This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2854374, IEEE Internet of Things Journal

Sentinel: Breaking the Bottleneck of Energy Utilization Efficiency in RF-Powered Devices

Songfan Li, Li Lu, *Member, IEEE,* Muhammad Jawad Hussain, Yalan Ye, and Hongzi Zhu, *Member, IEEE,*

***Abstract*—As a result of the limited available energy, RF-powered devices must be capable of efficiently utilizing scarce energy by planning task execution according to the current harvested energy. However, the energy utilization efficiency is challenging to be improved in RF-powered devices, since sensing the harvested energy consumes a significant amount of energy that should be used for task execution.**

In this study, we propose *Sentinel*, a novel low power method to sense the harvested energy. *Sentinel* is fully delegated to detect the energy for the device, while the device does not participate in the energy sensing. By this means, the computing overhead of the device is reduced. *Sentinel* works with low energy consumption, and functions as a trigger to activate the device when, and only when, the energy reaches an expected energy threshold. We also present a lightweight scheme to set the desired thresholds so that *Sentinel* achieves detecting any expected thresholds. We implement *Sentinel* by off-the-shelf components and conduct experiments to show that *Sentinel* consumes only 5.2% of energy overhead of the general energy sensing technique. With *Sentinel*, we show that the energy utilization efficiency can be improved up to 94.9%, outperforming the best existing works at 64.7% in the WISP platform.

***Index Terms*—Energy, RF-Powered Devices, Energy Efficiency, Energy Polling, Energy Trigger, Energy Utilization Efficiency**

I. Introduction

ONE of the key advantages of RF-powered devices is that they are maintenance-free without the need to replace the battery. RF-powered devices harvest energy from ambient radio frequency (RF) signals including Wi-Fi [1], cellular [2] and ultra high frequency (UHF) RF [3]. After harvesting sufficient energy, RF-powered devices perform sensing and computation, and then upload the results to transceivers through wireless communications. Recently, the development of ultra-low-power backscatter communication has enabled RF-powered devices to directly interact with commercial networked devices (e.g. mobile phones, RFID readers and Wi-Fi access points) with tiny energy budgets [4]–[9]. The maintenance-free feature combined with the low-power communication technology opens up a range of applications in the Internet of Things and mobile scenarios, such as passive implantable devices [10], [11], smart dust [12], maintenance-free structural health monitoring [13], [14] and smart cities [15]–[17].

S. Li, L. Lu M.J. Hussain and Yalan Ye are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, 611731, P.R. China (E-mail: sfli@std.uestc.edu.cn; luli2009@uestc.edu.cn; j19197@gmail.com; yalanye@uestc.edu.cn).

H. Zhu is with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, 200240, P.R. China (E-mail: hongzi@cs.sjtu.edu.cn).

Copyright (c) 2012 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.

A fundamental problem for RF-powered devices is efficient utilization of the harvested energy. The energy is very scarce, since the available RF signals in ambient are three to six orders of magnitude lower than the required energy for RF-powered devices. In this case, the devices have to carefully and efficiently utilize the limited energy to execute tasks in terms of computation, sensing and communication. To this end, the basic strategy is dynamically adapting task executions according to the currently harvested energy. Specifically, RF-powered devices first have to sense how much energy they have (this is called *Energy Sensing*) and then plan the task executions accordingly.

Energy Sensing, however, is the bottleneck of the energy efficiency due to the high energy overhead. For example, we observe that the overhead hits 51.2% of the harvested energy on average over distances in Intel WISP[1] [3], which means that only up to half of the energy can be utilized for task executions. The high energy consumption stems from the fact that existing devices make use of a polling method to be aware of the energy. The polling method involves the usage of an analog-to-digital converter (ADC) as an energy sensing unit to sample the energy hundreds or thousands times per second. The results of the samples are processed by the device subsequently. However, the energy consumed by each ADC sample is power-starving. For example, the consumption equals the energy budget for transferring 27 bits of data through backscatter communication on MOO[1] [18].

Recently, Dewdrop [19] introduced a dynamic polling scheme to reduce the energy overhead of *Energy Sensing*. The scheme continually adapts the polling frequency (i.e. the number of samples per second) according to the current harvesting rate, which lowers the frequency if the ambient energy is weak. Thus, the number of redundant polling operations is reduced, so that the energy consumption of *Energy Sensing* is decreased. However, the consumption is still heavy for RF-powered devices. We show that the energy efficiency of Dewdrop is only $\sim$64.7% in the evaluation section.

