FP-H: A Real-Time Energy Aware Scheduler with Fixed Priority Assignment for Sustainable Wireless Devices

Maryline Chetto

FP-H: A Real-Time Energy Aware Scheduler with Fixed Priority Assignment for Sustainable Wireless Devices

Maryline Chetto
Nantes Université, École Centrale Nantes
CNRS, LS2N, UMR 6004, F-44000 Nantes, France
maryline.chetto@univ-nantes.fr

***Abstract*—This paper addresses the scheduling issue in a real-time computing system such as a wireless sensor node which is supplied with regenerative energy present in the environment. We consider preemptive task scheduling with fixed priority assignment. We propose a novel energy harvesting aware scheduling approach, namely FP-H. We show how processing time and energy should be assigned to the deadline constrained tasks in a short-term perspective so as to guarantee energy neutrality whenever possible.**

***Index Terms*—Autonomous sensor, real-time computing, energy harvesting, energy neutrality, Fixed Priority Scheduling.**

I. Introduction

Advancements in energy, computing and wireless communication technology allow to offer new services for people and make smart environments and better quality of life. For example, traffic monitoring, car parking monitoring, space detection and public transportation monitoring are use cases for smart cities. Batteries are the obvious way of powering wireless devices. However, regular battery replacement or recharging is vital to ensure longtime operation for them. Such a requirement generally implies high maintenance costs, especially when the devices are accessible with difficulty. Note in addition that millions of sensors may be deployed around the city. The sensors have to work autonomously for very long periods of time without any supervision or human intervention. Smart city applications will expand only with a reduced operational cost and increased sustainability model for these low powered and low maintenance sensors. As battery powered sensors stop working after several years, replacing millions of primary batteries leads to environmental pollution. Energy Harvesting (EH) then appears as a potential alternative to address this autonomy issue (see Figure 1). Ambient light as mechanical energy can be drawn from the environment so as to supply small electronic devices including wireless sensors quasi perpetually [1].

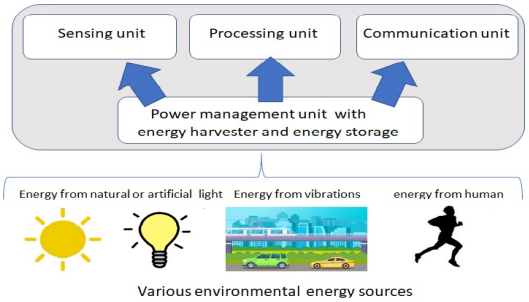

Fig. 1. Energy harvesting to supply small devices

Nonetheless, relying on variable environmental energy in a sensor node makes it challenging to guarantee reliable operation over the lifetime horizon. Thus, new methods and techniques are necessary to assess the longtime behavior of any EH-powered device which, in addition may have real-time constraints. The common characteristic of the so-called RTEH (Real-Time Energy Harvesting) systems is periodicity of activities that generally involve sampling, processing the sensed value, transmitting data, etc. [2], [3] as depicted in Figure 2. Energy neutrality is the property of an RTEH system that first should consume no more energy than the harvested energy and second should respect its timing requirements in every circumstance. To make a system energy neutral will require to identify the available power output of the harvester such as solar panel, the capacity of the energy storage unit and the energy which is consumed by the different tasks in operation.
Clearly, the challenging questions to be solved in such a system are real-time scheduling [4] [5], power management and dimensioning. Firstly, how to assign a priority to each task in accordance to its importance and/or urgency? Secondly, how to dynamically adapt the activity of the processing unit so as to subsist perpetually, given the profile of the energy source and the timeliness requirements of the tasks? Thirdly, how to define the size of both the energy storage unit and the harvester in order to guarantee an acceptable quality of service?

