POLITECNICO DI TORINO
Repository ISTITUZIONALE

Battery-aware design exploration of scheduling policies for multi-sensor devices

Original
Battery-aware design exploration of scheduling policies for multi-sensor devices / Chen, Yukai; JAHIER PAGLIARI, Daniele; Macii, Enrico; Poncino, Massimo. - ELETTRONICO. - (2018), pp. 201-206. (Intervento presentato al convegno ACM Great Lakes Symposium on VLSI (GLSVLSI) tenutosi a Chicago, Illinois, USA nel May 23-25, 2018) [10.1145/3194554.3194588].

Availability:
This version is available at: 11583/2709747 since: 2020-02-22T22:24:57Z

Publisher:
ACM

Published
DOI:10.1145/3194554.3194588

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

12 October 2024

Battery-aware Design Exploration of Scheduling Policies for Multi-sensor Devices

Yukai Chen, Daniele Jahier Pagliari, Enrico Macii and Massimo Poncino
Department of Control and Computer Engineering
Politecnico di Torino, Turin, Italy

ABSTRACT

Lifetime maximization is a key challenge in battery-powered multi-sensor devices. Battery-aware power management strategies combine task scheduling with dynamic voltage scaling (DVS), accounting for the fact that the power drawn by the device is different from that provided by the battery due to its many non-idealities. However, state-of-the-art techniques in this field do not take into account several important aspects, such as the impact of sensing tasks on the overall power demand, the (operating point dependent) losses due to multiple DC-DC conversions, and the dynamic modifications in battery efficiency caused by different distributions of the currents in the temporal and in the frequency domains. In this work, we propose a novel approach to identify optimal power management solutions, that addresses all these limitations. Specifically, using advanced battery and DC-DC converter models, we propose methods to explore the scheduling space both statically (at design time) and dynamically (at runtime), accounting not only for computation tasks, but also for communication and sensing. With this method, we show that the battery lifetime can be increased by as much as 23.36% if an optimal power management strategy is adopted.

1 INTRODUCTION

Dynamic power management (DPM) of battery-powered multi-sensor devices (ranging from wearable electronics to wireless sensor nodes) has been intensively studied by researchers, and many strategies have been proposed over the years[1--7]. Many of these solutions are battery-aware, in the sense that they specifically take into account the fact that there is a substantial difference between the power consumed by the hardware and the one actually provided by the battery [3--7]. The reason for this discrepancy is twofold: first, a DC-DC converter is typically placed between the battery and the load. Second, a battery cannot provide any arbitrary amount of power with the same efficiency; this is due to the so-called ''rated capacity'' effect, which states that a battery is less efficient in providing increasingly larger currents.

The many solutions in the literature that account for these non-idealities are mostly focused on the power management of the hardware in terms of *scheduling of the computational tasks executed on the microprocessor*[3, 4, 6]. While these works provide relevant general guidelines for task scheduling, they show several shortcomings. First, the great majority of these works focuses only on *computational tasks*, despite the fact that *sensing tasks* normally have significantly larger power demands [7, 8], and thus impact on the battery depletion. Moreover, sensing tasks are typically independent from each other, thus they can be freely parallelized or serialized, if this is beneficial for the battery (e.g. in order to minimize peak currents or to maximize idle intervals). In some cases, the impact of sensing on the total power demand could be so high as to render the many alternative schedules of computational tasks roughly indistinguishable from the perspective of battery lifetime optimization. Second, most works neglect the fact that each system component may work at a different voltage level. In general, each sensor requires a dedicated DC-DC converter, different from that of the micro controller unit (MCU) or core and from that of the RF transceiver. Each of these converters will be working in a different operating point, with consequently different efficiencies. Third, the battery models used in these works (e.g., the one of [9]) are *steady-state*: they are not able to track the instantaneous changes in voltage and state-of-charge (SOC) of the battery, and rather calculate the overall duration of the battery for a given workload. Therefore, most solutions only propose *static* scheduling algorithms, i.e. in which the schedule is fixed throughout the system operation [3, 4, 6, 7]. Moreover, many of these works overrate the so-called recovery effect in batteries, providing solutions that are biased towards the insertion of idle times among operations. However, recent works have shown that in typical Li-Ion batteries the recovery effect is virtually absent [10].

