Energy Scheduling for Task Execution on Intermittently-Powered Devices

Mohsen Karimi
mkari007@ucr.edu
University of California Riverside
Riverside, California

Hyoseung Kim
hyoseung@ucr.edu
University of California Riverside
Riverside, California

ABSTRACT

Intermittently-powered embedded devices (IPDs) are getting widespread attention these days. However, running periodic real-time tasks on these devices remains a challenging problem due to the lack of support for data freshness guarantees, timekeeping, and schedulability analysis. Especially, while many sensing tasks require long atomic operations for data acquisition from sensors, most prior work on IPDs assumes compute-only workloads and disregards such sensor operations. In this paper, we present a new energy scheduling scheme to execute periodic real-time tasks with atomic sensing operations. Our scheme keeps track of time and ensures the periodic execution of sensing tasks while efficiently utilizing intermittent power sources. We provide schedulability analysis to determine if a task is schedulable in a given charging setup, and extend this idea for scheduling multiple tasks. As a proof-of-concept, we design a custom programmable RFID tag device, called R'tag, and demonstrate the effectiveness of our proposed techniques in a realistic sensing application. We compare the baseline approach and the proposed scheme in both simulation and real platforms. Experimental results show that the proposed method outperforms the baseline approach in terms of task scheduling, timekeeping, and periodic sensing.

CCS CONCEPTS

• **Computer systems organization → Real-time operating systems**; **Embedded systems**; **Sensor networks**.

KEYWORDS

energy scheduling, task scheduling, intermittently powered devices, real-time systems

1 INTRODUCTION

Battery-less intermittently-powered devices (IPDs) have gained much interest due to their potential to facilitate wireless sensor networks and Internet of Things (IoT). These devices, powered by intermittent power sources such as sunlight, heat, vibration, and radio signals, have diverse applications including smart home, agriculture, and health monitoring [1, 4]. Owing to the fact that they do not have a battery that needs to be replaced, they can continue to run even for decades without much maintenance effort. Furthermore, they can be deployed in extreme environments where batteries do not perform well.

Data freshness and timely execution is the key requirement of many sensing tasks, which is also the case for those running on IPDs. If a certain event in the environment is sensed long after the actual occurrence, the reaction may be either ineffective or dangerous. IPDs often divide long operations into multiple subtasks and execute them over multiple charging/discharging cycles. Depending on energy availability, data collected earlier may become stale in the middle of processing, and in such a case, it would be better to collect new data instead of processing the remaining part of the stale data. For example, in blood sugar monitoring for a diabetic to release proper amount of insulin, any late or improper action would cause a catastrophic damage.

Another challenge is ensuring the long atomic (indivisible) execution time required for data acquisition from sensor peripherals. Many sensors, especially environmental and chemiresistive sensors, take a long time, several seconds to minutes, to initiate and to collect reliable data from them. Hence, IPDs should be able to run for a relatively long time without intermittent power disruptions. If the power goes off, all the operation should start over again from the beginning since the state of sensor peripherals cannot be saved and resumed.

Prior work on IPDs has focused on "forward progress" guarantees, by checkpointing intermediate results in non-volatile memories [2, 13] or by providing a programming language [6] to divide the tasks into fragments while preserving data consistency. Capybara [2] uses multiple capacitor banks to mitigate the atomic execution time problem. However, similar to other approaches like [8, 11], the device goes off whenever it exhausts the energy and loses the notion of time so it is unable to schedule sensing tasks with periodic execution requirements. MayFly [6] partially solves this problem with an external circuitry for timekeeping of up to 17 minutes in power failures. InK [19] provides an event based kernel to manage periodic sensing on IPDs. It relies on an external timer that keeps track of time while the microcontroller (MCU) is in low power mode and uses interrupts to wake up the device. However, none of these approaches have considered scheduling policies for periodic sensing tasks, e.g., when to start execution for each task, and provided analytical foundations for timing guarantees, e.g., task schedulability under intermittent power supply.

