SOFTWARE: PRACTICE AND EXPERIENCE
Softw. Pract. Exper. 2016; **46**:1181–1200
Published online 16 September 2015 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/spe.2373

Design and analysis of target-sensitive real-time systems

Giorgio Buttazzo*,†, Carmelo Di Franco and Mauro Marinoni

Scuola Superiore Sant'anna, Pisa, Italy

SUMMARY

A significant number of real-time control applications include computational activities where the results have to be delivered at precise instants, rather than within a deadline. The performance of such systems significantly degrades if outputs are generated before or after the desired target time. This work presents a general methodology that can be used to design and analyze target-sensitive applications in which the timing parameters of the computational activities are tightly coupled with the physical characteristics of the system to be controlled. For the sake of clarity, the proposed methodology is illustrated through a sample case study used to show how to derive and verify real-time constraints from the mission requirements. Software implementation issues necessary to map the computational activities into tasks running on a real-time kernel are also discussed to identify the kernel mechanisms necessary to enforce timing constraints and analyze the feasibility of the application. A set of experiments are finally presented with the purpose of validating the proposed methodology. Copyright © 2015 John Wiley & Sons, Ltd.

Received 31 October 2014; Revised 27 July 2015; Accepted 21 August 2015

KEY WORDS: target-sensitive applications; real-time computing; embedded systems

1. INTRODUCTION

Real-time systems are computing systems in which some computations have to be performed within precise bounded delays in order to guarantee a desired performance. Examples can be found in several application domains including avionics, automotive, manufacturing, plant control, health care, and transportation. Some of these systems are quite sensitive to timing issues in the sense that their overall performance is strictly related to the time at which outputs are produced.

Many theories have been developed in the literature to analyze the behavior of time-sensitive systems in which computational activities need to be executed *within* a specified deadline. This means that the corresponding task is allowed to execute anywhere in a time interval defined by its activation time and its deadline. In many cases, completing the task as soon as possible or just at its deadline does not cause significant performance difference, as for instance in sensory acquisition and control, video encoding and decoding, image processing, and many other activities. In such systems, a jitter in the output does not degrade the overall system performance significantly. In fact, as long as the average sampling rate is in the specified range, the jitter is filtered out by the inertia of the actuators (e.g., in robotics) or by the human vision system (e.g., in multimedia display).

In other applications, however, the time at which the output is produced (denoted as *target time*) strongly affects the overall system performance. For example, studies on human perception have shown that the human ear is capable of detecting a time jitter on the order of a few milliseconds, especially in rhythmic melodies and syncopated music [1, 2]. As a consequence, in computer-generated music, the notes produced by a program have to be played at precise target times, with a tolerance of a few milliseconds. To describe such a timing property, Dannenberg and Jameson [3]

*Correspondence to: Giorgio Buttazzo, Scuola Superiore Sant'anna, Pisa, Italy.
†E-mail: g.buttazzo@sssup.it

introduced a new task model where a deadline is not interpreted as the maximum time instant at which the execution must be completed, but as the best target time at which the output should be produced. In this case, the listener perceives a degradation in performance not only if a note is generated too late, but also if it is played too early with respect to the target time. Later, Brandt and Dannenberg [4] identified the most relevant mechanisms that should be implemented in a real-time kernel to support music software characterized by low latency.

Other target-sensitive applications where outputs have to be produced 'on time' with a specified tolerance are those where a moving object has to be tracked by a robot system [5–9]. In this situation, the maximum tolerated timing error depends on the object speed and size.

The schedulability analysis of real-time tasks that must be executed at precise instants has been investigated by several authors under different assumptions. For example, Jensen, Locke, and Tokuda [10] introduced the use of *utility functions* to evaluate the performance of time-sensitive applications and measure the quality of a produced result as a function of the task finishing time.

Chen and Muhlethaler [11] investigated how to schedule a set of value-based tasks under the criterion of maximizing the sum of tasks' contributions. Farzinvash and Kargahi [12] introduced the concept of instant value function to compute a value as a function of the instant at which a job is executed and proposed a scheduler that tries to maximize the total value accumulated by the task set.

To schedule overlapping events generated around the same target time, Guerra and Fohler [13, 14] presented an algorithm based on gravitational field. In their approach, every task is considered as a mass denoting a different importance level, free to oscillate on a pendulum hanged at the desired target time. Then, the schedule is derived by computing the mass distributions along the timeline resulting from the equilibrium condition. A Markov decision process has been adopted by Tidwell *et al.* [15] to derive optimal scheduling policies that maximize the cumulative value achieved by periodic tasks executing in stochastic non-preemptive intervals.

Although the papers cited in the previous text proposed new scheduling algorithms for target-sensitive task sets, no one addressed the problem of how to use such algorithms to provide a set of guidelines for the design and implementation of such systems. Buttazzo *et al.* [16] provided a preliminary overview of the implementation issues for a specific target-sensitive application and illustrated the problems that can occur when coding the software on a real-time operating system. This paper extends the work in [16] by considering a more precise and formal analysis, presenting a more general methodology, and validating the approach with additional experiments.

Contributions Although each of the previous papers provides a solution for a particular problem, a general methodology for designing and analyzing target-sensitive systems is still missing. The present work is aimed at integrating several contributions to address different aspects involved in the development of such type of systems, where some results must be delivered at predefined time instants, with a desired tolerance. In particular, this paper

1. shows how to derive the timing constraints of the computational tasks from the mission requirements and the features of the physical system;
2. presents different implementation approaches, discussing the weak and strong points of the different solutions; and
3. experimentally evaluates the behavior of the different solutions, showing how the performance guarantee can be predicted from the analysis.

For the sake of clarity, the proposed methodology is illustrated on a sample case study, used as a reference application to concretely illustrate the various steps.

Paper organization The rest of this paper is organized as follows. Section 2 presents the task model used to describe the real-time computational activities and summarizes the analysis techniques that can be used to verify the application schedulability; Section 3 describes the system taken as a reference to illustrate the proposed approach. Section 4 presents how to derive the timing constraints as a function of the mission requirements and system features; Section 5 discusses some implementation issues; Section 6 reports a set of experiments performed on the physical platform, and finally, Section 7 summarizes our conclusions.

DOI: 10.1002/spe

2. TASK MODEL AND BACKGROUND ANALYSIS

This section presents how to model and schedule the computational activities involved in a target-sensitive system. Then, the appropriate methods for performing the feasibility analysis of the application are presented under both fixed priority and deadline-based scheduling.

A generic real-time application can be modeled as a task set Γ of n periodic or sporadic real-time tasks, that is $\Gamma = \{\tau_1, \ldots, \tau_n\}$.

Each task τ_i is characterized by a release offset Φ_i, a worst-case execution time C_i, a relative deadline D_i, and a minimum interarrival time T_i, which is equivalent to the period if τ_i is periodic. The ratio $U_i = C_i / T_i$ represents the task utilization factor, and the total utilization of the task set is denoted as U:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}. \tag{1}$$

In this paper, it is assumed that all the periodic tasks are synchronously released at some predetermined time. Two possible scheduling algorithms are considered for managing the application tasks: a generic fixed priority scheduler and the earliest deadline first (EDF) algorithm [17].

Each task may include non-preemptive regions for guaranteeing the atomicity of certain operations, including critical sections for accessing globally shared resources. In particular, for each task τ_i, q_i denotes the length of the longest non-preemptive region of task τ_i.

The following sections summarize the schedulability tests that can be performed under fixed priority and EDF scheduling in the presence of non-preemptive regions.