In this work we proposed a novel design exploration method for battery-aware power management in multi-sensor devices, that targets all the aforementioned limitations. Our approach accounts for the impact of all major system tasks, including sensing and communication, and considers the scheduling freedom offered by the independence of sensing operations as a new optimization dimension for battery lifetime optimization. Moreover, it makes use of a DC-DC converter efficiency model that accounts for dependencies on the operating point of each component [11], and of an advanced circuit-equivalent battery model [12], capable of expressing battery non-idealities related to instantaneous current values distribution and frequency spectrum. Thanks to these models, we are able to show that a *dynamic* scheduling solution, in which the scheduling is recomputed at the beginning of every sampling interval, can provide additional lifetime benefits compared to a static approach. In Section 4, we show that scheduling according to our exploration can provide up to 18.48% lifetime increase if computed statically, and up to 23.36% if updated dynamically.

2 MOTIVATION

The workload of a multi-sensor device is structured as a periodic sequence of the following major tasks: sensing, computation, and transmission. Figure 1-(a) shows an example timing diagram in which the sensing tasks are started **in parallel** at the start of the period, and followed by processing and transmission. After these operations (of a total duration of T_{active}) the system enters an idle state for a time $T_{idle} \gg T_{active}$, resulting in a typically very low duty cycle $D = T_{active}/T_{idle}$.

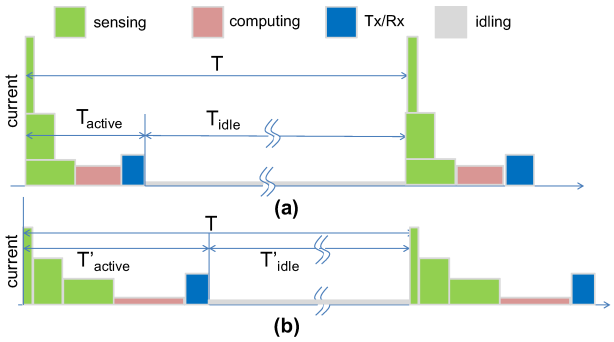

Figure 1: Conceptual workload of the system considered in this work: (a) one in which T_{active} is minimized, and (b) one in which T_{active} is maximized while keeping the current peaks as low as possible.

This schedule is driven by the objective of *maximizing the idle time*. Therefore, multiple sensing operations are run in parallel, and for the same reason the computation task is run at the maximum possible speed in order to shorten T_{active} as much as possible. While this might be reasonable from the perspective of the power consumed by the system, it might lead to sub-optimal results when a battery is involved. As a matter of fact, it is well-known that a battery is particularly sensitive to large current variations [13, 14]. Stacking together multiple sensing tasks, which are incidentally the most power-hungry operations, would unnecessarily deplete battery charge. Similarly, relaxing execution speed of the MCU might help avoid a too strong depletion of the battery. Figure 1-(b) shows an alternative schedule with the sensing tasks serialized and the computing task executed at the lowest possible speed. This yields a duty cycle $D' = T'_{active}/T'_{idle} > D$.

Nevertheless, when the system is in idle state, the power consumption P_{off} is likely to be 2 or 3 orders of magnitude smaller than in the active phase. Therefore, there will be a limit in stretching the activity phase after which an increase of the duty cycle will not be convenient any more. The objective of this work is to provide an insight of such trade-off by exploring different alternatives using a power model of a battery that can accurately track the battery efficiency as a function of current distribution (and not just instantaneous current values) [12] in order to derive the optimal tasks scheduling policy for prolonging the lifetime of designed system.

3 METHOD AND ALGORITHMS

3.1 System Characterization

3.1.1 System Architecture. The device consists of the following components:

- A wireless transceiver implementing some given protocol. We assume that each received and sent message has a fixed-size payload so that transmission requires always the same time T_{tx} and T_{rx}. Power consumed during transmission and reception are P_{tx} and P_{rx} respectively, with $P_{tx} > P_{rx}$. We assume also that the transceiver has approximately zero power when idle. Notice that in our scenario, without loss of generality, we assume that the devices only transmit the sensed information to a central gateway and do not receive any data.
- A set of N_s sensors, each sensing a different quantity. Each sensor takes a time $T_{s,i}$ to complete sensing and requires a power $P_{s,i}$. Notice that we assume that the sensing task includes analog-to-digital conversion.
- A micro controller unit (MCU) that executes a given algorithm on the sensed data. We assume that the MCU has N_p power states associated to different voltage/frequency points. At a given voltage/frequency point $i = 1, ..., N_p$, the MCU takes $T_{c,i}$ to execute and requires a power $P_{c,i}$. Without loss of generality we assume that lower indices represent lower power levels, $T_{c,i} > T_{c,i+1}$ and $P_{c,i} < P_{c,i+1}$.
  Besides these active states, the MCU has an off state in which it consumes a power $P_{off} > 0$ due to static power consumption. In this state the CPU cannot execute instructions.