In this paper, we propose an energy scheduling scheme for real-time task execution to address the aforementioned limitations of prior work. Our proposed scheme provides an energy model that is specifically designed to capture the charging and discharging characteristics of IPDs as well as the periodic execution requirements of sensing tasks. Unlike prior work, our scheme controls the charging level of capacitors, which allows storing more energy

EWiLi'19, October 2019, New York, NY, USA

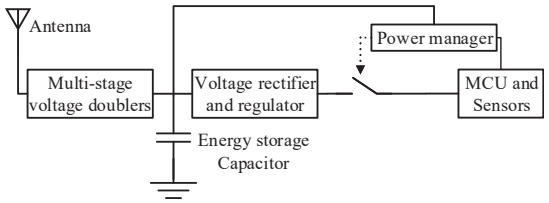


Figure 1: Block diagram of an RFID energy harvesting device

than what is required to just turn on the MCU. Our scheme computes the required level of voltage to complete the given amount of computation, including sensor operations, and makes the device wait until that level is reached. This enables the execution of a long indivisible sensor operation, which was not possibly doable by prior work. Furthermore, our scheme enables timekeeping, which is important to check the freshness of obtained data from sensors. We analyze the schedulability of a single periodic task under our scheme and provide illustrations on multi-task scenarios. We develop a prototype hardware and software system for evaluation. Experimental results from simulation and real hardware indicate that our scheme outperforms the baseline approach and can satisfy the timing requirement of periodic real-time tasks.

2 SYSTEM MODEL

In this section we describe the hardware and software properties of the system which we use in the rest of the paper.

2.1 Hardware Characteristics

Each energy harvesting device has the following main units: energy harvester unit, energy storage unit, power management unit, energy conversion unit, and processing unit. The energy harvester unit converts the energy coming from the power source (e.g. light, wind, vibration, and RF signal) to a type of energy (e.g. voltage and electrical current) that can be accumulated in the energy storage unit. The storage unit, usually consisting of capacitors, can store the energy to be used to power the system. In the energy conversion unit, voltage converters, rectifier, and regulator are used to convert an energy source to the desired voltage for electrical circuits. The power management unit controls when to store energy and when to use the energy to power up the system. Finally, in the processing unit, MCUs and sensors are used to perform the desired operations.

Figure 1 shows the general block diagram of a radio-frequency identification (RFID) energy harvesting device. In this figure, each unit is specifically designed to harvest the RF energy to be used for the MCU and sensors. When the stored energy, i.e., the voltage of the capacitor, reaches a specific level (called *power-on threshold*), the power management unit switches to turn on the MCU and sensors. When the voltage goes down to the minimum voltage level for the MCU and the rest of the circuit (called *power-off threshold*), the system is turned off until the capacitor voltage is recharged to the power-on threshold.

Depending on how energy is consumed, there are multiple types of energy discharging in IPDs. We will analyze the detailed characteristics of each discharging type under our proposed scheduling scheme in Section 4. Below we give the energy harvesting model used in this paper.

Energy Harvesting Model. To find the voltage equations for the circuit, we assume the energy source to be a fixed power instead of a fixed voltage or current source because what is received from the RFID reader is a fixed power. We consider a parallel resistor R_p to the capacitor, which consists of the equivalent storage capacitor, and also the rest of the circuit's resistor in parallel with capacitor. Therefore, the harvesting circuit would become an RC model with a fixed power source and the voltage of the capacitor can be calculated by solving the following equation:

$$\frac{P}{V} = C_s \frac{dV}{dt} + \frac{V}{R_p} \tag{1}$$

By solving this equation, the voltage of the capacitor at time t can be calculated as:

$$V = \sqrt{PR_p - e^{\frac{-2t}{C_s R_p}} * \left(PR_p - V_0^2\right)} \tag{2}$$

where V_0 is the voltage of capacitor at $t = 0$, and P is the power received from the power source after going through all the voltage doubler stages. Based on Eq. (2), the time to reach from voltage V_0 to V, where $V > V_0$, can be calculated as:

$$t_{charging} = \frac{C_s R_p}{2} Ln\left(\frac{PR_p - V_0^2}{PR_p - V^2}\right) \tag{3}$$

The power source is not always available to an IPD. We characterize the availability of the power source with two parameters, C_c and T_c, which mean that for a period of T_c, the reader charges the device for at least C_c time units. This can represent both stationary and mobile wireless chargers, e.g., [3].

2.2 Software Characteristics

We consider periodic real-time tasks, where a task τ_i is a characterized by its worst case execution time C_i, relative deadline D_i, and period T_i. For simplicity, we assume the initial arrival time of all tasks is zero. We also consider the implicit deadline which means $D_i = T_i$. Accordingly, a task i can be expressed as $\tau_i := (C_i, T_i)$.