2.1. Fixed priority analysis

The schedulability analysis of fixed priority periodic or sporadic tasks can be efficiently performed by the workload analysis [18]. This method is based on the concept of level-i workload $W_i(t)$, which represents the cumulative execution request generated by task τ_i and all higher-priority tasks over an interval of length t. Assuming tasks are ordered by decreasing priority, the level-i workload $W_i(t)$ can be expressed as

$$W_i(t) = C_i + \sum_{k=1}^{i-1} \left\lceil \frac{t}{T_k} \right\rceil C_k \leqslant t. \tag{2}$$

Then, the feasibility of a set Γ of n periodic or sporadic tasks can be verified through the following theorem.

Theorem 1 (Lehoczky–Sha–Ding, 1989)
A set Γ of n periodic tasks is schedulable by a fixed priority algorithm if and only if

$$\forall i = 1, \ldots, n \quad \exists t \in (0, D_i] \ : \ W_i(t) \leqslant t. \tag{3}$$

This result has been extended by Yao, Buttazzo, and Bertogna [19] to allow the usage of non-preemptive regions.

Theorem 2 (Yao–Buttazzo–Bertogna, 2011)
Let Γ be a set of n periodic tasks in which each task τ_i may include non-preemptive regions of maximum length q_i. Then, Γ is schedulable with a fixed priority algorithm if for all $\tau_i \in \Gamma$, there exists a time $t \in (0, D_i]$ such that

$$B_i + W_i(t) \leqslant t \tag{4}$$

where

$$B_i = \max_{k>i} \{q_k\}. \tag{5}$$

DOI: 10.1002/spe

Under fixed priority scheduling, a target-sensitive task can be guaranteed to execute at its activation time provided that it is assigned the highest priority level. In the presence of non-preemptive regions, however, the worst-case start time delay that task τ_1 can suffer is equal to the longest non-preemptive region of the lower priority tasks, that is, $B_1 = \max\{q_2, \ldots, q_n\}$.

2.2. EDF analysis

The schedulability analysis of periodic or sporadic tasks under EDF can be efficiently performed by the processor demand criterion [20]. This method is based on the concept of demand bound function $\text{dbf}(t)$, which represents the overall computation time of the jobs with a deadline no greater than t, that is,

$$\text{dbf}(t) = \sum_{i=1}^{n} \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor C_i \leqslant t. \tag{6}$$

Then, the feasibility of a set Γ of n periodic or sporadic tasks can be verified through the following theorem.

Theorem 3 (Baruah–Rosier–Howell, 1990)
A set Γ of n periodic tasks is schedulable by EDF if and only if $U < 1$ and

$$\forall t \in \mathcal{D} \quad \text{dbf}(t) \leqslant t, \tag{7}$$

where $\mathcal{D}$ denotes the set of all deadlines no greater than a certain value given by the minimum between the hyperperiod $H = lcm(T_1, \ldots, T_n)$ and the following bound

$$L_b = \max\left(D_{max}, \frac{\sum_{i=1}^{n}(T_i - D_i)U_i}{1 - U}\right)$$

where $D_{max} = \max\{D_1, \ldots, D_n\}$.

This result has been extended by Bertogna and Baruah [21] to allow the usage of non-preemptive regions. The blocking time introduced by non-preemptive regions is taken into account through a *blocking function* $Q(D)$, which returns the maximum amount of time for which a job with deadline D can execute non-preemptively without jeopardizing the schedulability of the task set. In particular, the blocking function $Q(D)$ is computed as follows:

$$\begin{cases} Q(D_1) = D_1 - C_1 \\ Q(D_k) = \min\left(Q(D_{k-1}),\ D_k - \text{dbf}(D_k)\right) \end{cases}$$

Then, the resulting schedulability test is stated by the following theorem.

Theorem 4 (Bertogna–Baruah, 2010)
Let Γ be a set of n periodic tasks in which each task τ_i may include non-preemptive regions of maximum length q_i. Then, Γ is schedulable by EDF if $\forall i = 1, \ldots, n \quad q_i \leqslant Q(D_i)$ and

$$\forall t \in \mathcal{D} \quad Q(t) + \text{dbf}(t) \leqslant t, \tag{8}$$

where $\mathcal{D}$ denotes the set of deadlines defines in Theorem 3.

Under EDF, a target-sensitive task can be guaranteed to be executed as soon as possible if it is assigned the shortest relative deadline compatible with the shortest worst-case response time, that is, $D_1 = C_1 + B_1$, where $B_1 = \max\{q_2, \ldots, q_n\}$.

DOI: 10.1002/spe

3. REFERENCE SYSTEM

In this work, the platform illustrated in Figure 1 has been used as a reference example for a generic target-sensitive system. In this setup, a laser pulse of a given duration δ has to be generated at a proper time so that the light passes through the hole made on a rotating disk. The laser beam is positioned to point to the upper position of the disk (reference position), and it is correctly aligned to hit the target, consisting of a photoresistor (P) located behind the disk, as illustrated in the figure. In these conditions, the goal of the controller (PC) is to compute the precise instant of time at which the laser has to be switched on to pass through the hole, when it is in the upper disk position.

The radius of the hole, its position on the disk, and the radius of the laser spot are assumed to be known, whereas the angular velocity ω of the disk is not known and has to be estimated by a sensor, consisting of an optical encoder (E) mounted on the shaft of a DC motor (M), before the reduction. It is assumed that the value of ω changes slowly and can be considered to be constant within a revolution.

It is worth noting that the design issues discussed for the considered system are similar to those found in many other real world applications, from target tracking in defense systems to ignition scheduling in engine control systems.

The notation used throughout the paper to describe the system parameters and variables is reported in Table I.

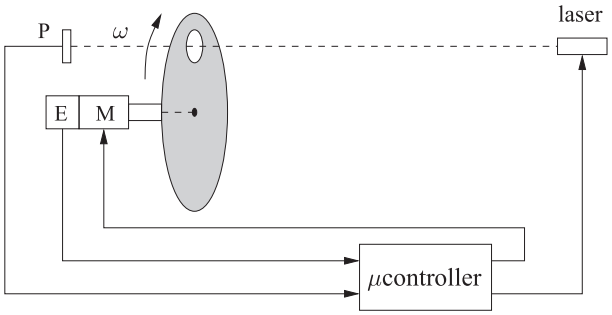

Figure 1. Platform used as a reference system.

Table I. Notation used for the parameters of the system.

R	Distance of the hole center from the disk center
R_B	Radius of the beam
R_H	Radius of the hole
D	Distance between the disk and the shooting device
θ	Angular position of the hole at the current reading
θ_{old}	Angular position of the hole at the previous reading
θ_0	Angular position of the hole in the uppermost position (reference position)
$\Delta\theta$	Angular difference between current and reference position
ω	Estimated angular velocity of the disk
t	Current time
t_0	Future time the hole will be in the uppermost disk position
Δt_a	Triggering delay of the laser device
δ	Laser pulse duration
t_s	Shooting time computed by the system
Δt	Time interval between current and firing instant
T_s	Time interval between two angular readings
T_{safe}	Sampling period to ensure that the target is within the safe region
ε_θ	Angular position error
ε_ω	Speed measurement error
ε_t	Time estimation error

DOI: 10.1002/spe

The next section derives the tolerances and the constraints among the variables necessary to define the application timing constraints.