Table 1 summarizes the key parameters of the components of the system.

Component	Mode	Exec. Time	Power
Transceiver	TX	T_{tx}	P_{tx}
	RX	T_{rx}	P_{rx}
Sensors	Active	$T_{s,i}$	$P_{s,i}$
	Idle	--	$P_{off_{s,i}}$
MCU	Active	T_c	$P_{on}(T_c)$
	Idle	--	$P_{off_{mcu}}$

Table 1: Power/Performance parameters of the system.

3.1.2 Workload Characterization. As described in Section 2, the workload is structured as a periodic sequence of the following major tasks: sensing, computation, and transmission. We assume that there exist two degrees of freedom in the possible temporal sequence of the workload: (1) the scheduling of the sensing operations, and (2) the power state at which the computational phase is executed. Transmission is assumed to take a fixed time. How these two phases are executed will determine the resulting T_{active}, and since T is considered to be fixed (determined by the application constraints), it will also determine the duty cycle D.

3.1.3 Characterization of the Power Domains. An accurate analysis of the impact on the battery of the different workload distributions in a period implies an accurate characterization of the various power domains involved in the system. Considering only the power consumed by each task as in Table 1 to determine the battery depletion in each cycle would not be accurate at all. Conversely, it is essential to consider the actual voltage and current levels of each component.

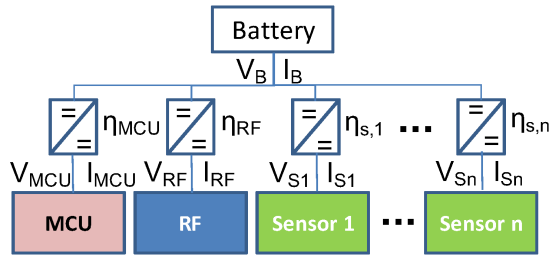

Figure 2: Characterization of the power domains involved.

Figure 2 suggests that each component operates in a different *power domain* with its own voltage level and current consumption [15]. Furthermore, such power level is in general different

from the output power of the battery, and therefore each power domain must be ''adapted'' by an appropriate DC/DC converter. Obviously, if multiple components share the same voltage level, a single DC/DC converter can be shared accordingly. This has important consequences for the assessment of the battery lifetime for two reasons:

(1) the DC-DC converters are not ideal components, and each one does not fully convert 100% of the battery power into usable power. The lost power is dissipated as heat. This is represented by an efficiency factor $\eta < 1$; for instance, $P_{MCU} = \eta_{MCU} \cdot P_{battery}$. This implies that the power consumption by a component actually translates in a larger power demand to the battery P/η.
(2) Due to the implementation of such converters, normally the efficiency decreases for low output currents [15], which, incidentally, is the exactly what we target when we maximize the idle time.

Therefore, besides an accurate battery model that can track current variations and its distribution, a model of DC/DC converters in which efficiency is (at least) sensitive to output current is needed.

3.2 Models

3.2.1 Component Models. From the perspective of the battery energy, all components can be regarded as generic **loads** that consume an instantaneous power given by the product of a current demand and a voltage level. Therefore, components that are assumed to be non power-manageable such as the transceiver and the sensors, are simply characterized by their current and voltage (Figure 2).

Conversely, the MCU has N_p power states, each one corresponding to a given (voltage, frequency) pair and an associated current consumption I_i. The frequency of power state i impacts the computation time $T_{c,i}$, whereas the voltage V_i and I_i determine the power level. Notice that as each power state is associated to a different voltage level, it will result in a different efficiency of the corresponding DC/DC converter.

3.2.2 Battery Model. Analyzing the impact of workload scheduling on battery lifetime requires an advanced battery model, able to accurately account for the effects of load current variations on the usable battery capacity. Specifically, the model must express both the capacity dependency on current *magnitude*, and on current *dynamics* [12]. Indeed, recent works have shown that the available capacity is influenced by the current *value* distribution (e.g. constant and varying load currents with the same average value affect the capacity differently), as well as by its *frequency* spectrum (e.g. periodic load currents with the same amplitude and shape, but different frequencies). To account for all these effects, we employ the circuit equivalent battery model described in [12], shown in Figure 3.