In this study, we propose *Sentinel*, a novel low power method to sense energy in RF-powered devices. *Sentinel* derives from an observation that in most applications, the results of the polling are only used for checking whether the harvested energy is above (or below) one of the energy thresholds. If so, the device then executes a related task according to the threshold; otherwise the device keeps waiting for enough energy represented by the threshold. Thus, *Sentinel* abandons the straight-forward polling method. Instead, *Sentinel* attempts

[1]The WISP and MOO are both RF-powered devices.

2327-4662 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2854374, IEEE Internet of Things Journal

to exploit a trigger method, which activates the device when, and only when, the energy reaches a threshold. By this means, the heavy samples in the polling can be totally avoided, so that the energy consumption of *Energy Sensing* is reduced.

The main idea is that *Energy Sensing* of the device is totally delegated to *Sentinel*. The device does not need to participate in *Energy Sensing*, and can stay idle in low power mode before the energy reaching a threshold. In this duration, only *Sentinel* keeps working as a trigger to detect the harvested energy with low energy consumption. When the energy reaches the threshold, *Sentinel* activates the device for task executions. In addition, the expected thresholds in *Sentinel* are configurable to enable the device supporting diverse threshold demands for tasks. To achieve *Sentinel*, we need to address the following two main challenges.

(a) How can Sentinel achieve energy sensing without sampling the harvested energy? Sentinel employs an energy supervising unit to compare the analog harvested energy and the energy threshold directly. The supervisor functions as a low power trigger. Specifically, the supervisor awakens the device by outputting a trigger signal once the energy is above (or below) the threshold. The signal can be directly detected and identified by the kernel control unit ($i.e.$ micro-controller) in the device. Thus, the supervisor enables *Sentinel* achieving low power *Energy Sensing* without the sampling.

(b) How can Sentinel set an energy threshold according to diverse threshold demands for tasks? The expected threshold demands are described by the software in the device, while the monitored thresholds are defined by the hardware in the supervisor. In order to achieve software-tunable thresholds, we leverage digital potentiometers in the supervisor to receive digital threshold demands from the device. Further, according to the demands, the potentiometers adjust the threshold parameters in the hardware to set the new desired thresholds.

Compared to the energy polling, *Sentinel* achieves lower energy consumption in *Energy Sensing* due to two reasons. First, the trigger method enables the device to avoid processing a significant number of energy data from the ADC. Thus, the computing overhead of the device is reduced. Second, the supervisor consumes less energy than the ADC. This is because the supervisor directly detects the energy instead of the power-heavy analog-to-digital conversions in the ADC .

To show the feasibility of our design, we build the prototype of *Sentinel* by commercial off-the-shelf (COTS) components. We conduct experiments on the WISP platform [3] to show that *Sentinel* only consumes up to 11.7 μJ per second to detect the energy, only 0.69 nJ to activate the device and 39.7 nJ to set a threshold. The energy overhead is only 5.2% of the energy polling. With *Sentinel*, the energy utilization efficiency is improved from 64.7% (at the currently best) to 94.9% compared to existing approaches.

Our contribution is three-fold:

- We investigate the relationship between energy utilization efficiency and *Energy Sensing*, and find that the energy consumption of *Energy Sensing* becomes the bottleneck to further improve the efficiency in RF-powered devices.
- We propose *Sentinel* to make the device being aware of the energy with low energy cost. With *Sentinel*, more energy is exploited to execute tasks so that the utilization efficiency is improved up to 94.9% in the WISP.
- We present an approach to design the software-tunable supervisor to support *Sentinel* monitoring any expected thresholds with low energy consumption. With the supervisor, *Sentinel* consumes only 5.2% of energy requirement compared to the energy polling.

The rest of this study is organized as follows. First, we analyze the reason why the energy overhead of *Energy Sensing* becomes the bottleneck to further improve energy utilization efficiency in section II. Then, we illustrate the overview of *Sentinel* and the challenges in section III. Further, we present the proposed design of *Sentinel* in section IV. Next, we implement and evaluate *Sentinel* in section V and VI respectively. Finally, we introduce related works of this study in section VII and conclude this paper in section VIII.