4. TOLERANCES AND CONSTRAINT DERIVATION

The simplest method for estimating the angular speed ω is to sample the disk position at regular intervals T_s and divide the difference of two consecutive angular readings $(\theta - \theta_{old})$ by the period T_s between the readings, that is

$$\omega = \frac{\theta - \theta_{old}}{T_s}. \tag{9}$$

If ε_θ denotes the angular error of a measurement, the estimated speed will be affected by a maximum error equal to

$$\varepsilon_\omega = \frac{2\varepsilon_\theta}{T_s}. \tag{10}$$

If the direction of rotation is not known, to make a correct estimation, the angular difference of two consecutive samples cannot be higher than π, that is

$$|\omega| \leqslant \frac{\pi}{T_s}. \tag{11}$$

Also, to ensure that consecutive angular readings have different values (i.e., to have $\theta \neq \theta_{old}$), in the period T_s, the disk must cover an angle greater than $2\varepsilon_\theta$, which gives a lower bound on ω:

$$|\omega| \geqslant \frac{2\varepsilon_\theta}{T_s}. \tag{12}$$

For a given period T_s, the range of ω values in which the speed is correctly estimated is given by Equations (11) and (12). Vice versa, if the disk speed is known to be in a given range $[\omega_{min}, \omega_{max}]$, the two speed limits can be used to constrain the period T_s:

Constraint 1

$$\frac{2\varepsilon_\theta}{\omega_{min}} \leqslant T_s \leqslant \frac{\pi}{\omega_{max}}. \tag{13}$$

If θ_0 denotes the reference angular position, the angular difference $\Delta\theta$ with respect to θ_0 is given by

$$\Delta\theta = \theta_0 - \theta + \begin{cases} 2\pi & \text{if } \frac{\theta_0 - \theta}{\omega} < 0 \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

In particular, $\Delta\theta$ is a positive value and represents the angular displacement the disk has to cover from the current position to reach the reference angle θ_0 in the direction of rotation. The time required to cover this angular displacement (assuming a constant speed) is equal to

$$\Delta t = \frac{\Delta\theta}{|\omega|}. \tag{15}$$

As a consequence, the time t_0 at which the hole reaches the angle θ_0 is

$$t_0 = t + \Delta t = t + \frac{\Delta\theta}{|\omega|}. \tag{16}$$

Note that t_0 also represents the time at which the light must reach the photoresistor. For deriving a more general analysis, we assume that the laser device is affected by a triggering delay Δt_a, so that it must be triggered a time Δt_a in advance to hit the photoresistor at t_0.

DOI: 10.1002/spe

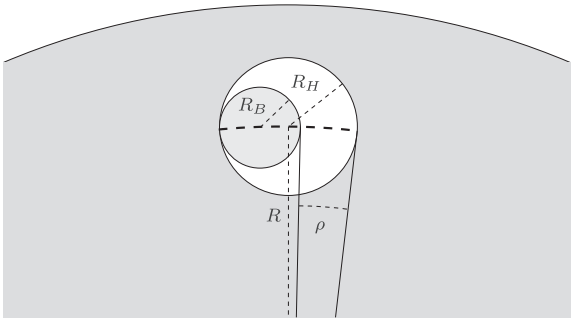

Figure 2. Tolerance caused by the difference between R_H and R_B.

In conclusion, the fire signal must be given at time $t_s = t_0 - \Delta t_a$; hence, another constraint is

Constraint 2

$$t_s = t + \frac{\Delta\theta}{|\omega|} - \Delta t_a. \tag{17}$$

Observe that the interval Δt computed by Equation (15) is affected by an error ε_t, which (in the worst case) is given by

$$\varepsilon_t = \frac{\Delta\theta + \varepsilon_\theta}{|\omega| - \varepsilon_\omega} - \frac{\Delta\theta}{|\omega|}. \tag{18}$$

If the hole center is positioned at distance R from the disk center, such a timing error, at speed ω, causes a maximum target displacement of $\varepsilon_t|\omega|R$, which is tolerated only if it is less than or equal to the difference between the hole radius R_H and the radius R_B of the laser spot,[‡] as illustrated in Figure 2; that is, if

$$\varepsilon_t|\omega|R \leqslant R_H - R_B. \tag{19}$$

The difference $R_H - R_B$ represents the chord approximation of the corresponding arc, which is reasonable for small angular values. Hence, its value is expressed in radians, and the angular tolerance ρ that ensures a successful shot is

$$\rho = \arcsin\left(\frac{2(R_H - R_B)}{R}\right). \tag{20}$$

The inequality given in Equation (19) can also be written as

$$\varepsilon_t \leqslant \frac{\rho}{|\omega|}. \tag{21}$$

A safety condition as a function of $\Delta\theta$ can be derived by substituting Equation (18) into Equation (21):

$$\Delta\theta \leqslant |\omega|T_s\frac{\rho - \varepsilon_\theta}{2\varepsilon_\theta} - \rho. \tag{22}$$

The right-hand side of Equation (22) represents the maximum angular difference $\Delta\theta_{max}$ at which the target is guaranteed to be caught at speed ω:

$$\Delta\theta_{max} = |\omega|T_s\frac{\rho - \varepsilon_\theta}{2\varepsilon_\theta} - \rho. \tag{23}$$

Observe that a value $\Delta\theta_{max} > 2\pi$ means that the target hit can be guaranteed even by planning the shoot more than one rotation ahead.

[‡]The derivation of the R_B value for laser spot is explained in Section 6.1.

DOI: 10.1002/spe

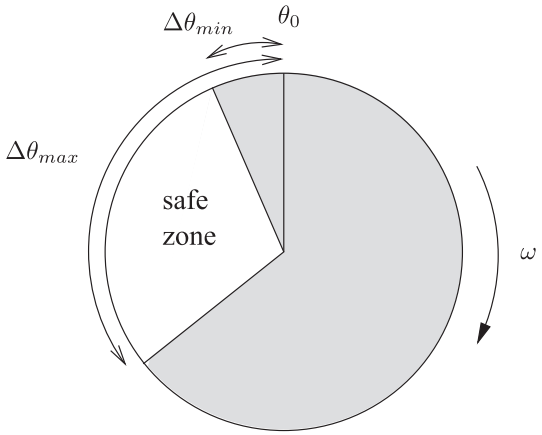

Figure 3. Safe region for computing the firing instant.

Note that $\Delta\theta_{max}$ reduces with ω, because small speed values increase the interval Δt in which the speed error ε_ω is integrated. Therefore, $\Delta\theta_{max}$ must be reduced to contain the angular drift within the tolerance ρ.

Likewise, a minimum angular interval $\Delta\theta_{min}$ is needed before firing:

$$\Delta\theta_{min} = |\omega|\Delta t_a. \tag{24}$$

Note that a value $\Delta\theta_{min} > 2\pi$ means that the firing time must be planned at least one rotation ahead.

In conclusion, a correct behavior can be guaranteed only if the firing time is set when the angular target distance $\Delta\theta$ from θ_0 is within the safe interval $[\Delta\theta_{min}, \Delta\theta_{max}]$. This means that to ensure that the firing time falls in the safe region, the encoder must be sampled with a period no larger than a maximum value T_{safe} given by

Constraint 3

$$T_{safe} = \frac{\Delta\theta_{max} - \Delta\theta_{min}}{|\omega|}. \tag{25}$$

An example of safe region for the case of a disk rotating clockwise is illustrated in Figure 3.

Another constraint results from the fact that, at the shooting instant, the laser is activated for a duration δ. Such an interval cannot not be too long, because an earlier laser activation would keep the laser on until the hole crosses the photoresistor, so accounting for a correct shot even on a wrong prediction. To prevent such a phenomenon, δ must be set equal to the minimum time T_{ph} just sufficient to activate the photoresistor, and this time cannot be larger than the photoresistor visibility time T_{vis}, which is given by

$$T_{vis} = \frac{2(R_H - R_B)}{\omega_{max} R}. \tag{26}$$

Hence, to avoid such a problem, it must be $T_{ph} \leqslant T_{vis}$, which means

Constraint 4

$$T_p \leqslant \frac{2(R_H - R_B)}{\omega_{max} R}. \tag{27}$$

5. TASK STRUCTURE AND FUNCTIONAL BEHAVIOR

The previous section derived three constraints, obtained from Equations (13), (17), and (25), respectively. They will be used to set the periods and deadlines of three corresponding tasks, which interact through a shared buffer, as depicted in Figure 4:

DOI: 10.1002/spe

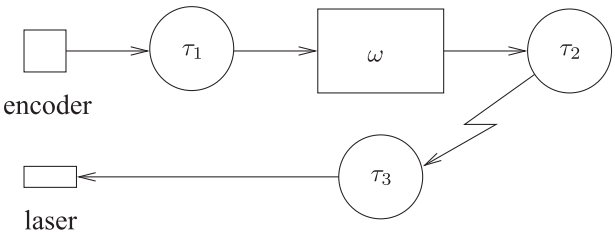

Figure 4. Tasks structure of the application.