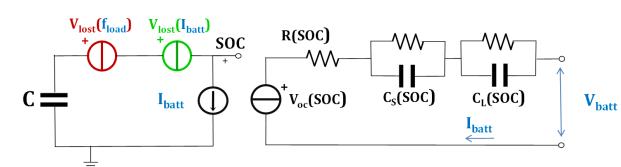

Figure 3: Battery model incorporating current magnitude and frequency dependence.

The circuit consists of two main sections. On the left, the capacitor C represents the nominal battery capacity in Ah; the current generator models the load current (I_{batt}), and the two voltage generators express the dependency of capacity on load current magnitude and frequency respectively. On the right side, the voltage-controlled oscillator models the dependence of the battery open-circuit voltage V_{oc} on the SOC of the battery. Finally, the RC network represents the internal impedance of the battery. All the model parameters can be extracted directly from the battery datasheet [12].

3.2.3 Converter Model. DC-DC converters connecting different power domains to the battery are characterized by an efficiency factor $\eta < 1$, defined as:

$$\eta = \frac{P_{out}}{P_{battery}} = \frac{P_{out}}{P_{battery} + P_{loss}} \quad (1)$$

We use the efficiency model introduced in [11], which expresses η as a quadratic function of the converter input voltage, output voltage, and current.

3.3 Exploration Algorithm

We envision two scenarios of increased accuracy and complexity. A **static** scenario, in which the schedule is determined off-line based on the effect of each possible schedule on the battery charge, with the latter assumed to be 100% charged. Such optimal schedule is then kept fixed throughout the lifetime of the battery. In the **dynamic** scenario, the schedule is re-determined at the beginning of each new sampling interval; this is because, as the battery gets discharged, its capability of handling large current decreases, and therefore it is possible that the best schedule changes according to the battery state-of-charge.

3.3.1 Static Scenario. In this case, the optimal scenario is obtained by evaluating off-line all possible schedules and assessing their impact on the SOC of battery. Since the degrees of freedom in the schedule are the arrangement of the sensing tasks and the power/speed of the computation task, the evaluation simply consists of an exhaustive exploration of all possible schedules. This is feasible since both the number of possible sensors and the power states of the MCU are small (< 10, and typically about 3--5).

In spite of the small number of sensors, the possible combinations of schedules increases rapidly. The number of combinations is 14 for 3 tasks, 81 for 4 tasks, and 821 for 5 tasks. Figure 4 shows the 13 possible schedules of three sensing tasks, shown for simplicity as all requiring the same time and current. The top left configuration does all the sensing in parallel, whereas those on the second row are different combinations of a serialized sensing.

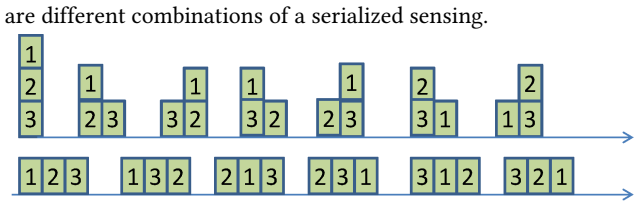

Figure 4: The 13 possible schedules of three sensing tasks.

Notice that in principle we should consider all possible orders of executions, since the battery is sensitive to different current demand over time. However, as shown in [4], it can be proved that among all possible schedules of a set of tasks, those in non-increasing order of power consumption are always the most battery-efficient

ones. This well-known result implies that some schedules will be dominated by non-increasing ones and can simply be disregarded. For instance in Figure 4, three of the schedules in the first row (i.e., (3), (1 + 2), (2), (1 + 3), and (1), (2 + 3)) can be neglected in the enumeration. As a result, the actual number of possible schedules to be explored depends on the specific power consumption associated to each task and cannot be enumerated analytically.

The search algorithm is thus a straightforward exhaustive search in the space of the possible schedules:

```
1.  foreach combination of sensing tasks
2.    foreach power mode of the MCU
3.      build schedule S
4.      if (S is not-increasing in current values)
5.        S ➡ current and voltage waveform (I(t), V(t))
6.        transform (I(t), V(t)) into I_batt(t), V_batt(t)
          using the converter efficiencies
7.        apply I_batt(t), V_batt(t) to the battery and
          evaluate the SOC loss ΔSOC
8.        if (ΔSOC < ΔSOC_min)
9.          S_opt ← S
10.       endif
11.     endif
12.   endfor
13. endfor
```

3.3.2 Dynamic Scenario. In this scenario, the only difference with respect to the previous case is that the calculation of the battery SOC loss resulting from the schedule ΔSOC (Line 7 in the algorithm) is done not by calculating it on a fully charged battery but rather accumulating them over each cycle. More precisely if the optimal schedule in cycle i results in a loss ΔSOC_i for a battery with capacity C_i at the beginning of that cycle, in the next cycle the power demand impact of the schedules will be applied to a battery of capacity $C_{i+1} = C_i - \Delta SOC_i$.