Due to the nature of sensing applications on IPDs, tasks are scheduled in a non-preemptive manner. Hence, a higher priority task cannot preempt a lower priority task when the lower priority task is already running on the device. Note that non-preemptive scheduling is common in IPDs, as shown in [3, 19].

3 CHALLENGES

To understand the challenges of IPDs in sensing applications, we conduct a case study using WISP, a well-known RFID-harvesting device [15]. Based on [5], the power received by WISP can be calculated as:

$$P_r = \frac{G_s G_r \eta}{L_p} \left(\frac{\lambda}{4\pi(d+\beta)}\right)^2 P_t \tag{4}$$

where G_r is the reception antenna gain, G_s is the transmission antenna gain, η is the rectifier efficiency, L_p is the polarization loss, λ is the wavelength of the RF signal, d is the distance from tag to reader, P_t is the transmission power, and β is the adjustment parameter to adjust Friis' free space equation for short distance.

The parameters for the above equation are as follows. We use an RFMAX S9028PCL polarized directional antenna which has the transmission gain of $G_s = 8dBi$. WISP has a linear dipole antenna;

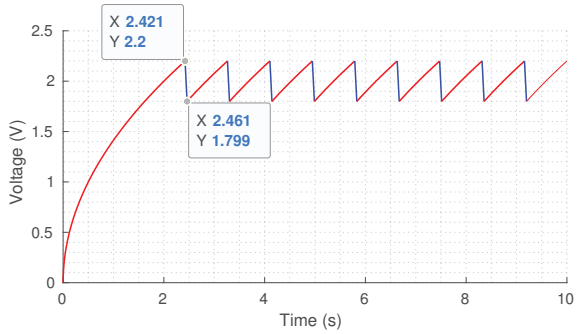


Figure 2: Charging and discharging cycles of an IPD

therefore, the reception gain is $G_r = 2dBi$. WISP works on 915MHz frequency so wavelength would be about $\lambda = 0.327$. For the other parameters, we use the values of WISP reported in [5]: $\beta = 0.2316$, $\eta = 0.125$, and $L_p = 2$. Since all the parameters except d are constant, we can rewrite Eq. (4) to:

$$P_r = \alpha \left(\frac{1}{d+\beta}\right)^2 P_t \tag{5}$$

where α and β are constant. As it can be inferred from Eq. (5), the power received by the device is fixed when it is located at a fixed distance to the reader. For the distance of $60cm$ and the power transmission of $1W$, the power reception would be about $1mW$.

Let us consider an energy harvesting circuit following the RC model discussed in Section 2 with $R_p = 1G\Omega$ and $C_s = 100\mu F$. Based on this harvester and the WISP parameters obtained above, we now analyze the charging and discharging characteristics of an IPD. Figure 2 shows the voltage level of the device when it is always getting charged by a stationary RFID reader. In this figure, red lines are charging cycles (i.e., the device is turned off) and blue lines are discharging cycles (i.e., the device is on and can execute tasks). The power-on and power-off thresholds, both of which are determined by the hardware, are 2.2V and 1.8V, respectively.

We discuss three major challenges observed from this case study. The first one is that in sensing applications, tasks often need to execute atomically without any power disruptions. As can be seen in Figure 2, the maximum execution time allowed for a task is about only 40ms. Therefore, any task that needs more than 40ms of continuous execution can never complete its job. For example, most gas sensors are designed to work at a specific temperature and have an internal micro-heater to maintain that temperature. Since the heater takes time to reach the desired temperature, any intermittent power loss would lead to a failure in sensing operation and sensor data could not be obtained at all.

The second challenge is that tasks on an IPD are unable to capture samples at specific times because the device can sample data only when it receives enough power from the power source, e.g., discharging cycles (blue lines) in Figure 2. The problem becomes more complicated if the RFID reader is not always available, e.g., mobile readers and line-of-sight obstructions. In many sensing applications, capturing samples at specific time is required to guarantee the validity of data freshness and the resulting processing.

The last challenge is timekeeping. The device goes off after each discharging cycle and it loses the track of time it is working. Even if a real time clock is integrated into the device, it would also go off after each power loss. Therefore, any application that needs the notion of time progress over long periods would not be able to run on IPDs. Without that, it is very hard (if possible at all) to schedule periodic sensing tasks and check the freshness of obtained data.