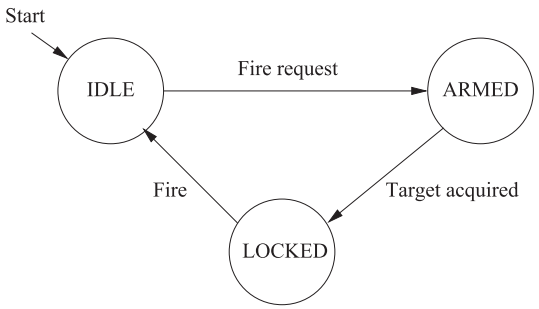

Figure 5. States diagram of the application.

- The first task, τ_1 (`estimate_speed`), is periodically activated to acquire the encoder with a period $T_1 = T_s$ (*constraint 1*). It derives the current position θ of the disk and, using the value (θ_{old}) detected in the previous run, estimates the disk angular velocity ω using Equation (9), and writes it into the shared buffer.
- The second task, τ_2 (`plan_shooting`) is also periodic. Every period T_2, it reads the current target position θ and the disk speed ω from the buffer, and, if the hole is found in the safe region, it calculates the shooting time t_s using *constraint 2*. This value is used to set an event at time t_s for activating the third task τ_3, which will actually trigger the laser (or the shooting device). To ensure that τ_2 is activated at least once when the current target position is within the safe region, the period T_2 is set according to *constraint 3*.
- Task τ_3 (`fire`) is a sporadic routine activated at time t_s to trigger the shooting device.

Figure 5 illustrates the state diagram describing the functional behavior of the application. At the beginning, the system is in the `IDLE` state waiting for a 'fire request'. When a 'fire request' is given by the user, the application switches to `ARMED` mode, in which the system is continuously sampled until the hole is found in the safe region. At this time, the firing instant t_s is computed by task τ_2 and the system switches to `LOCKED` mode, waiting for the shooting device to fire. Once the firing command is given, the system is switched back to the `IDLE` state by task τ_3.

5.1. Implementation notes

This section discusses some guidelines that should be followed in the implementation of the application tasks. For the sake of simplicity, the pseudo code of the tasks makes use of the following functions:

- `current_time()` returns the current time of the system;
- `set_state(value)` sets the system state at the value specified in the argument (`IDLE`, `ARMED`, or `LOCKED`;
- `read_sensor()` acquires the encoder and returns the current angular position of the hole;
- `write_buffer(value)` writes the value passed as argument in the shared buffer;
- `read_buffer()` reads the buffer and returns the current value of the shared variable;
- `post_task(`τ, t_s`)` sets the activation of task τ at time t_s;

DOI: 10.1002/spe

```
task   estimate_speed_A {
float  th, th_old;     // current and previous position
float  w;              // estimated disc speed

       th_old = read_sensor();

       while (1) {
            wait_for_period();
            th = read_sensor();
            w = (th − th_old) / task_period;
            write_buffer(w);
            th_old = th;
       }
}
```

Figure 6. Implementation A of task τ_1 for the estimation of the disk speed.

- `wait_for_period()` suspends the execution of the calling task until the next periodic activation.

At initialization, the system state is set to `IDLE` and is brought in the `ARMED` state by a fire request issued by the user (e.g., by pressing a button).

5.2. Estimation of the disk velocity

If task τ_1 (`estimate_speed`) is assigned the highest priority, then consecutive jobs are exactly separated by a period T_s, thus the angular speed of the disk can be computed by Equation (9). A first implementation of such a task (denoted as implementation A) is reported in Figure 6. However, note that if τ_1 is assigned a medium priority, consecutive samples will be separated by a varying interval because of the interference of higher-priority tasks. In such a general case, the disk angular speed should be estimated taking into account the actual time difference between consecutive samples:

$$\omega = \frac{\theta - \theta_{old}}{t - t_{old}}. \tag{28}$$

The corresponding task implementation (denoted as implementation B) is shown in Figure 7.

It is worth observing that the speed estimation computed by implementation B is correct only if the current time is always acquired just after reading the sensor, as written in the code. A preemption between the two instructions would introduce a delay that would affect the speed computation. In order to avoid such a situation, the instructions for reading the sensor and the current time should be executed atomically, encapsulating them in a non-preemptive region. The task implementing such a solution (denoted as implementation C) is reported in Figure 8.

The non-preemptive regions, however, may introduce a potential blocking time (Q) to the higher-priority task that has to be accounted in the schedulability analysis, as reported in Section 2.

Note that the blocking time Q also affects the bounds of the safety region, because it has to be added to ε_t in Equation (19). Thus, $\Delta\theta_{max}$ becomes:

$$\Delta\theta_{max} = |\omega| T_s \frac{\rho - \varepsilon_\theta}{2\varepsilon_\theta} - \rho - |\omega| Q \frac{|\omega| T_s - 2\epsilon_\theta}{2\epsilon_\theta}. \tag{29}$$

5.3. Shooting the target

Based on the speed estimated by τ_1, task τ_2 (`plan_shooting`) computes the shooting time t_s to hit the target according to Equation (17). Note that, if the hole is not inside the safe region shown in Figure 3, the shooting time cannot be correctly estimated, hence no action is executed by τ_2,

DOI: 10.1002/spe

```
task   estimate_speed_B {
float  th, th_old;     // current and previous position
float  t, t_old;       // current and previous time
float  w;              // estimated speed

       th_old = read_sensor();
       t_old = read_current_time();

       while (1) {
            wait_for_period();
            th = read_sensor();
            t = current_time();
            w = (th − th_old) / (t − t_old);
            write_buffer(w);
            th_old = th;
            t_old = t;
       }
}
```

Figure 7. Implementation B of task τ_1 for the estimation of the disk speed.

```
task   estimate_speed_C {
float  t, t_old;       // current and previous time
float  th, th_old;     // current and previous position
float  w;              // estimated speed

       disable_preemption();
       th_old = read_sensor();
       t_old = current_time();
       enable_preemption();

       while (1) {
            wait_for_period();
            disable_preemption();
            th = read_sensor();
            t = current_time();
            enable_preemption();

            w = (th − th_old) / (t − t_old);
            write_buffer(w);
            th_old = th;
            t_old = t;
       }
}
```

Figure 8. Implementation C of task τ_1 for the estimation of the disk speed.