Notice that the implementation of this scenario requires that at the beginning of an interval some code calculates or estimates such optimal, SOC-aware scheduling and arranges the tasks accordingly. Notice that in this case the timing profile Figure 1 should be slightly modified to insert a small computation/estimation task at the beginning of the period.

The calculation of the optimal schedule would clearly have a significant overhead and an exhaustive exploration could not be executed in a real-life implementation; we therefore report results for the dynamic scenario mostly as a reference, in order to show how accounting for the battery SOC in the calculation might affect the overall battery lifetime and the resulting schedule.

Given the unfeasibility of an on-line implementation of the search, an interesting challenge becomes the extraction of a rule of thumb for the selection of the scheduling that could help at least approaching the optimal solution. For instance, a qualitative indication about preferring a compact (with minimal T_{active}) vs. a stretched schedule as a function of the battery SOC. In the result section we will try to infer some possible policy that could be quickly estimated at the beginning of each cycle.

4 EXPERIMENTAL RESULTS

4.1 Simulation Setup

4.1.1 System used in the simulation. We used SystemC to specify a multi-sensor device equipped with four sensors, namely, infrared, wind speed and direction, gas and PM2.5 sensors. The multi-sensor device is based on the system proposed in [7]; we added two additional sensors in our experiment to enlarge the space for exploration in order to extend our analysis to a more complex system. We assume that the MCU takes care of the scheduling of the sensors activities by controlling their power state (idle, active).

Table 2 lists all components in such system, and their corresponding electrical characteristics, power supply voltage and current, within different modes.

Component	Mode	Voltage (V)	Current (mA)
CC2420 [16]	TX	1.8	18.8
	RX	1.8	17.4
	Idle	0.0	0.00
Infrared Sensor [7]	Active	5.0	10.0
	Idle	2.5	0.0001
Wind Sensor [17]	Active	12.0	50.0
	Idle	5.0	0.0010
Gas Sensor [7]	Active	5.0	168.9
	Idle	2.0	0.0015
PM2.5 Sensor [18]	Active	5.0	220.0
	Idle	2.0	2.00
Atmega128l [19]	Active	2.0 – 3.0 – 5.0	1.0 – 6.0 – 18.0
	Idle	3.0	0.002

Table 2: Electrical parameters of the designed system.

The different voltage and current values of the 8-bit Atmega1281 microcontroller shown in the last row of table 2 mean that it has three different power modes, they are 2.0V with 1.0mA, 3.0V with 6.1mA and 5.0V with 18.0mA.

Parameters	UR16650ZT	CSC93-3B0024
Rated Capacity	2100mAh	2000mAh
Nominal Voltage	3.7V	3.9V
Weight	41.0g	17.0g
Cut-off voltage	3.0V	2.0V

Table 3: Manufacturer's parameters of the selected batteries.

4.1.2 Battery used in the simulation. We selected two popular rechargeable Lithium batteries for our experiments, namely, the PANASONIC UR16650ZT and the ELECTROCHEM CSC93--3B0024. Table 3 shows the key specifications of both batteries. They have similar nominal capacities, 2100mAh and 2000mA respectively. The weight of battery is a critical point when design such kind of multi-sensor devices, therefore, the main advantage of CSC93-3B0024 is its lightness (less than half of the weight of the Panasonic UR16650ZT), while its weak point is its lower power rating, which makes it more sensitive to the larger currents compared to the UR16650ZT.

We used the methodology described in [14] to extract a circuit-equivalent model of the two batteries that includes also the dependence on current magnitude and load frequency as shown in section 3.2. This model cannot track the capacity recovery-effect during idle intervals, which was widely used in many previous works on scheduling of wireless sensor nodes. However, as shown in recent research [10], the recovery effect is overrated and often misinterpreted; the experimental analyses of [10] do not show any evidence of the recovery effect in Lithium-Ion batteries.