4 PROPOSED METHOD

In this section, we first present our energy model that captures the energy demand and supply of periodic tasks running on IPDs, and then present our scheduling scheme.

4.1 Modeling Energy Demand and Supply

We categorize the sources of discharging into three types: *decaying*, *processing*, and *waiting*. First, decaying occurs when the device is not receiving any power from the power source and the voltage threshold of the storage unit is below the minimum voltage needed for turning on the MCU (power-on threshold). In this case, since the circuit is not ideally open circuit and the parallel equivalent resistor exists in the capacitor, the device loses some energy gradually. Second, processing occurs during the time when the capacitor voltage is above the minimum threshold, the power management unit turns on the system, and the MCU is executing tasks. Lastly, waiting happens when the device is turned on but is put into low power mode. It is worth noting that the waiting-induced discharging does not occur in most prior work but we explicitly model it due to our scheduling scheme given in the next subsection. Specifically, our scheme accumulates energy beyond the power-on threshold by putting the device in low power mode and delaying the execution of tasks until it gets enough energy to run tasks with long atomic operations. More details will be described later.

For simplicity, we assume all the discharging rates to be linear. Due to the fact that the frequency of the MCU is remained fixed, the task execution time is independent of the supply voltage so we use m_D, m_P, and m_W to denote the discharging rates of decaying, processing, and waiting, respectively. During the waiting time, the system consumes some amount of energy but the power reception from the power source is assumed to be higher than the power consumption in waiting time. In other words, the capacitor voltage can increase in waiting time if the power source is available. This is true for most commercial MCUs since the power consumption in lower power mode is orders of magnitude lower than that in active mode, e.g., 0.4μA vs. 100μA in MSP430.

The charging rate of the device can also be approximated to be linear by a slope of m_c, where m_c can be calculated based on Eq. (3) when V_0 is the minimum voltage threshold of the device and V is the maximum voltage that the capacitor can hold based on its specifications. Therefore, the charging slope can be calculated as:

$$m_c = \frac{V_{min\_th} - V_{max}}{\frac{C_s R_p}{2} Ln\left(\frac{PR_p - V^2_{min\_th}}{PR_p - V^2_{max}}\right)} \tag{6}$$

where C_c and T_c are the charging time and charging period of a power source, respectively.

Thus, the worst-case accumulation voltage during each charging period, T_c, can be calculated as:

$$\Delta V = \frac{m_c * C_c - m_d * (T_c - C_c)}{T_c} * \Delta t \tag{7}$$

where $m_d = \max\{m_W, m_D\}$ is the worst-case discharging rate, and m_W and m_D are the discharging slope of voltage drop during waiting and decaying time, respectively. If we consider an accumulation rate $m_a = \frac{m_c * C_c - m_d * (T_c - C_c)}{T_c}$, the voltage of the capacitor at time t with n periodic tasks can be calculated as:

$$V_{cap}(t) = m_a * t - \sum_{i=1}^{n} \left(\left\lfloor \frac{t}{T_i} \right\rfloor + s_i \right) * C_i * m_{Pi} + V_0 \qquad (8)$$

where m_{Pi} is the processing discharging rate for a task τ_i. s_i is 1 or 0 if τ_i's last released job for its period at t has been executed or not, respectively. s_i is a runtime variable that is set to 0 at the beginning of each period and 1 when the task finishes its execution for that period. Figure 3 shows how s_i changes for three tasks. In reality, each job of a task may contain several segments with different discharging rates. However, in this work, the largest discharging rate among the segments is considered as the whole job's discharging rate. Therefore, the worst-case discharging rate of all the jobs of the same task τ_i is considered to be equal to m_{Pi}.

4.2 Single Task Scheduling

Based on the aforementioned energy model and voltage calculations, we first assume there is only one periodic task in the system and analyze the schedulability of the single task case. We will show in the next subsection that how this analysis can be extended for multi-task systems.