DOI: 10.1002/spe

which is then suspended until the next period. When the hole is detected within the safe region, then t_s is estimated, and the activation of τ_3 is set at time t_s. When executed, task τ_3 (`fire`) triggers the shooting device and switches the system state to IDLE. The pseudo code of tasks τ_2 and τ_3 is reported in Figures 9 and 10, respectively. Observe that the interval between two consecutive executions of task τ_2 can be larger than period T_2, because of the interference (I_2) caused by the higher-priority tasks (τ_1 and τ_3). It follows that, to ensure that the shooting time falls in the safe region, we have to guarantee that $T_2 + I_2 \leqslant T_{safe}$, that is (by Equation (25))

$$T_2 \leqslant \frac{\Delta\theta_{max} - \Delta\theta_{min}}{|\omega|} - I_2. \tag{30}$$

By exploiting Equations (23) and (24), the constraint on T_2 can be expressed as follows:

$$T_2 \leqslant T_s \frac{\rho - \varepsilon_\theta}{2\varepsilon_\theta} - \frac{\rho}{|\omega|} - \Delta t_a - I_2. \tag{31}$$

```
task  plan_shooting {
float t, t_s;          // current and shooting time
float th;              // current hole position
float w;               // estimated disc speed

      while (1) {
          wait_for_period();
          if (state == ARMED) then
              w = read_buffer();
              <compute Δθ_max by Eq. (23)>;
              <compute Δθ_min by Eq. (24)>;

              disable_preemption();
              t = current_time();
              th = read_sensor();
              <compute Δθ by Eq. (14)>;

              if (Δθ ∈ [Δθ_min, Δθ_max] ) then
                  <compute t_s by Eq. (17)>;
                  post_task(fire, t_s);
                  set_state(LOCKED);
              }
              enable_preemption();

          }
      }
}
```

Figure 9. Pseudo code of task τ_2 for computing the shooting time.

```
task  fire {

      trigger_laser();
      set_state(IDLE);
}
```

Figure 10. Pseudo code of task τ_3 for firing.

DOI: 10.1002/spe

Observe that, for achieving a correct execution, task τ_3 must be activated exactly at time t_s and immediately scheduled by the kernel. Therefore, τ_3 must be assigned the highest priority, so that it can preempt all the other application tasks. However, being τ_3 a sporadic task, its interference has to be properly accounted in the analysis to guarantee the application feasibility, as shown in Section 2.

To characterize the worst-case interference of the sporadic task, its minimum interarrival time has to be determined based on the system features. From the behavior described in the previous text, task τ_3 can be activated no more than once for every disk rotation, hence $T_3 = 2\pi/\omega_{max}$.

Moreover, being τ_3 the task with the highest priority, it cannot be preempted by other tasks, but it can be blocked by the longest non-preemptive region Q; thus, its relative deadline can be set as $D_3 = C_3 + Q$. The relative deadlines of the other two tasks, τ_1 and τ_2, can be set equal to their periods ($D_1 = T_1$ and $D_2 = T_2$), because they are implemented to be tolerant to the interference.

6. EXPERIMENTAL VALIDATION

A set of experiments have been carried out on a real platform, which has been developed with the purpose of validating the proposed approach and comparing the behavior of the various implementations presented in Section 5.

6.1. Hardware platform

The test system consists of a compact disk mounted on a brushed DC servomotor having a gear reduction ratio of 9.68:1. The hole position angle has been measured by an optical encoder (mounted on the motor shaft, before the reduction) having 48 pulses per rotation, leading to an angular error on the disk equal to

$$\varepsilon_\theta = \frac{360}{48 \cdot 9.68} = 0.775 \text{ deg } \left(13.52 \cdot 10^{-3} \text{ rad}\right). \tag{32}$$

The plant controller was implemented on a Flex[§] board based on a Microchip 16-bit dsPIC microcontroller running the application on top of a real-time kernel.

The shooting device consists of a laser pointer, pointing toward a photoresistor positioned behind the disk and acquired by a 12-bit analog-to-digital converter. The firing signal is generated by a digital output line connected with the laser. During firing, the laser is activated for an interval $\delta =$ 400 μs, which is equal to the time T_{ph} just sufficient to switch the photoresistor on. The hole has a radius $R_H = 3$ mm, and its center is positioned at distance $R = 53$ mm from the disk center. In this setup, the shooting time Δt_a can clearly be neglected.

To determine the value of R_B in our system, a specific experiment has been carried out to characterize the effect of the laser spot on the used photoresistor. Keeping the laser always on, the photoresistor output was acquired for different angular values of the hole around the reference position. Figure 11 shows the photoresistor response (in volts) to the laser beam as a function of the hole angular position θ. Note that the value of θ_H is the maximum angular position for which the photoresistor is sensible to the laser beam and it is related with the hole radius (R_H) by the relation $R_H = R\tan(\theta_H)$. The interval $[-\theta_T, \theta_T]$ represents the angular range in which the photoresistor output is above the interrupt activation threshold (V_{th}). Hence, as clearly depicted in the figure, the laser spot radius can be estimated as

$$R_B = R\tan(\theta_H - \theta_T). \tag{33}$$

The measured values of θ_T and θ_H resulted to be 2 deg ($3.49 \cdot 10^{-2}$ rad) and 3.24 deg ($5.65 \cdot 10^{-2}$ rad), respectively, so the equivalent laser spot radius resulted to be $R_B = 1.15$ mm.

The maximum motor speed is 4000 deg/s (69.81 rad/s), but to observe the system behavior when the speed exceeds the bound given by Equation (11), we considered ω_{max} = 3000 deg/s (52.36 rad/s). A minimum rotation speed ω_{min} = 500 deg/s (8.72 rad/s) was also imposed to derive a safe value of

[§]Flex board web site: http://www.evidence.eu.com/products/flex.html.

DOI: 10.1002/spe

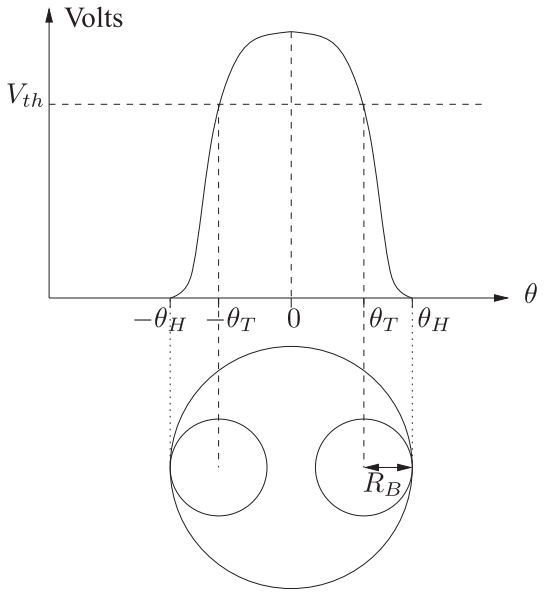

Figure 11. Photoresistor output for different hole angular values around the reference position.

T_2, based on Equation (31). In this setting, the bounds of the safe region computed by Equations (24) and (23) resulted to be

$$\begin{cases} \Delta\theta_{min} = 0 \\ \Delta\theta_{max} = 1.035|\omega|T_s - \rho \end{cases}$$

where, as in Equation (20), $\rho = 2.38$ deg ($41.52 \cdot 10^{-3}$ rad).

Also note that, with the current settings, constraint 4 is satisfied, being $T_{ph} = 400$ ms, which is less than the photoresistor visibility time

$$T_{vis} = \frac{2(R_H - R_B)}{\omega_{max} R} = 1 \text{ ms}.$$

6.2. *Software implementation*

The application has been developed on top of the ERIKA Enterprise kernel [22], which is an open-source real-time kernel for embedded platforms compliant with the OSEK/VDX standard [23]. In ERIKA, a task is a sequence of jobs, each resulting from the execution of a function written in C code. For each periodic task, an alarm is used to activate its jobs with the specified period. A kernel functionality is in charge of managing the alarms through a hardware timer configured with a granularity set by the application. Tasks' initializations are performed inside the main function, together with the alarms activations. According to the OSEK/VDX standard, all the tasks and the corresponding scheduling policy are set statically at compile time. The developer can decide among different scheduling algorithms, as fixed priorities (FPs) and earliest deadline first (EDF) [17].