The rest of the sections in the paper is organized as follows. In Section 2, a qualitative review on scheduling techniques for RTEH systems is carried out. In Section 3, the details of the system model under consideration, for next-generation

autonomous real-time systems, are presented. Section 4 describes a new energy harvesting aware scheduler, named FP-H. A discussion on applicability of the scheduler is presented in section 5 followed by the conclusion in Section 6.

II. Related works

In this section, we survey the previous studies which are directly related to the proposed approach. The research on the design of self-sustainable devices started at the beginning of the 2000's. The classical real-time schedulers including RM (Rate Monotonic) and EDF (Earliest Deadline First) [6] fail in energy harvesting systems where the supply energy is intermittent. In the latters, energy must be treated as an equally important resource as time. In particular, we have to characterize the tasks by both processing time and energy consumption. And we have to achieve the online monitoring of available energy in the storage unit as well as the online prediction of environmental energy produced in near future.
The first work in the literature that explored task scheduling in monoprocessor RTEH systems is reported in [7]. It addressed frame-based tasks with voltage and frequency scaling capabilities. The Lazy Scheduling Algorithm (LSA) proposed by Moser et al. [8] is an optimal EDF-based algorithm based on as late as possible policy which applies to any set of deadline constrained tasks. Liu et al. extended LSA with EA-DVFS and HA-DVFS which are DVFS-based algorithms [9], [10]. They slow down the processor so as to save energy and speed up task execution in case of overflowing harvested energy. In [11], an optimal preemptive fixed-priority scheduling algorithm called PFP_{ASAP} was proposed, assuming a constant power source. In [12], EDF was proved to be the best non idling scheduler. In [13], Chetto presents a strongly optimal scheduler, namely ED-H, for the general RTEH model with no restriction on task arrival profile and energy source profile. ED-H is an idling variant of EDF where two key values are computed on-line: slack time and slack energy. The dynamic power management joined to the EDF priority assignment rule guarantees optimality in terms of scheduling and energy neutrality whenever possible. An exact schedulability test is given for a generic set of deadline constrained jobs, assuming accurate prediction of the incoming energy budget. In summary, most research works addressed the EDF dynamic priority assignment rule [14]. Even if EDF allows higher processor utilization than fixed-priority schedulers, this scheduler is not commonly integrated in the commercial RTOS.

A. System Model and Assumptions

Hereafter, we address an RTEH system that consists of three major parts: energy harvester, computing module supporting the real-time software and rechargeable energy storage with limited capacity.

B. System Model

We consider a real-time embedded system which is supplied from an energy source through an energy harvester such as solar panel. The energy harvested from time t_1 to t_2 is calculated with following formula $E_s(t_1, t_2) = \int_{t_1}^{t_2} P_p(t)dt$ where $P_p(t)$ is the worst case charging rate (WCCR).

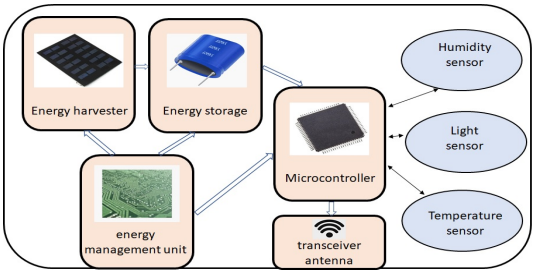

Fig. 2. Framework of an autonomous sensor node

The energy storage such as a rechargeable battery allows to continue operation even when no energy is harvested from the environment. It has nominal capacity C. It stores the extra amount of energy harvested for immediate or future use. We assume as negligible energy wasted in charging and discharging the battery.
At a given time t we have $C_{min} \leq C(t) \leq C_{max}$ where $C(t)$ is the amount of energy available in the storage at time t.