Under our scheduling scheme, when the power management unit turns on the device, the voltage level required for task execution and the waiting time for the system to reach that voltage are computed. Then, a timer is programmed to wake up the MCU after that amount of time and then MCU goes to low power mode. There are multiple ways to implement the waiting time for task execution. It can be implemented on the MCU itself by using the low power mode with timer capabilities, e.g., LPM3 in TI MSP430. Another way is to add an external ultra low-power programmable real-time clock, e.g., 14nA with Ambiq AM08xx RTC [12], so that it can wake the MCU up by an interrupt. In the latter case, the MCU can be put into a deeper sleep mode, e.g., LPM4 in MSP430, since the MCU's clock sources and timers can be turned off.

For a single task ($n = 1$), the waiting time for the task can be calculated as follows:

$$W(t) = \max\left\{ \frac{V_{cap} - (V_{min\_th} + m_{P1} * C_1)}{m_a}, t - T_1 * \left\lceil \frac{t}{T_1} \right\rceil \right\} \qquad (9)$$

where V_{cap} is the current voltage of the capacitor that can be calculated from Eq. (8).

For a single task to be schedulable, m_a should be positive. This means that the voltage level of the capacitor should increase during the charging period of the power source. In addition, the capacitor voltage should reach the desired voltage level for the task's execution, meaning that the following condition should be met:

$$m_a * T_1 \geqslant m_{P1} * C_1 \qquad (10)$$

If m_a is non-positive, the charging rate is not enough to keep the device on during each charging period. Thus, it will cause the device to be turned off before the start of the next charging period. In this case, we cannot keep track of time and cannot use the aforementioned methods to check the schedulability of the task. The power source needs to be available to the device more frequently, i.e., smaller T_c, or longer in each period, i.e., larger C_c.

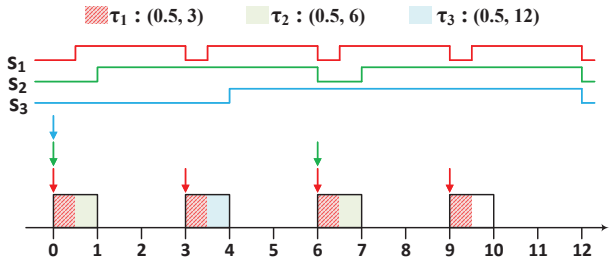


Figure 3: Periodic server example for multi-task scheduling

4.3 Multi-Task Scheduling

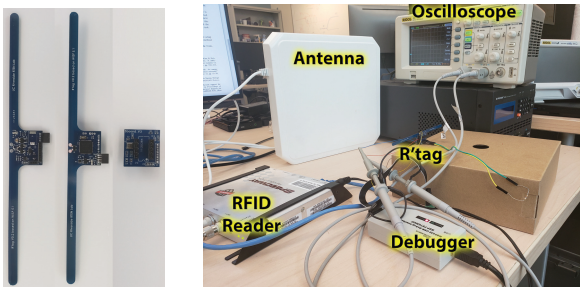
In order to ensure the schedulability of multiple periodic tasks, we propose to abstract the resource demand of the tasks as a *periodic server* [16–18] and schedule them by using the budget of the server. The periodic server is characterized by the two parameters, budget and budget replenishment period, which correspond to the execution time and the period of a periodic task, respectively. Hence, once the server parameters are chosen, the schedulability of the server can be analyzed with Eqs. (9) and (10). It should be noted that Eq. (8) still holds for calculating the voltage of the capacitor at time t.

The scheduling of tasks within the periodic server in an IPD should be done in a non-preemptive manner and is different from that in conventional real-time systems. Thus, existing hierarchical schedulability analysis for preemptive tasks in periodic servers [7, 9, 10, 14] are inapplicable to our problem. Instead, we formulate this as a variant of the bin-packing problem with additional constraints. The items to be packed are the jobs of tasks, and the amount of budget per period is the size of a bin. The number of bins in this problem is given by T_{lcm}/T_s, where T_{lcm} is the hyper-period of all periodic tasks and the server period T_s. Each task τ_i has m items where m is the number of jobs arriving during the hyper-period, i.e., T_{lcm}/T_i. Figure 3 shows an example of using a periodic server resource to execute three tasks, τ_1, τ_2, and τ_3, in a periodic server with period of 3 and budget of 1. An important thing here is that no more than one item (job) from the same task can be allocated to the same bin, and the items should be spaced across bins with respect to their task period. This add complexity to the original bin-packing problem which is already NP-hard. We leave the development of an efficient algorithm to solve this problem as future work. It should be noted that since non-preemptive scheduling is considered, no context switching happens and the only overhead would be the time for selecting the next task, which can be implemented as table-based scheduling [19].