II. Analysis of Energy Utilization Efficiency

Several existing efforts have been proposed to improve the energy utilization efficiency in RF-powered devices. Generally speaking, these efforts mainly focus on the task execution planning based on *Energy Sensing*. We discuss these works in this section to understand why the energy consumption of *Energy Sensing* becomes the bottleneck to further improve the efficiency.

At a high level, all existing works execute tasks based on awareness of the harvested energy. Otherwise, blind executions may incur task failures, which consequently waste the energy. We classify existing works into two categories according to different types of execution planning.

- **BurstExecution** [19]–[22]: The device accomplishes a complete task at one-time once upon accumulating enough energy for the execution. The opportunity of every task execution is represented by a start threshold. Specifically, the task is launched once the energy is above the task's start threshold. *BurstExecution* defines tasks as small programs that can run to completion in a single lifecycle. Thus, this category is suitable for short-time tasks, such as temperature and humidity sensing.
- **Checkpoint** [23]–[26]: For the tasks that could not be completed in a single lifecycle, the device can use *Checkpoint* to protect the tasks against execution failures. *Checkpoint* slices a task into a number of pieces and executes the pieces as many as possible according to the energy. When the power is running up, the device will save the progress of the task execution and recover it once the device is powered up again. Thus *Checkpoint* is proper for long-time tasks, such as encryption algorithms and mathematical calculations.

We observe that the two categories above both require the device being aware of the harvested energy at the time. Specifically, *BurstExecution* executes a task when the energy is above a start threshold; *Checkpoint* needs to save the execution progress before the energy outage.

To make the device energy-aware, the general strategy for *Energy Sensing* is the energy polling with software. The polling employs an ADC component to periodically sample the

2327-4662 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2854374, IEEE Internet of Things Journal

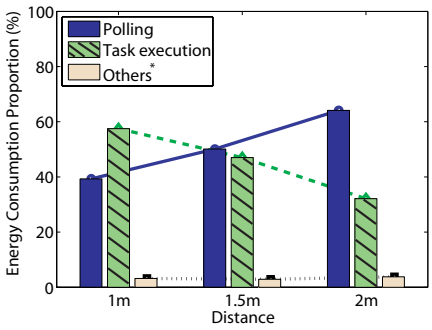


Fig. 1. Energy consumption breakdown in the WISP. We observe that the proportion of the polling increases along with the distance. This happens because the polling overhead is maintained as the polling frequency is fixed apart from distance, while the harvested energy is attenuating along with the distance. *This part of the energy is used by other aspects, for example, system initialization.

harvested energy. The results can be subsequently processed by software. This method is commonly used in most platforms as it can be conveniently implemented by software.

However, the polling approach is not suitable to be employed in RF-powered devices due to the high energy consumption. To understand this, we empirically measure the energy overhead of the polling by conducting an experiment on the WISP platform. In the experiment, we use an RFID reader as an energy source to power up the WISP device at a 1 m distance. We then measure the energy consumption of the polling[2]. We repeat the experiment at different distances and show the result in Fig. 1. We can find that the energy consumption of the polling is very heavy, consuming 51.2% of the harvested energy on average over different distances, and even more than the energy consumption of the task beyond 1.5 m.

If the polling can be cancelled, the saved energy can be utilized by task executions, so that the energy utilization efficiency can be improved. We observe that existing works concentrate upon the task execution planning to improve the energy efficiency. However, the efficiency is difficult to be further improved if the energy consumption of the energy polling cannot be reduced. As the polling is the only general approach to sense energy, the energy consumption of *Energy Sensing* becomes the bottleneck to further improve the efficiency. In the next section, we introduce a brief overview of the proposed *Sentinel* to achieve the low power *Energy Sensing* and the corresponding challenges.

III. Sentinel In A Nutshell

A. Overview of Sentinel

At a high level, the target of *Sentinel* is to detect energy with a low energy overhead. The main concept behind *Sentinel* is best understood with an example. We start with a typical

[2]To accurately measure the energy consumption of the polling, we calculate the consumption by $N_{times} \times E_{ave}$, where N_{times} is the number of the polling per second, and E_{ave} is the average energy consumption per sample. N_{times} can be easily counted by experiments and E_{ave} can be found in the datasheet of the ADC. In addition, the energy consumption of task execution can be measured by the same way.