4.1.3 DC-DC converter used in the simulation. Each component has its individual DC-DC converter to connect to the battery in the system. We selected LTC3789 buck-boost switching mode DC-DC converter from Linear Technology [20], whose main applications are distributed DC power systems and high power battery-operated

devices. Similar to the battery model, we used the manufacturer's information to extract the parameters for the converter loss model[15]. We adopted the model proposed in [11] and whose conversion efficiency is function of input voltage, output voltage and current as shown in section 3.2.

4.2 Static scheduling

In this experiment, we first simulate the UR16650ZT. As illustrated in [14], the rated capacity effect of such battery becomes very evident when battery current exceeds $\approx 0.9A$. For this reason, only schedules in which all the sensors work in parallel can strongly affect the available capacity of this battery, while the others ones only limitedly stress the battery. Figure 5 shows the distribution of the battery lifetime for all possible scheduling policies. Since the PANASONIC UR16650ZT has good capability to provide higher current without losing much available capacity, the difference between shortest lifetime and longest lifetime is only 4.53%.

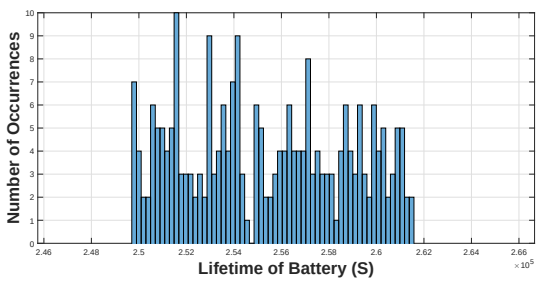

Figure 5: UR16650ZT Lifetime Distribution.

The longest lifetime shown in the Figure 5 is $2.6156 \cdot 10^5$ seconds, whose corresponding optimal scheduling is illustrated in Figure 6. The PM sensor and GAS sensor work in series at first, then WIND sensor and IR sensor work in parallel. Although the MCU has three different power states related to different voltage and current, the optimal scheduling does not select the low power state. The reason is that the current in 5.0V power state is only 18mA which does not significantly deplete the available capacity of the battery. Therefore, the optimal policy select the 5.0V power state to save computation time then increase the idle period.

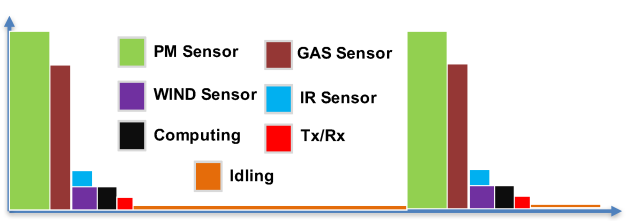

Figure 6: The Optimal Static Scheduling for UR16650ZT .

Our experiments also reveal that, as expected, the worst scheduling policies are the ones with all sensors working in parallel, because the aggregate currents impact the usable capacity. Due to space limits, we do not show the temporal diagram of worst case; Figure 7 shows the battery current profile for both worst- and best-case scheduling over four cycles. Notice that these current profiles do not match the current values listed in Table 2, because the actual battery current is determined by (i) the voltage ratio between battery and each component, and (ii) by the efficiency of DC-DC converter.

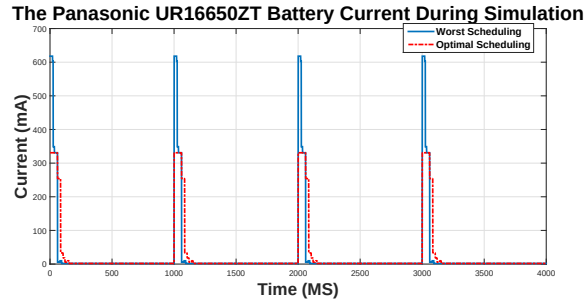

Figure 7: Battery Current Profiles of Worst/Best Schedules.

We then used the CSC93-3B0024 and run all possible schedules. Figure 8 exhibits the battery lifetime distribution for all possible schedules. The difference between the best and worst scheduling is now more pronounced due to the higher sensitivity of the CSC93-3B0024 to larger currents. The best/worst battery lifetimes are $2.5952 \cdot 10^5/2.1155 \cdot 10^5$, respectively, a 18.48% difference. The scheduling that yields the longest battery duration almost identical to the one for the UR16650ZT shown in Figure 6, with the minor difference that the execution by the MCU uses 3.0V power state as opposed to the 5.0V of the UR16650ZT which scales down one level compare to 5.0V power state, but the ones of MCU works in 2.0V and 5.0V power states have very tiny difference compare with MCU works in 3.0V power state that is reason why there are three scheduling policies occupy the longest lifetime column in Figure 8. Compare with PANASONIC , such more pronounced difference verifies ELECTROCHEM CSC93-3B0024 is more sensitive to the current values, which leads our exploration process becomes more useful. The battery current profiles within worst- and best-case of ELECTROCHEM are similar to ones shown in Figure 7, the differences are current values.