The real-time software that we are interested has independent jobs which may be the invocation requests of N periodic tasks. $\tau = \{\tau_1, \tau_2, ..., \tau_N\}$. If τ_i is a periodic task, the jobs generated by τ_i along time are also statically specified. We will focus particularly on the jobset generated by τ because the source energy is variable and the scheduling sequence varies according to different time operational conditions.
We will consider a generic set of jobs $J = \{J_1, J_2, ..., J_n\}$. J_i is completely specified by four-tuple (r_i, C_i, E_i, d_i). It respectively gives the release time, worst case execution time (expressed in time units and normalized to processor computing capacity), worst case energy consumption (expressed in energy units such as joule) and deadline of J_i. Energy consumed by the processor for executing any job is not necessarily proportional to the computation time of that job. J_i has to receive C_i units of execution and E_i units of energy in the interval $[r_i, d_i)$. $d_{Max} = \max_{0 \leq i \leq n} d_i$ and D is the longest relative job deadline i.e. $D = \max_{1 \leq i \leq n} (d_i - r_i)$. Energy consumed by the jobs on $[t_1, t_2)$ is denoted $E_c(t_1, t_2)$.

III. FP-H: the Optimal Scheduling Algorithm

A. Fixed priority scheduling

Under fixed priority scheduling, all the jobs which are generated by a given periodic task inherit the same fixed priority statically assigned to that task. The jobs then compete for the processor at run time using their priority. In this paper, we are concerned with fixed task priority assignment or fixed job priority assignment. The scheduling issue in RTEH systems is twofold: first how to assign priorities to jobs and second how to decide when to execute a job and when to let the processor in the sleep mode. Consequently, priority assignment issue and processor management issue have to be

considered independently to design energy harvesting aware scheduling algorithms.
Hereafter, we consider the following definition of optimality: Suppose that a priority assignment is given, say PA. A scheduler will be said optimal if it finds a valid schedule (i.e. with no deadline missing) for every PA-schedulable jobset (i.e. at least one valid schedule exists for this jobset with the priority assignment PA).

B. Clairvoyance and idling requirements

Energy limitation and variation may affect respect of timing requirements and schedulability of jobs. Figure 3 enables us to show that deadline missing can happen if a scheduling scheme does not consider future production and future consumption of energy to take its online decision. In other words, an efficient scheduling algorithm should be clairvoyant.

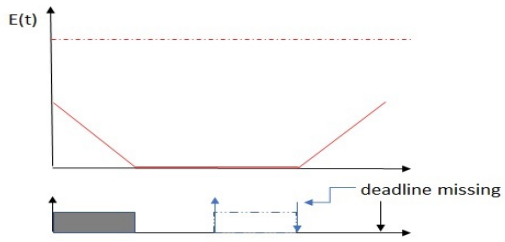

Fig. 3. Deadline missing because of no clairvoyance.

For example, Figure 3 depicts a job which executes and consequently consumes energy. Thereafter, another job releases and no sufficient energy is available for it so as to complete its execution by its deadline whatever its priority. Considering future arrivals of jobs as well as their energy requirements is required to decide whether to put the processor in the active mode or in the sleep mode. The schedule depicted in Figure 4 illustrates that We have to examine not only the energy that is currently stored in the SM but also the energy that will be produced by the environmental source and consumed by jobs in future.

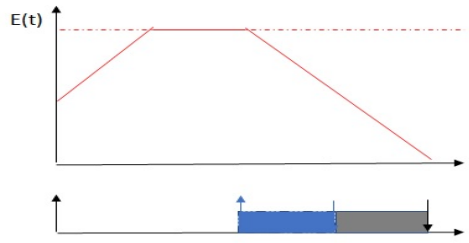

Fig. 4. Smart power management with no deadline missing.

C. Central definitions

Consider a FP-H schedulable jobset at the current time t_c. We assume that J_c is the highest priority job ready for execution at t_c. We address the question of deciding if J_c can be executed while avoiding a future deadline missing caused by energy starvation. We have shown that this requires computation of a variable called *preemption slack energy* at t_c. If the preemption slack energy is zero, the processor has to be put in the sleep mode for battery recharging and the duration will be equal at most to the slack time computed at t_c. Let us recall that the slack time is the maximal time interval for the processor to stay idle while guaranteeing no deadline missing.