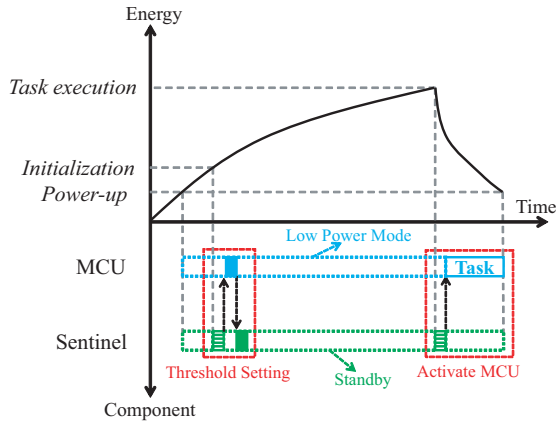


Fig. 2. A typical working sequence diagram of *Sentinel*. Once the energy reaches the initialization threshold, *Sentinel* activates the MCU to select a task to be executed. The MCU then sets an expected start threshold for the task execution to *Sentinel*, and switches into LPM waiting to be awakened. Further, *Sentinel* keeps standby to detect the harvested energy with low energy consumption, and then activates the MCU to execute the task once the energy reaches the threshold. The chief advantage of *Sentinel* is that the MCU can avoid processing a significant amount of energy data and can just stay in LPM to reduce the energy consumption.

working sequence of *Sentinel* with *BurstExecution* to have an overview (the sequence of *Checkpoint* is very similar to this, only adding a threshold for checkpointing before a task execution). As illustrated in Fig. 2, we show the harvested energy over time at the top part. At the bottom part, the sequence diagram of *Sentinel* is demonstrated. In the example, we consider three energy thresholds, *i.e. power-up*, *initialization* and *task execution*. We describe the sequence in the order of time as follows.

- **When the energy reaches *power-up*.** The threshold *power-up* is the natural minimum energy (voltage) requirement for the device to work. At this time, the microcontroller unit (MCU) and *Sentinel* are both powered up, but will not immediately run for the system initialization as the harvested energy is small. Thus, the MCU stays in low power mode (LPM) to accumulate energy. The MCU in LPM does nothing with the exception of being activated by external signals. In addition, *Sentinel* keeps in standby mode to detect the energy with low energy consumption. In standby mode, *Sentinel* works but only performs energy sensing and awaits threshold setting without other actions.
- **When the energy reaches *initialization*.** The threshold *initialization* is the default threshold of *Sentinel*. Once *initialization* is reached by the energy, *Sentinel* detects this event and outputs a trigger signal to activate the MCU. Then, the MCU switches to active mode and selects a task nominated for the execution from a task set. Next, the MCU sets a new threshold to *Sentinel* according to the demand of the selected task. Finally, the MCU switches to LPM to await being activated by *Sentinel*, while *Sentinel* continues the standby mode to detect the harvested energy.
- **When the energy reaches *task execution*.** The threshold *task execution* is the new threshold set by the MCU for

2327-4662 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2854374, IEEE Internet of Things Journal

starting the task execution (different tasks may request diverse start thresholds). At this moment, *Sentinel* activates the MCU to wake up and execute the selected task.

From the above example, we highlight that *Sentinel* is leveraged to sense energy for the device with low energy consumption. As the concern of the energy overhead, *Sentinel* employs an energy supervisor as the energy sensing unit instead of the ADC. Different from the power-starving analog-to-digital operations in the ADC, the supervisor works as a trigger that directly compares the input energy and the energy threshold. In addition, the supervisor also achieves the function of activating the MCU once the energy reaches the threshold, which simply generates a trigger signal that can be directly identified by the COTS MCUs.

Sentinel achieves lower energy consumption than the energy polling due to two reasons. First, the computing overhead of the MCU is less. The MCU in the polling has to work in active mode for processing a significant number of energy data from the ADC. In contrast, the MCU in *Sentinel* can avoid the processing of the energy data and can remain in LPM to save energy in energy sensing. Second, the energy sensing unit is more lightweight. The ADC consumes huge amounts of energy to convert analog energy values (voltage) to digital energy data, for example, 165.8 μJ per second for energy polling in the WISP. Conversely, the supervisor detects analog energy values directly without the analog-to-digital conversion, and therefore consumes less energy than the ADC. We show the energy consumption of the supervisor is only 5.2% of the overhead of the energy polling in section VI-B.