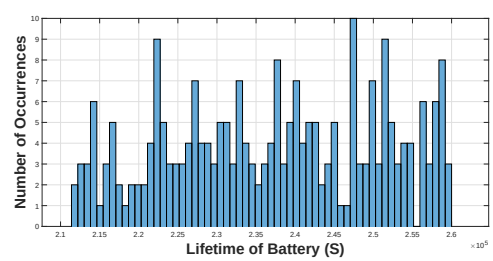

Figure 8: CSC93-3B0024 Lifetime Distribution.

In the above analysis we set the period T at 1 s. While this is a reasonable value, for some applications it might be case that the system does not need to sense data that frequently. For the designer it is important to know for which values of T the impact of different schedules is relevant. Intuitively, as T gets larger, the active section of the period (sensing + computation + transmission) will take an increasingly smaller portion and the power consumption will be dominated by idle power. We therefore ran the same exploration

of static schedules for different lengths of the cycle time and for both batteries. We used values of T of 1s, 60s, 600s and 1800s and calculated the normalized range (max-min) of the battery lifetime. Figure 9 shows the results: the lifetime range expectedly decreases with larger cycle times.The percentage value of y-axis in Figure 9 means the different battery lifetimes between optimal scheduling policy and worst scheduling policy, it illustrates the space of our design exploration reduced when the period is increased.

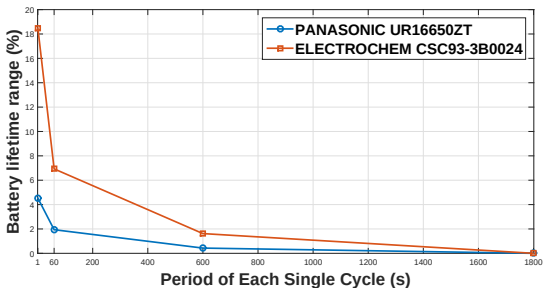

Figure 9: Lifetime difference between optimal and worst scheduling with different period values.

This analysis is essential to assess if and when the scheduling is a relevant dimension of the power optimization of the device. Our analysis show that for cycle times longer than a few minutes the impact of the scheduling in terms of battery duration becomes negligible (below 2% for the weakest battery). Therefore, our exploration of schedules is relevant for multi-sensor devices with activation periods in the order of seconds to minutes. Notice that this consideration applies to **all** methods in the literature discussed in Section 1; however, such analysis was not carried out before.

4.3 Dynamic Scheduling

For the dynamic scheduling scenario, we assume that at the beginning of each cycle, the MCU executes a small task in which it explores the optimal schedule in order to get smallest SOC loss, as described in Section 3.3.2. We assume that the MCU consumes approximately the same power required for the processing of the sensing data in the state that consumes the most power in Table 2.

Since a cycle-specific schedule aims at the smallest SOC loss per cycle, the corresponding lifetime of battery will be longer than the static scheduling scenario. For instance, when T = 1s, our results show that the longest battery lifetimes of UR16650ZT and CSC93-3B0024 are $2.6999 \cdot 10^5$ and $2.7604 \cdot 10^5$ seconds, respectively, the lifetimes are prolonged of 3.266% and 6.368% compared to the longest lifetimes for the same batteries under the optimal static schedule.

We tested above two batteries using T = 1s and T = 60s. Results show that the resulting schedules are identical for both periods and for both batteries and change three times during the battery lifetime. Initially, the optimal schedule is the same as the static schedule shown in Figure 6; in the first change, wind and IR sensor eventually start to operate serially; the second and third changes correspond to the MCU switching to a lower power state. Notice that the time-points at which the changes of scheduling occur are different for the two batteries with two different T, as reported in Table 4. As expected, the CSC93-3B0024 battery always changes schedule before the earlier than UR16650ZT due to its higher sensitivity to higher current values.

$T = 1s$			
Battery	1st Change	2nd Change	3rd Change
UR16650ZT	68133s	204399s	233715s
CSC93-3B0024	54499s	163156s	201244s

$T = 60s$			
Battery	1st Change	2nd Change	3rd Change
UR16650ZT	733320s	2380380s	2777160s
CSC93-3B0024	641040s	2014860s	2289720s

Table 4: Times of schedule change with $T = 1s$ and $T = 60s$.