Let us show how to identify the slack time at current time t_c for a generic PA-schedulable jobset which is not necessarily issued from periodic tasks. For clarity, let us recall useful definitions.

- A priority level-j busy period is a time interval during which jobs, of priority π_j or higher, that were released at or after the start of the busy period, but before its end, are either executing or ready to execute.
- Assume that for each priority level-j, there is only one job, say J_j. Let α_j be the set of time instants called scheduling points of job J_j defined as :

$$\alpha_j = \{t;\, t = r_k,\, \pi_k > \pi_j,\, r_j < r_k < d_j\} \cup \{d_j\} \quad (1)$$

- The slack time of job J_j at time t_c, denoted by $ST_j(t_c)$ is the largest duration such that, if jobs with a priority higher than J_j start within a delay at most equal to $ST_j(t_c)$, J_j will meet its deadline.
- The slack time of jobset J at time t_c, denoted by $ST(t_c)$ is the largest duration such that, if the execution of any job starts with a delay at most equal to $ST(t_c)$, all jobs will meet their deadline.

It was proved in [15] how to compute the slack time of job J_j at time t_c and finally the slack time of jobset J at time t_c.

Proposition 1:

$$ST_j(t_c) = \max_{\substack{t \in \alpha_j \\ t > t_c}} (t - t_c - \sum_{\substack{\pi_k \geq \pi_j \\ t_c < r_k < t,}} C_k) \quad (2)$$

Proposition 2:

$$ST(t_c) = \min_{t_c \leq r_j < d_c} ST_j(t_c) \quad (3)$$

From Proposition 1, it clearly results that if there is some time t_c and some job J_j such that $ST_j(t_c) = 0$, the processor has to be busy at t_c in order to guarantee the time validity of the FP-H schedule. To guarantee the respect of all job deadlines, we need to compute the level-j slack time for all priority levels and take the smallest one as the slack time of the jobset, as stated in Proposition 2.

In a similar way to the time domain, we introduce the notion of slack energy for the energy domain:

- The slack energy of job J_j at time t_c, denoted by $SE_j(t_c)$ is the largest amount of energy which may be consumed

from t_c by jobs with a lower priority that guarantees no deadline missing for J_j caused by energy starvation.
- The preemption slack energy of the jobset J at current time t_c is the largest amount of energy which may be consumed by the currently active job J_c that guarantees no deadline missing caused by energy starvation for future higher priority jobs

.

Proposition 3:

$$SE_j(t_c) = \max_{\substack{t \in \alpha_j \\ t > t_c}} (E(t_c) + E_p(t_c, t) - \sum_{\substack{\pi_k \geq \pi_j \\ t_c < r_k < t,}} E_k) \quad (4)$$

Proposition 4:

$$PSE(t_c) = \min_{\substack{\pi_j > \pi_c \\ t_c < d_j < d_c}} SE_j(t_c) \quad (5)$$

If there is some time t_c and some job J_j such that $SE_j(t_c) = 0$, Proposition 3 says that the processor has to either be idle or execute a higher priority job at t_c in order to guarantee the energy validity of the FP-H schedule i.e. absence of deadline missing caused by energy starvation. Consequently, if there is some time t_c such that $PSE(t_c) = 0$, the processor has to be idle from time t_c.