5 IMPLEMENTATION

This section describes the hardware and software setup we use to implement our proposed methods.

We developed an RFID-based energy harvesting tag device, called *R'tag*, to do experiments. It follows the design of WISP [15] and has the same MSP430 MCU, RF circuits, and antennas. In addition,



(a) (b)

Figure 4: (a) R'tag and sensor board, (b) Experimental setup

we integrated the tag with extra external I/Os that can be used for ADC reading for analog and digital measurements. Furthermore, we added a large super capacitor that can be used to store more energy that enables the device to run for longer time.

For the sensing purpose, we designed a pluggable sensor board PCB that can be mounted on R'tag and measure high resistance values generated by chemiresistive sensors which are widely used in sensing applications. It is also equipped with Bosch BME680, an integrated environmental sensor that can measure temperature, pressure, humidity, and total volatile organic compounds in the air. Both R'tag and the sensor board are shown in Figure 4a

For the reader, we use an Impinj Speedway Revolution R420 UHF RFID Reader that can generate up to 30dBm power to charge the tag and -84dBm reception sensitivity to receive the messages sent from the tag. The Ethernet interface is used to connect RFID Reader to the PC to read data received from the tag by the reader. We also use RFMAX S9028PCL polarized directional antenna which has the transmission gain of $G_s = 8dBi$.

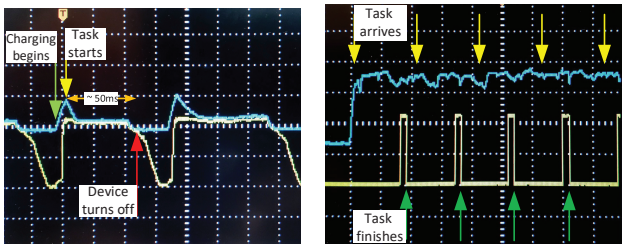
6 EVALUATION

6.1 Experimental Results

Figure 4b shows the experimental setup that is used in this paper. Our setup consists of the Impinj RFID reader which is connected to the PC via Ethernet cable, an R'tag device which is located about 25cm away from the reader's antenna, our sensor board attached to the R'tag, and a $10M\Omega$ resistor connected to the sensor board as an example of a chemiresistive sensor. A $47\mu F$ capacitor is used in the storage unit to store energy during each charging period. The voltage is measured by an oscilloscope attached to R'tag via wiring.

In many chemiresistive sensors, the resistance value is so high that it cannot be measured by simple methods like voltage dividers, due to the current leakage of I/O and ADCs. Hence, we added a small additional capacitor to the sensor board in parallel with the resistance measurement. By checking the time to charge this capacitor and considering the charging rate of the parallel RC circuit, we can estimate the resistance value. In our case, for $10M\Omega$ resistance, it takes up to 120ms to charge the capacitor and completes the measurement. We assume this needs to be done every 1s. Thus, $C_1 = 120ms$ and $T_1 = 1s$.

Figure 5a shows when running the resistance measurement task with the baseline approach which executes the task whenever power is available. The blue line is the voltage of the storage unit capacitor and the yellow line is the voltage of the last regulator that powers



(a) (b)

Figure 5: (a) Task fails execution under the baseline approach. Blue line: capacitor voltage, yellow line: regulator output voltage. x-axis=20ms/div, y-axis=1V/div (b) Task executes periodically under the proposed method. Blue line: capacitor voltage, yellow line: task execution. x-axis=500ms/div, y-axis=1V/div

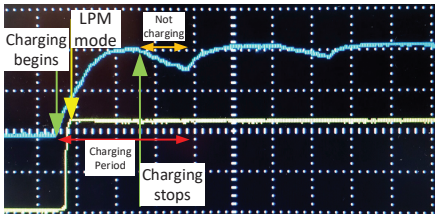


Figure 6: Charging curve of the capacitor when the MCU is in LPM3 mode. Blue line: capacitor voltage, yellow line: regulator output voltage. x-axis=20ms/div, y-axis=1V/div