To test the system near the limit conditions, the period of task τ_1 has been set to the maximum value allowed by Equation (13), that is

$$T_1 = T_s = \frac{\pi}{\omega_{max}} = 60 \text{ ms},$$

while period T_2 has been set within the bound expressed in Equation (31), considering $I_2 = 0$ and $\omega = \omega_{min} = 500$ deg/s (8.72 rad/s) (which gives $T_2 \leqslant 61.84$ ms). In all the experiments, $T_2 = 55$ ms and $T_3 = 2\pi/\omega_{max} = 120$ ms (equal to the minimum interarrival time of task τ_3).

A data logging task, τ_{log}, is also included in the application in order to monitor the main variables and send them to a PC via a serial line. To avoid extra blocking interference, all logged variables are shared by means of double buffering.

DOI: 10.1002/spe

Table II. Task set parameters used in the experiments.

Task	C_i (ms)	T_i (ms)	U_i	Priority under FP	Relative deadline under EDF (ms)
τ_1	0.1	60	$1.66 \cdot 10^{-3}$	2	60
τ_2	0.07	55	$1.27 \cdot 10^{-3}$	3	55
τ_3	0.6	120	$5.00 \cdot 10^{-3}$	5	0.6
τ_d	3.5	35	0.1	4	35
τ_{log}	5.1	1000	$5.10 \cdot 10^{-3}$	1	1000

FP, fixed priority; EDF, earliest deadline first.

Table III. Set of experiments carried out on the reference system.

Experiment	Output	Free variable	Scheduler
Exp. 1	Miss ratio	ω	FP
Exp. 2	Miss ratio	U_d	FP
Exp. 3	Miss ratio	ω	FP
Exp. 4	Miss ratio	ω	EDF

FP, fixed priority; EDF, earliest deadline first.

Finally, in order to test the different implementations under variable workload conditions, a higher-priority disturbing task τ_d has been introduced, with a period $T_d = 35$ ms and a computation time C_d, which can be increased by a factor k, so that $C_d = 3.5 \cdot k$ ms. As a result, the utilization of the disturbing task is given by

$$U_d(k) = C_d \cdot k / T_d = k/10$$

and defined in such a way that the system reaches a total utilization of 1.0 for $k = 9$. Table II summarizes the timing parameters of the task set (higher values of priority denote higher priority). It is worth noting that, without the disturbing task, the application is schedulable under both FP and EDF. With the disturbing task, the analysis reported in Section 2 shows that the application is schedulable under both FP and EDF for $k \leqslant 9$.

6.3. Experiments

This section presents a set of experiments on the reference system considered in the paper aimed at validating the proposed approach. The metrics for evaluating the system performance is the *miss ratio*, that is the ratio of the cumulative number of missed and skipped shots and the number of full disk rotations performed in the experiment (where a shot is triggered at each rotation). The first three experiments are executed under fixed priority scheduling, whereas the fourth experiment has been carried out under EDF scheduling, using the parameters reported in Table II. Table III briefly summarizes the set of experiments.

6.3.1. Experiment 1. The first experiment is aimed at comparing the three implementations of task τ_1 presented in Section 5 (referred to A, B, and C), with the purpose of evaluating their impact on the system performance as a function of the disk speed ω. In this test, the speed is varied from 2760 deg/s (48.17 rad/s) to 3400 deg/s (59.34 rad/s), and the disturbing task is not present (i.e., $U_d = 0$). For each speed value, 800 full disk rotations are performed. The output of this test is reported in Figure 12.

Note that, being $\omega_{max} = 3000$ deg/s (52.36 rad/s), all the three implementations quickly reach a miss ratio equal to 1.0 as soon as the rotation speed crosses the limit value of 3000 deg/s (52.36 rad/s). For $\omega \leqslant 2900$ deg/s (50.61 rad/s), all the three implementations exhibit a good performance (miss ratio = 0.0) because, in the absence of the disturbing task, they are able to correctly calculate ω with enough precision.

DOI: 10.1002/spe

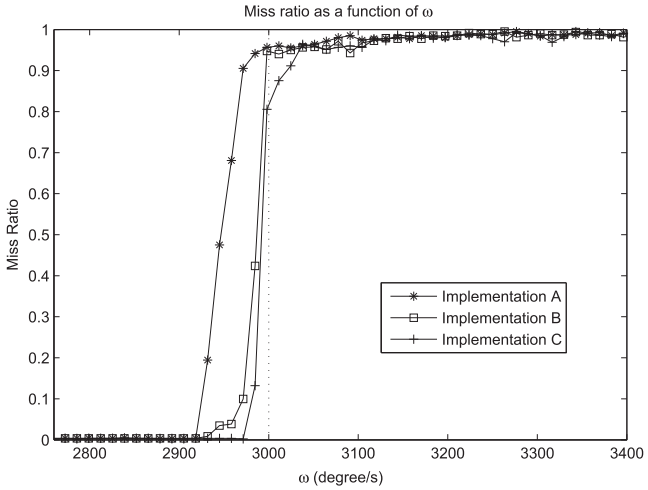

Figure 12. Miss ratio as a function of ω for the three implementations (A, B, and C) of task τ_1.

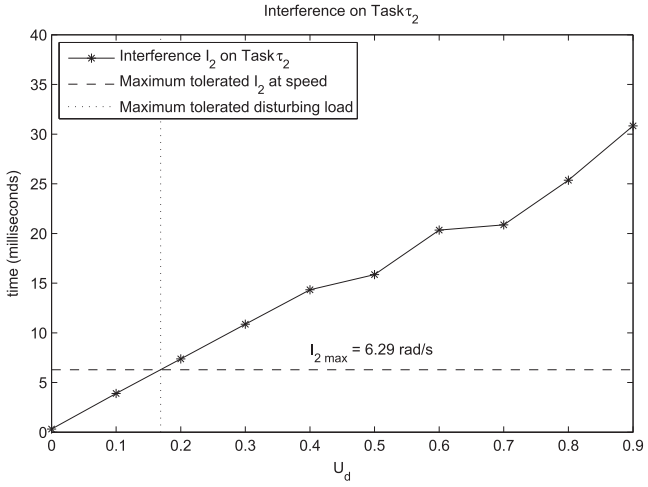

Figure 13. Actual interference measured on τ_2 as a function of U_d. The dashed line denotes the value of I_{2max}.

The different behavior of the three versions mainly appears for rotation speeds in the range [2900 deg/s, 3000 deg/s] (50.61–52.36 rad/s). As expected, implementation A provides the worst performance, because ω is estimated by Equation (9), without taking into account the execution jitter. However, because there is no disturbing task, the jitter is minimum and allows a fairly accurate estimate up to a speed $\omega = 2900$ deg/s (50.61 rad/s). Implementation B takes jitter into account, and as there is no disturbing task, its behavior is roughly equal to implementation C. In fact, both implementations B and C present a negligible number of missed targets even for ω values quite near to ω_{max}.

6.3.2. Experiment 2. The second experiment has been carried out with the objective of evaluating the robustness of the three implementations of τ_1 in the presence of an increasing interference. This was performed by setting a fixed disk speed and measuring the actual interference I_2 on task τ_2 for increasing values of the disturbing load U_d, obtained by increasing the factor k from 0 to 9. The value of ω has been set to $\omega = 2818$ deg/s (49.19 rad/s), which is a high value that still guarantees (as shown in Figure 12) a negligible miss ratio for all the three implementations without disturbing load ($U_d = 0$). Figure 13 shows the interference measured on τ_2 as a function of the utilization of the disturbing task U_d.

DOI: 10.1002/spe

The dashed line represents the maximum interference ($I_{2max} = 6.28$ ms) that can be tolerated by the system to catch the target, computed by Equation (31) for $\omega = 2818$ deg/s (49.19 rad/s). Note that the value at which the two lines intersect represents the maximum disturbing load ($U_d^* \simeq 0.18$) that guarantees the correct performance.