D. Informal description

As the classical fixed priority scheduler PA, the energy aware scheduler FP-H still preemptively schedules the jobs according to a given priority assignment rule PA. Before authorizing the highest priority job to be executed by the processor, the residual energy in the battery must be sufficient to supply it. Furthermore, the energy consumed by it must not provoke energy starvation for future higher priority jobs. In addition to the interference with other jobs with higher priorities a job may be postponed because of necessary processor idling. This delay occurs because the processor cannot continue working without injuring energy starvation.
The online problems the power management procedure has to deal with are first to decide whether the processor can enter the active mode and if so, second to compute the maximum amount of energy that may be consumed for preserving the energy feasibility of all the subsequent higher priority jobs. Let us note that, by definition of priority, J_c cannot have any impact on the execution of any job with priority less than π_c. Thus, let us introduce the so-called *preemption slack energy of the jobset J at time t_c* denoted by $PSE(t_c)$. An idle time is forced if $PSE(t_c) = 0$. Otherwise, the processor is authorized to be busy consuming at most $PSE(t_c)$ units of energy until J_c be finished or preempted by a higher priority job.

E. The scheduling scheme

Hereafter, $L_r(t_c)$ is the list of uncompleted jobs in J ready for execution at t_c. The FP-H scheduling algorithm uses the following rules:

- At any time t_c, the priority order defined by the FP priority assignment rule allows to select the future running job in $L_r(t_c)$.
- The processor is idle in $[t_c, t_c + 1)$ if one of the following conditions is satisfied:
  1) $L_r(t_c) = \emptyset$.
  2) $L_r(t_c) \neq \emptyset$ and $E(t_c) = 0$.
  3) $L_r(t_c) \neq \emptyset$ and $PSE(t_c) = 0$
- The processor is busy in $[t_c, t_c + 1)$ if one of the following conditions is satisfied:
  1) $ST(t_c) = 0$.
  2) $E(t_c) = C$.
- The processor can be busy or idle in $[t_c, t_c + 1)$ otherwise.

These rules say that the processor should be necessarily idle if the energy storage is deplenished or if execution of any job will imply energy starvation for at least one future occurring job (i.e. the system has no preemption slack energy). Recharging power is wasted only when there are no ready jobs and the storage is full at the same time. Decisions of FP-H are based on computation of $PSE(t_c)$. This supposes to know short term prediction of the energy harvesting rate . Methods have been developed which aim at providing prediction with high accuracy, low computation complexity and low memory requirement [16]. The energy storage stops to recharge if the slack time is zero. Methods for computation of slack time are reported in [4]. FP-H assumes here that the energy thresholds are respectively 0% and 100% of the storage capacity. Nevertheless, other threshold may be specified, thus shortening duration of the discharging and recharging phases and increasing the number of switches between active mode and power-down mode of the processor.

F. Optimality statement

Theorem 1: The scheduling algorithm FP-H is optimal for any priority assignment rule.

Theorem 1 was established in [15]. It says the following: if FP-H does not success in building a valid schedule for any FP-schedulable jobset J, then no other processor management policy that respects the priority assignment FP will be able to build a valid schedule.

IV. Illustrative example

The following is an example to illustrate the energy aware scheduler FP-H and the performance gain compared to the classical scheduler FP. We consider an RTEH system composed of the jobset J whose time and energy parameters are in Table I. The hardware platform has an energy storage unit with capacity $C = 10$ mJ. The harvesting power is set to 0 mW from initial time instant 0 during 7s and then, 2 mW.

Assume that energy supply is not limited. Figure 5 depicts the FP schedule where all the jobs are executed in the ASAP mode. As no deadline is missing, we say that the jobset is time-schedulable under the FP priority assignment rule with $\pi_1 > \pi_2 > \pi_3 > \pi_4$.

Job J_i	π_i	C_i (s)	r_i (s)	d_i (s)	E_i (mJ)
J_1	1	1	7	13	10
J_2	2	1	5	12	10
J_3	3	1	6	14	2
J_4	4	1	0	15	2

TABLE I
PARAMETERS OF THE JOBS

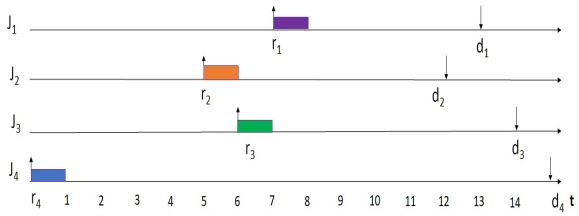

Fig. 5. FP scheduling with no limitation in energy supply

Consider now that we apply a classical scheduler under a fixed priority assignment rule. Figure 6 shows us that, firstly a non idling scheduler cannot face to energy depletion, and second a non clairvoyant scheduler cannot avoid deadline missing caused by energy starvation.