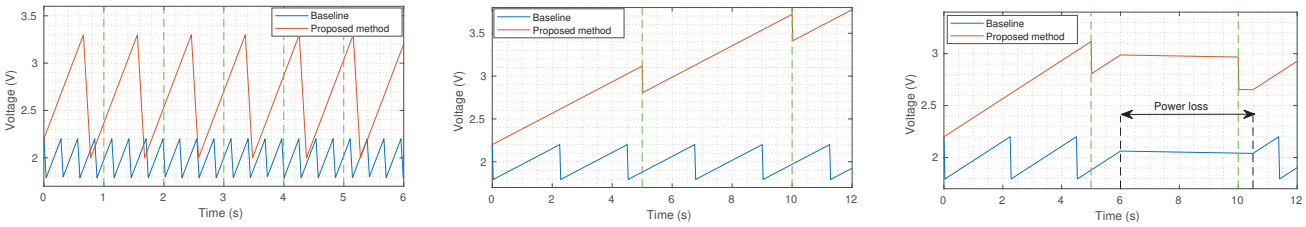
the system. As it can be inferred from the figure, by using the baseline approach, there is no way to successfully complete the task's execution which takes 120ms. Since the task is not resumable in the next power sequence, the MCU restarts the task from the beginning in every power cycle.

To use our proposed scheme, we first need to find the charging and discharging rates. Based on the Figure 5a, the discharging rate of the task can be approximated as 12V/s. To find m_c and m_d, we put the task in sleep mode and find the charging and discharging rate of the capacitor. Figure 6 shows the charging and discharging rates based on the charging cycles in the worst-case scenario. Then Eq. (9) is used to find the next waiting time to run the task. We set one I/O pin to high during the measurement to observe the execution pattern of the task by using the oscilloscope. Figure 5b shows that the task can meets the deadline and the voltage never goes down below the power-off threshold to turn off the device. Therefore, the device always keeps running, the task meets the deadlines, and the MCU can keep track of time.

6.2 Simulation Results

This section evaluates the effect of task execution time and period on the proposed scheme in simulation. We also compare the performance characteristics of the baseline and the proposed method.

We first use the same task parameters that have been used for the previous experiments. Figure 7a shows the results under the baseline and the proposed method. Green lines show the arrival times of the task. As shown in the figure, the baseline fails to run the task continuously for $120ms$, and since the task is not resumable, it restarts in every power cycle. However, with the proposed method, the task can finish its job before the deadline



(a) (b) (c)

Figure 7: Charging and discharging behavior of the tag when running a task with (a) τ:(120ms, 1s), (b) τ:(25ms, 5s), (c) τ:(25ms, 5s) in the presence of power loss

and is always schedulable. One may raise the following question: *what if the capacitor of the baseline is precharged to a higher voltage level?* If that happens, the first job execution of the task might be successful, but since there is no control over the power-on-cycles of the device, the energy will deplete after the first job and the same problem will appear again starting from the second job of the task.

Next, we use different parameters for the task so that it can be finished even with the baseline approach. An additional requirement imposed here is that the task must start its execution at the beginning of each task period. Figure 7b shows the result for the task with $(25ms, 5s)$. While the task under the baseline approach can finish its job, it results in unnecessarily frequent executions since the whole system goes off and restarts in every charging cycle. Furthermore, it cannot start execution exactly at the beginning of the period. On the other hand, the task under the proposed method is executed exactly following the timing requirements, and the system can conserve remaining energy for other tasks.

In practical scenarios, power sources may become unexpectedly unavailable. Figure 7c shows the same scenario as in Figure 7b with a sudden power loss from time 6 to 10.5. Under the proposed method, the device can keep track of time even in the presence of unexpected power losses and can start task execution at the beginning of the next period at time 10. However, the baseline is unaware of elapsed time and simply starts task execution when the device is charged at around time 11.4.

7 CONCLUSION

In this paper, we proposed a new energy scheduling scheme for periodic real-time task execution on intermittently-powered devices (IPDs). We first presented an energy model that considers the energy supply and demand of real-time tasks on an IPD charged by a periodic power source. This model was then used to derive the voltage level of the energy harvesting unit and the schedulability analysis of a task running on the device. To schedule tasks on the device, waiting in low power mode and delayed task execution were proposed. Our techniques enable accumulating enough energy to ensure the execution of tasks with indivisible atomic operations. They also achieve timekeeping in the presence of intermittent power losses.