5 CONCLUSIONS

In this paper, we present a framework to evaluate static and dynamic schedules to optimize the lifetime of multi-sensor devices powered by batteries. We use advanced battery and DC-DC converter models in our experiments to get more accurate results. The two proposed scheduling algorithms have been applied to two different batteries and two different working situations for estimating the increment of lifetime. Our results show that the optimal schedule according to our exploration can achieve a 18.48% lifetime increment when using a static schedule, and as much as 23.36% for the dynamic schedule.

REFERENCES

[1] Simunic, T., Benini, L., Glynn, P. and De Micheli, G., ‘‘Dynamic power management for portable systems,’’ Proc. MobiCom 2000, pp. 11-19.
[2] Shin, Y., Choi, K. and Sakurai, T., ‘‘Power optimization of real-time embedded systems on variable speed processors,’’ Proc. IEEE/ACM ICCAD 2000, pp. 365-368.
[3] Luo, J. and Jha, N.K., ‘‘Battery-aware static scheduling for distributed real-time embedded systems,’’ Proc. IEEE DAC 2001, pp. 444-449.
[4] Chowdhury, P. and Chakrabarti, C., ‘‘Static task-scheduling algorithms for battery-powered DVS systems,’’ IEEE Trans. VLSI Syst., vol. 13, no. 2, pp. 226-237, 2005.
[5] Yu, X., Xiaosong, X. and Wenyong, W., ‘‘Priority-based low-power task scheduling for wireless sensor network,’’ Proc. IEEE ISADS 2009, pp.1-5.
[6] Wei, K., Zhang, W., Yang, Y., Song, G. and Zhang, Z., ‘‘Battery-Aware Task Scheduling for Energy Efficient Mobile Devices,’’ IEICE Trans. Fundamentals, vol. E97-A, no. 9, pp. 1971-1974, 2014.
[7] Hong, S., Kim, D. and Kim, J.E., ‘‘Battery aware real time task scheduling in wireless sensor networks,’’ Proc. IEEE RTCSA 2005, pp. 269-272.
[8] Schonle, P., et al, ‘‘A multi-sensor and parallel processing SoC for wearable and implantable telemetry systems,’’ Proc. IEEE ESSCIRC 2017, pp. 215-218.
[9] Rakhmatov, D., Vrudhula, S. and Wallach, D.A., ‘‘A model for battery lifetime analysis for organizing applications on a pocket computer,’’ IEEE Trans. VLSI Syst., vol. 11, no. 6, pp. 1019âĂŞ1030, 2003.
[10] Narayanaswamy, S., et al, ‘‘On battery recovery effect in wireless sensor nodes,’’ ACM TODAES vol. 21, no. 4 (2016): pp. 60:1-60:28.
[11] Park, S., Wang, Y., Kim, Y., Chang, N. and Pedram, M., ‘‘Battery management for grid-connected PV systems with a battery,’’ Proc. ACM/IEEE ISLPED 2012, pp. 115-120.
[12] Chen, Y., Macii, E. and Poncino, M., ‘‘A circuit-equivalent battery model accounting for the dependency on load frequency,’’ Proc. IEEE DATE 2017, pp. 1177-1182.
[13] Doerffel, D. and Sharkh, S.A., ‘‘A critical review of using the Peukert equation for determining the remaining capacity of lead-acid and lithium-ion batteries,’’ J. Power Sources 155, no. 2, (2006): pp. 395-400.
[14] Chen, Y., Macii, E. and Poncino, M., ‘‘Frequency domain characterization of batteries for the design of energy storage subsystems,’’ Proc. IFIP/IEEE VLSI-SoC 2016, pp. 1-6.
[15] Lee, W., Wang, Y., Shin, D., Chang, N. and Pedram, M., ‘‘Power conversion efficiency characterization and optimization for smartphones,’’ Proc. ISLPED 2012, pp. 103-108.
[16] http://www.ti.com/lit/ds/symlink/cc2420.pdf
[17] http://www.fischer-barometer.de/katalog/dokumente/meteo/E451213_14.pdf
[18] http://breathe.indiaspend.org/wp-content/uploads/2015/11/nova_laser_sensor.pdf
[19] http://www.atmel.com/products/microcontrollers/avr/default.aspx
[20] http://cds.linear.com/docs/en/datasheet/3789fc.pdf