In order to verify the adherence of the theoretical bound I_{2max} provided by Equation (31) with the actual performance of the different task implementations, the miss ratio has been monitored as a function of the disturbing load U_d, for the same angular velocity $\omega = 2818$ deg/s (49.19 rad/s) used in the experiment. The results of this test are reported in Figure 14, where each value in the plot represents the miss ratio over 800 full disk rotations.

As it is clear from the plots, implementation A (which does not take the interference into account) starts missing the target for low values of U_d. Implementation B (which takes interference into account but does not guarantee an atomic computation of ω) exhibits a better performance than A but still produces some misses at values of $U_d \leqslant U_d^*$. Implementation C not only guarantees a zero-miss ratio for $U_d \leqslant U_d^*$, but it is also able to keep a good performance for slightly higher U_d values, because the actual interference may be less than the maximum one considered in the analysis.

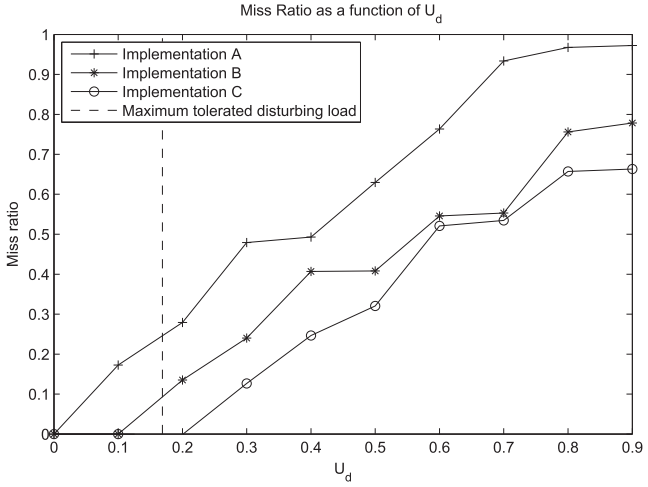

Figure 14. Miss ratio as a function of U_d for the three implementations (A, B, and C) of task τ_1.

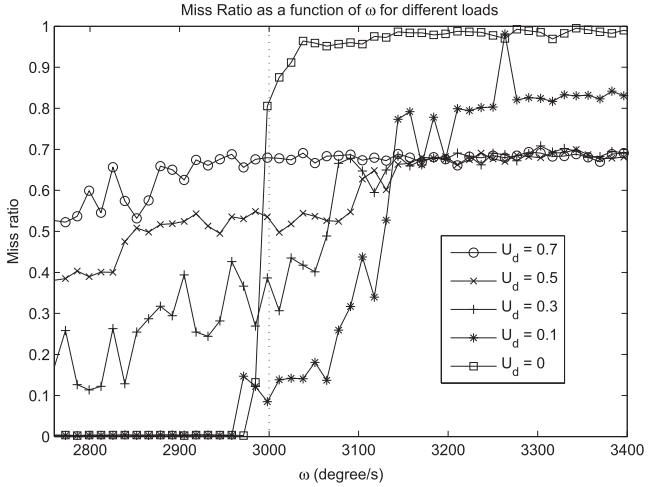

Figure 15. Miss ratio as a function of ω for different disturbing loads.

DOI: 10.1002/spe

6.3.3. Experiment 3. A third experiment has been performed to evaluate how the miss ratio of implementation C is affected by the rotation speed for different disturbing loads. As in the first experiment, ω was varied from 2760 deg/s (48.17 rad/s) to 3400 deg/s (59.34 rad/s), performing 800 full disk rotations for each point of the plot and for five different disturbing loads ($U_d = 0, 0.1, 0.3, 0.5$, and 0.7). The results of this experiment are shown in Figure 15.

Note that when $\omega < \omega_{max}$, the system never misses the target if the utilization of the disturbing load is below the critical value ($U_d \leqslant U_d \simeq 0.18$). For higher disturbing loads, the number of misses increases proportionally to the interference. For $\omega > \omega_{max}$, it is interesting to observe that an increasing disturbing load produces a side effect that allows obtaining a correct speed estimate once in a while. This can be explained by noting that, in the presence of high disturbing load, the jitter on τ_1 increases, causing the execution of consecutive jobs to vary a lot within its period. As a consequence, it may often happen that two consecutive jobs execute close to each other, so leading to a correct speed estimate, even for $\omega > \omega_{max}$. Figure 16 illustrates the phenomenon described in the previous text, where an increased interference C_d may reduce the interval $\tilde{T}_s = t_2 - t_1$ between successive readings.

6.3.4. Experiment 4. A final experiment has been carried out to test the performance of the system under EDF scheduling. Task-relative deadlines were assigned to be equal to task periods, except for the sporadic task τ_3, whose relative deadline was set equal to its worst-case computation time. In particular, because the difference between the two scheduling algorithms can only be appreciated under heavy workload conditions, this test was planned to verify how the miss ratio of implementation C is affected by the rotation speed for increasing disturbing loads. As in experiment 3, ω was varied from 2760 deg/s (48.17 rad/s) to 3400 deg/s (59.34 rad/s), performing 800 full disk rotations

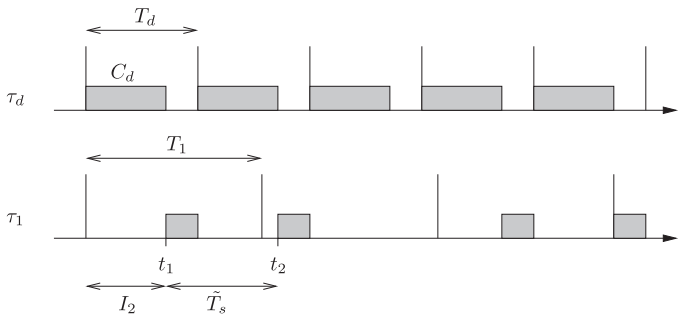

Figure 16. Side effect of the interference that may reduce the interval between successive readings.

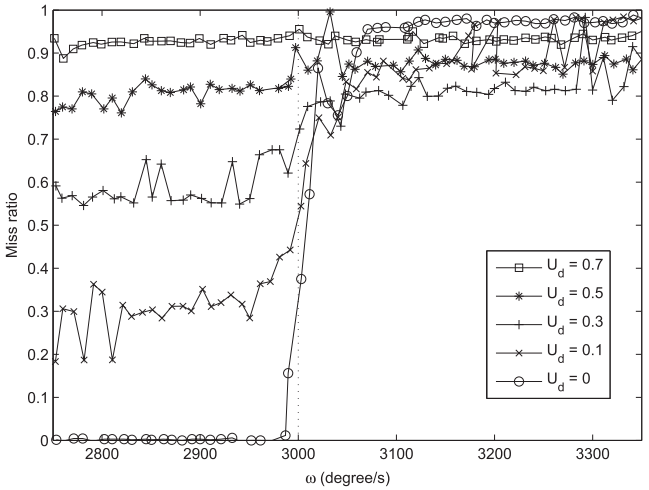

Figure 17. Miss ratio as a function of ω for different disturbing loads under earliest deadline first scheduling.