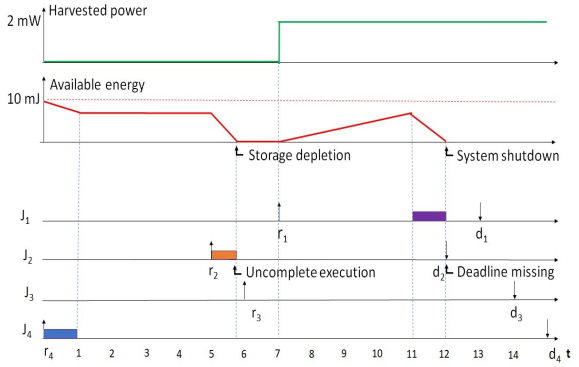

Fig. 6. FP Scheduling with no clairvoyance.

The valid FP-H schedule is depicted on Figure 7. To decide whether or not J_4 may start execution at $t_c = 0$, the energy availability is checked by computing the preemption slack energy with equation (4). As $SE_2(0) = 0$, the processor has to idle so as the energy storage does not deplenish. The slack time is computed so as to determine the latest time when to stop the idle mode according to equations (2) and (3). We have $ST(0) = 10$. However, as the energy storage is full when J_2 releases at time 5, J_2 immediately starts execution since there is no advantage in delaying the job. J_2 finishes at time 6 where the energy storage is totally depleted. Since $ST(6) = 6$, the processor is put in the idle mode until time 12 where J_1, J_2 and J_4 may be executed as late as possible with no deadline missing. Note that J_3 and J_4 execute by consuming energy at the same speed as that it is produced. This example illustrates that the premature consumption of only two units of energy by J_4 was enough to make the system faulty when it was possible to postpone its execution with no deadline violation (see Figure 6).

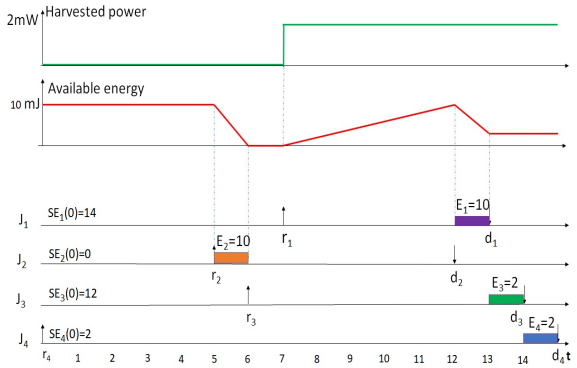

Fig. 7. The optimal FP-H schedule

V. IMPLEMENTATION CONSIDERATIONS

Probably the greatest difference between energy-neutral and conventional battery-operated systems is their behavior in situations where a system has deplenished its energy storage. At this point, an energy-neutral system switches into a sleep mode and wakes up as soon as enough energy has been harvested to resume task execution. In contrast, a classical battery-operated system reaches the end of its lifetime. For this purpose, an energy-neutral system needs to provide means to safely store its state in persistent memory after having detected that no more energy is available in the storage. Furthermore, the system must have detailed knowledge about the energy consumption of tasks, here called WCEC (Worst Case Energy Consumption. In addition, the system must have detailed the minimum energy that should be available in the storage to autorize wake up and guarantee the system to execute tasks for at least a given amount of time.