For evaluation, we designed an RFID energy-harvesting device, R'tag, with a custom sensor board. The device was used to show the effectiveness of our scheme in a sensing application. We showed with experimental results that, given the same charging rate from the RFID reader, the proposed method could schedule the periodic sensing task while the baseline failed to finish even a single job of the task. We also performed simulation with various task parameters and the proposed method outperformed the baseline in task execution, energy usage efficiency, and timing correctness.

There are multiple interesting directions that can be built upon our work. First, for efficient multi-task scheduling on IPDs, one may consider developing algorithms to determine the parameters of the periodic server and to assign tasks to the server instances. Second, the proposed scheduling scheme can be applied to existing runtime and programming language frameworks for IPDs, such as Ink [19] and MayFly [6]. Furthermore, for a large task set with higher energy demands, Capybara [2] can be added to the hardware to enable switching to a higher capacity voltage storage bank.

Acknowledgements. This work is supported by the Research Program (POC2930) of the Korean Institute of Materials Science (KIMS).

REFERENCES

[1] G. Chen et al. A cubic-millimeter energy-autonomous wireless intraocular pressure monitor. In *IEEE International Solid-State Circuits Conference*, 2011.
[2] A. Colin, E. Ruppel, and B. Lucia. A Reconfigurable Energy Storage Architecture for Energy-harvesting Devices. In *ACM SIGPLAN Notices*, volume 53, 2018.
[3] Z. Dong et al. Enabling Predictable Wireless Data Collection in Severe Energy Harvesting Environments. In *IEEE Real-Time Systems Symposium*, 2017.
[4] D. Fan et al. EHDC: An energy harvesting modeling and profiling platform for body sensor networks. *IEEE J. of Biomedical and Health Info.*, 22(1):33–39, 2018.
[5] S. He et al. Energy provisioning in wireless rechargeable sensor networks. *IEEE Transactions on Mobile Computing*, 12(10):1931–1942, Oct 2013.
[6] J. Hester, K. Storer, and J. Sorber. Timely Execution on Intermittently Powered Batteryless Sensors. In *SenSys*, 2018.
[7] S. Hosseinimotlagh and H. Kim. Thermal-aware servers for real-time tasks on multi-core gpu-integrated embedded systems. In *RTAS*, 2019.
[8] H. Jayakumar et al. QUICKRECALL: A low overhead HW/SW approach for enabling computations across power cycles in transiently powered computers. In *IEEE International Conference on VLSI Design*, 2014.
[9] H. Kim and R. Rajkumar. Real-time cache management for multi-core virtualization. In *EMSOFT*, Oct 2016.
[10] H. Kim, S. Wang, and R. Rajkumar. vmpcp: A synchronization framework for multi-core virtual machines. In *IEEE Real-Time Systems Symposium*, 2014.
[11] K. Maeng and B. Lucia. Adaptive dynamic checkpointing for safe efficient intermittent computing. In *OSDI*, 2018.
[12] A. Micro. Ultra-low power rtcs. https://ambiqmicro.com/rtc/, June 2019.
[13] B. Ransford, J. Sorber, and K. Fu. Mementos: System support for long-running computation on rfid-scale devices. In *ASPLOS*, 2011.
[14] S. Saewong, R. Rajkumar, J. P. Lehoczky, and M. H. Klein. Analysis of hierarchical fixed-priority scheduling. In *ECRTS*, 2002.
[15] A. P. Sample et al. Design of an RFID-based battery-free programmable sensing platform. *IEEE Trans. on Inst. and Measurement*, 57(11):2608–2615, 2008.
[16] L. Sha, J. Lehoczky, and R. Rajkumar. Solutions for some practical problems in prioritized preemptive scheduling. In *IEEE Real-Time Systems Symposium*, 1986.
[17] B. Sprunt, L. Sha, and J. Lehoczky. Aperiodic task scheduling for hard-real-time systems. *Real-Time Systems*, 1(1):27–60, 1989.
[18] J. K. Strosnider, J. P. Lehoczky, and L. Sha. The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments. *IEEE Transactions on Computers*, 44(1):73–91, 1995.
[19] K. S. Yildirim, A. Y. Majid, D. Patoukas, K. Schaper, P. Pawelczak, and J. Hester. InK: Reactive Kernel for Tiny Batteryless Sensors. In *SenSys*, 2018.