DOI: 10.1002/spe

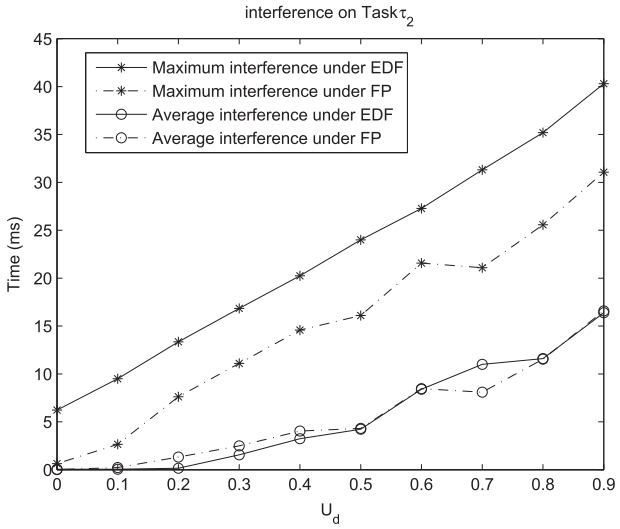

Figure 18. Average and maximum interference measured on τ_2 as a function of U_d under fixed priority (FP) and earliest deadline first (EDF) scheduling.

for each point of the plot and for five different disturbing loads ($U_d = 0, 0.1, 0.3, 0.5$, and 0.7). The results of this experiment are shown in Figure 17.

By comparing Figures 15 and 17, it can easily be seen that the general performance trends under FP and EDF are quite similar. However, note that for $\omega < \omega_{max}$, for the same values of U_d, the miss ratio achieved under EDF is slightly higher than under FP, because EDF distributes the jitter more evenly among the tasks [24]. To better explain this phenomenon, we measured the interference suffered by tasks τ_1 and τ_2 under both FP and EDF. As expected, the interference suffered by τ_2 (whose priority is higher than τ_1) is higher under EDF, as illustrated in Figure 18, hence the higher jitter. The interference on task τ_1 resulted to be comparable for both schedulers; hence, it is not reported here.

Also note that the compensation phenomenon observed under FP for $\omega > \omega_{max}$ (illustrated in Figure 16) occurs less frequently under EDF due to the more uniform jitter distribution among the tasks.

7. CONCLUSIONS

This paper presented a general methodology for the design and analysis of target-sensitive real-time systems, where the output of one or more tasks must be produced at precise time instants, rather than within a deadline. The proposed approach has been instantiated to a reference platform to practically show how to model the physical system, how to derive timing constraints, how to map specific functionalities into periodic and sporadic tasks, and finally how to verify the system schedulability and performance.

Three alternative implementations have been proposed and compared to evaluate the effect of the inter-task interference on the system performance. Also, the application has been executed on two different scheduling algorithms, available in the ERIKA real-time kernel: a fixed priority scheduler and earliest deadline first. The experimental results reported in the paper, besides verifying the effectiveness of the proposed procedure, confirmed that, if following the proper design guidelines, it is possible to implement a system that never fails its goal. Even in the presence of a high-priority disturbing load, it has been shown how the interference can be characterized and taken into account to meet the application requirements.

REFERENCES

1. Lunney HMW. Time as heard in speech and music. *Nature* 1974; **249**(5457):592.
2. Iyer V, Bilmes J, Wright M, Wessel D. A novel representation for rhythmic structure. *Proceedings of the International Computer Music Conference*, San Francisco, California, USA, 1997; 97–100.

DOI: 10.1002/spe

3. Dannenberg RB, Jameson DH. Real-time issues in computer music. *Proceedings of the 14th Real-Time Systems Symposium (RTSS '93)*, Raleigh-Durham, North Carolina, USA, December 1–3, 1993; 258–261.
4. Brandt E, Dannenberg R. Low-latency music software using off-the-shelf operating systems. *Proceedings of the Internationl Computer Music Conference*, San Francisco, California, USA, 1998; 137–141.
5. Houshangi N. Control of a robotic manipulator to grasp a moving target using vision. *Proceedings of the IEEE International Conference on Robotics and Automation*, Cincinnati, Ohio, USA, May 13–18, 1990; 604–609.
6. Allen PK, Timcenko A, Yoshimi B, Michelman P. Trajectory filtering and prediction for automated tracking and grasping of a moving object. *Proceedings of the IEEE International Conference on Robotics and Automation, (ICRA '92)*, Nice, France, May 12–14, 1992; 1850–1856.
7. Buttazzo GC, Allotta B, Fanizza F. Mousebuster: a robot for catching fast objects. *IEEE Control Systems Magazine* 1994; **14**(1):49–56.
8. Facchinetti T, Buttazzo G. A real-time system for tracking and catching moving targets. *Proceedings of the 5th IFAC International Symposium on Intelligent Components and Instruments for Control Applications (SICICA 2003)*, Aveiro, Portugal, July 9–11, 2003; 251–256.
9. Linderoth M, Robertsson A, Åström K, Johansson R. Vision based tracker for dart-catching robot. *Proceedings of the 9th IFAC International Symposium on Robot Control (SYROCO'09)*, Gifu, Japan, September 9–12, 2009; 883–888.
10. Jensen DE, Locke DC, Tokuda H. A time-driven scheduling model for real-time operating systems. *Proceedings of the 6th IEEE Real-Time Systems Symposium*, San Diego, California, USA, December 3–6, 1985; 112–122.
11. Chen K, Muhlethaler P. A scheduling algorithm for tasks described by time value function. *Real-Time Systems* 1996; **10**(3):293–312.
12. Farzinvash L, Kargahi M. A scheduling algorithm for execution-instant sensitive real-time systems. *Proceedings of the 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA '09)*, Beijing, China, August 24–26, 2009; 511–518.
13. Guerra R, Fohler G. A gravitational task model for target sensitive real-time applications. *Proceedings of the 20th Euromicro Conference on Real-Time Systems*, Prague, Czech Republic, July 2–4, 2008; 309–317.
14. Guerra R, Fohler G. On-line scheduling algorithm for the gravitational task model. *Proceedings of the 21st Euromicro Conference on Real-Time Systems (ECRTS 09)*, Dublin, Ireland, July 1–3, 2009; 97–106.
15. Tidwell T, Glaubius R, Gill CD, Smart WD. Optimizing expected time utility in cyber-physical systems schedulers. *Proceedings of the 31st IEEE Real-Time Systems Symposium (RTSS 2010)*, San Diego, California, USA, November 30 – December 3, 2010; 193–201.
16. Buttazzo GC, Franco CD, Marinoni M. Target-sensitive systems: analysis and implementation issues. *Proceedings of the 17th IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2012)*, Cracow, Poland, September 17–21, 2012; 1–8.
17. Liu CL, Layland JW. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the Association for Computing Machinery* 1973; **20**(1):46–61.
18. Lehoczky J, Sha L, Ding Y. The rate monotonic scheduling algorithm: exact characterization and average case behavior. *Proceedings of the 10th IEEE Real-Time Systems Symposium (RTSS'89)*, Santa Monica, California, USA, December 5–7, 1989; 166–171.
19. Yao G, Buttazzo G, Bertogna M. Feasibility analysis under fixed priority scheduling with limited preemptions. *Real-Time Systems* 2011; **47**(3):198–223.
20. Baruah SK, Rosier LE, Howell RR. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Journal of Real-Time Systems* 1990; **2**:301–324.
21. Bertogna M, Baruah S. Limited preemption EDF scheduling of sporadic task systems. *IEEE Transactions on Industrial Informatics* 2010; **6**(4):579–591.
22. Gai P, Lipari G, Abeni L, di Natale M, Bini E. Architecture for a portable open source real-time kernel environment. *Proceedings of the 2nd Real-Time Linux Workshop*, Orlando, FL, USA, November 2000; 1–9.
23. OSEK. *OSEK/VDX operating system specification 2.2.1.* OSEK Group: (Available from: http://www.osek-vdx.org), 2003.
24. Buttazzo GC. Rate monotonic vs. EDF: judgment day. *Real-Time Systems* 2005; **29**(1):5–26.

DOI: 10.1002/spe