VI. CONCLUSION

In this paper, we addressed the scheduling issue in a self-powered device with timeliness requirements expressed in terms of deadlines. The central challenge is to make this system energy neutral despite intermittent and variable production of environmental energy used to supply it. The paper reported a new fixed priority based scheduler which smartly defines the busy and sleep periods of the processor. The resulting intermittent computing framework allows to achieve an energy-neutral mode of operation with no wasted energy, no energy starvation and no deadline missing whenever possible. Most RTOS (Real Time Operating Systems) use fixed-priority scheduling where the developers assign each task a suitable static priority level to indicate its relative urgency. Consequently, integration of this new scheduler will permit RTOS to evolve and to meet the rising demand of energy harvesting technology.

REFERENCES

[1] F. Yildiz. *Potential ambient energy harvesting sources and techniques*. The Journal of Technology Studies, 35(1), pp. 40–48, 2009.

[2] K. S. Adu-Manu, N. Adam, C. Tapparello, H. Ayatollahi, and W. Heinzelman, Energy-harvesting wireless sensor networks (ehwsns): A review, ACM Transactions on Sensor Networks (TOSN), 14 (2) (2018).

[3] D. Ma, G. Lan, M. Hassan, W. Hu, and S. K. Das, Optimizing sensing, computing, and communication for energy harvesting iots: A survey, IEEE Communications Surveys and Tutorials, 22 (2) (2019) 1222-1250.

[4] J. W. S. Liu, Real-Time Systems, Prentice Hall, 2000, 592 pages.

[5] R. Davis, L. Cucu-Grosjean, M. Bertogna and A. Burns, A review of priority assignment in real-time systems, Journal of systems architecture, 65 (2016) 64-82.

[6] C.-L. Liu and J.-W. Layland, Scheduling algorithms for multiprogramming in a hard real-time environment, Journal of the Association for Computing Machinery, 20 (1) (1973) 46-61.

[7] A. Allavena and D. Mosse, Scheduling of Frame-based Embedded Systems with Rechargeable Batteries, Workshop on Power Management for Real-Time and Embedded Systems, 2001.

[8] C. Moser, D. Brunelli, L. Thiele, L. Benini, Real-time scheduling for energy harvesting sensor nodes, Real-Time Systems, 37 (3) (2007) 233-260.

[9] S. Liu, Q. Qiu and Q. Wu, Energy Aware Dynamic Voltage and Frequency Selection for Real-Time Systems with Energy Harvesting, in: Proc. of Design, Automation and Test in Europe, 2008, pp. 236-241.

[10] S. Liu, J. Lu, Q. Wu and Q. Qiu, Harvesting-Aware Power Management for Real-Time Systems with Renewable Energy, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, (2011) 1-14.

[11] Y. Abdeddaïm, Y. Chandarli and D. Masson, The Optimality of PFPASAP Algorithm for Fixed-Priority Energy-Harvesting Real-Time Systems, in: Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS), 2013.

[12] M. Chetto and A. Queudet, A Note on EDF Scheduling for Real-Time Energy Harvesting Systems, IEEE Transactions on Computers, 63 (4) (2014) 1037-1040.

[13] M. Chetto, Optimal Scheduling for Real-Time Jobs in Energy Harvesting Computing Systems, IEEE Transactions on Emerging Topics in Computing, 2 (2) (2014) 122-133.

[14] M. M. Sandhu, S. Khalifa, R. Jurdak and M. Portmann, Task Scheduling for Energy-Harvesting-Based IoT: A Survey and Critical Analysis, IEEE Internet of Things Journal, 8 (18) (2021) 13825-13848.

[15] M. Chetto, Fixed Priority Scheduling and Energy Neutrality for Autonomous Embedded Real-Time Systems, Technical Report, Nantes Université, January 2023.

[16] A. Kansal and M.B. Srivastava, An environmental energy harvesting framework for sensor networks, in: Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED '03), 2003, pp. 481–486.