (1998b) Task period selection and schedulability in real-time systems. In: Proceedings of the IEEE real-time systems symposium, RTSS'98

Sha L, Abdelzaher T, Årzén KE, Cervin A, Baker T, Burns A, Buttazzo G, Caccamo M, Lehoczky J, Mok AK (2004) Real time scheduling theory: a historical perspective. Real-Time Syst 28:101–155
Stankovic JA (1996) Strategic directions in real-time and embedded systems. ACM Comput Surv 28:751–763
Storch M, Liu JS (1996) Drtss: a simulation framework for complex real-time systems. In: Real-time and embedded technology and applications symposium. IEEE Press, New York
Sucha P, Kutil M, Sojka M, Hanzalek Z (2006) Torsche scheduling toolbox for Matlab. In: Computer aided control system design, IEEE international conference on control applications, IEEE international symposium on intelligent control, pp 1181–1186. IEEE Press, New York
Suh YS, Nguyen VH, Ro YS (2007) Modified Kalman filter for networked monitoring systems employing a send-on-delta method. Automatica 43(2):332–338
Tabuada P (2007) Event-triggered real-time scheduling of stabilizing control tasks. IEEE Trans Autom Control 52(9):1680–1685
Tabuada P, Wang X (2006) Preliminary results on state-trigered scheduling of stabilizing control tasks. In: 45th IEEE conference on decision and control, pp 282–287
Törngren M, Henriksson D, Årzén KE, Cervin A, Hanzalek Z (2006) Tools supporting the co-design of control systems and their real-time implementation; current status and future directions. In: IEEE international symposium on computer-aided control systems design, Munich, Germany
Velasco M, Martí P, Fuertes J (2003) The self triggered task model for real-time control systems. In: Work-in-progress session of the 24th IEEE real-time systems symposium
Velasco M, Martí P, Bini E (2008a) Control-driven tasks: modeling and analysis. In: Proceedings of the real-time systems symposium, pp 280–290
Velasco M, Martí P, Lozoya C (2008b) On the timing of discrete events in event-driven control systems. In: Proceedings of the 11th international workshop on Hybrid Systems: computation and control, HSCC'08, pp 670–673
Velasco M, Martí P, Bini E (2009a) Equilibrium sampling interval sequences for event-driven controllers. In: European control conference
Velasco M, Martí P, Bini E (2009b) On Lyapunov sampling for event-driven controllers. In: Proceedings of the 48th IEEE conference on decision and control, held jointly with the 28th Chinese control conference. CDC/CCC 2009, pp 6238–6243
Velasco M, Martí P, Fuertes JM, Lozoya C, Brandt SA (2010) Experimental evaluation of slack management in real-time control systems: coordinated vs self-triggered approach. J Syst Archit 56:63–74
Wang X, Lemmon M (2008a) Event design in event-triggered feedback control systems. In: 47th IEEE conference on decision and control. CDC 2008, pp 2105–2110
Wang X, Lemmon MD (2008b) State based self-triggered feedback control systems with l2 stability. In: 17th IFAC world congress
Wang X, Lemmon M (2009a) Self-triggered feedback control systems with finite-gain l2 stability. IEEE Trans Autom Control 54(3):452–467
Wang X, Lemmon M (2009b) Self-triggered feedback systems with state-independent disturbances. In: American control conference. ACC'09, pp 3842–3847
Zhao QC, Zheng DZ (1999) Stable and real-time scheduling of a class of hybrid dynamic systems. Discrete Event Dyn Syst 9:45–64

Camilo Lozoya received the degree in electronics engineering in 1993 from the Chihuahua Institute of Technology, Chihuahua, Mexico, and the Ph.D. degree in automatic control in 2011 from the Technical University of Catalonia, Barcelona, Spain. Currently, he is 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.

Pau Martí 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 Mälardalen University, Västeras, 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.

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. 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.



Josep M. Fuertes received the degree in industrial engineering and the Ph.D. degree from the Technical University of Catalonia (UPC), Barcelona, in 1976 and 1986, respectively.
From 1975 to 1986, he was a Researcher at the Institut de Cibernética (Spanish Consejo Superior de Investigaciones Científicas). In 1987, he became a Permanent Professor with the UPC. In 1987, he got a position for a year at the Lawrence Berkeley Laboratory, Berkeley, CA, where he was a Visiting Scientific Fellow for the design of the Active Control System of the W.M. Keck 10-m segmented telescope (Hawaii). From 1992, he was the responsible of the University Research Line in Advanced Control Systems and later of the Distributed Control Systems Group. From 1996 to 2001, he acted as a Spanish Representative at the Council of the European Union Control Association. His research interests are in the areas of distributed, networked and real-time control systems and applications. Since 1989, he has been coordinating projects at the national and international levels related to the aforementioned areas of expertise. He is a member of the Administrative Committee of the IEEE Industrial Electronics Society, where he is chairing the Technical Committee of Networked Based Control Systems and Applications. He has collaborated as the Organizer, Chairman, Session Organizer, and member of program committees for several international conferences.

Enrix X. Martin received the degree in computer science and the Ph.D. degree on informatics from the Technical University of Catalonia, Barcelona, Spain, in 1995 and 2003, respectively. He was cofounder of Visoft Technologies in 2002. He worked in Augmented Reality software for GIS with Spanish railroad company ADIF. Since 2003, he has been an assistant professor in the Department of System Engineering at the Technical University of Catalonia. He is developing a specific architecture to control and interface a robotic rollator, iWalker. His research interests include embedded systems, robotics and computer technology.