The rest of this study focuses on achieving three important objectives to realize the above design. (1) **Energy detection in standby:** how does *Sentinel* detect the harvested energy in standby mode? In addition, since *Sentinel* keeps standby most of the time, it is essential to investigate how to reduce the standby energy consumption. (2) **Activating the MCU:** how does *Sentinel* trigger the MCU once the energy reaches a threshold? We discuss the three parts in the next section. (3) **Dynamic threshold setting:** how does the MCU set a desired threshold to *Sentinel* according to the demand of a task?

B. Challenges of Sentinel

1) Challenge 1: Uncertain Input Energy: In this portion, we answer the question why *Sentinel* employs an energy supervisor to detect the energy rather than the ADC.

It seems that the ADC can be used to achieve *Sentinel* with the advantage of not modifying the existing hardware. Assuming that the incoming energy keeps constant and stable, we might exploit simple and lightweight algorithms[3] to predict how long to wait for the harvested energy reaching a threshold. If so, many redundant ADC operations can be avoided, so that the energy consumption is reduced. By this means, *Sentinel* may be capable of detecting the event of the energy reaching a threshold by employing the ADC.

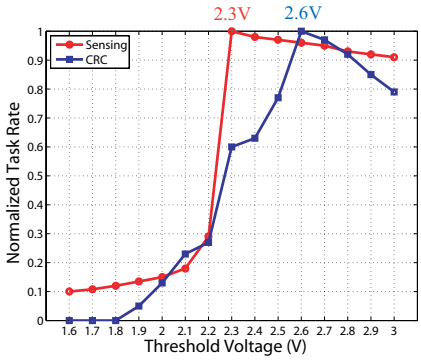


Fig. 3. The task rates of two tasks over different thresholds. With the results, we can find that the 2.3 V threshold is the best threshold for the sensor operating, while the 2.6 V threshold is the best one for the CRC computing. This happens because a low threshold may cause execution failures, while a high threshold may incur extra charging delays.

Unfortunately, the assumption above is impractical because the harvested energy varies all the time in RF-powered devices. The input energy varies mainly due to two reasons. First, RF multipath and shadowing are caused by unknown ambient obstacles, which result in uncertain incident RF power of the device. Second, varying input efficiency incurs uncertain harvested energy as the impedance of the harvester changes during the usage. For example, the authors in [18] investigate a varying empirical curve of harvesting efficiency on the platform MOO.

The uncertain input energy means that it is difficult to achieve energy predictions by simple algorithms. Furthermore, complex algorithms are too burdened to be implemented in RF-powered devices. For example, Freeha Azmat [27] presents an RF energy prediction model with 85% prediction accuracy, but the model requires huge computations to run machine learning algorithms. Thus, the usage of the ADC with prediction algorithms is not a valid solution to achieve *Sentinel*.

Solution. *Sentinel* employs an energy supervisor to detect the harvested energy. The supervisor contains a comparator circuit to compare the harvested energy and the defined threshold. The result is indicated by the comparator's output, which can be treated as a trigger signal to activate the MCU. Thus, the supervisor gains the merits of saving computing resources in the MCU and detecting the energy no matter how uncertain the ambient is. In other words, the supervisor achieves the objectives of both detecting the energy and activating the MCU. We illustrate the principle of the supervising circuit and the merits in section IV-A.

2) Challenge 2: Dynamic Threshold Demands: Sentinel detects energy thresholds for task executions, in which different tasks normally need diverse thresholds. For example, a sensing task needs to be executed at the threshold[4] of 2.2 V in the WISP [19]; a computing task will be executed at 2.6 V [24].

[3]In the charging stage, the time to wait is given by $time = \frac{E_{TH}-E_{now}}{P_{harvest}}$, where E_{TH} is the expected threshold, E_{now} is the currently harvested energy and $P_{harvest}$ is the assumed constant harvesting rate. In discharging stage, the time can be calculated through $time = \frac{E_{now}-E_{TH}}{P_{task}-P_{harvest}}$, where P_{task} stands for the power consumed by a task.

[4]The relationship between energy E and voltage V is described by $E = \frac{1}{2}CV^2$, where C is the capacitance of the reservoir (capacitor).

2327-4662 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2854374, IEEE Internet of Things Journal

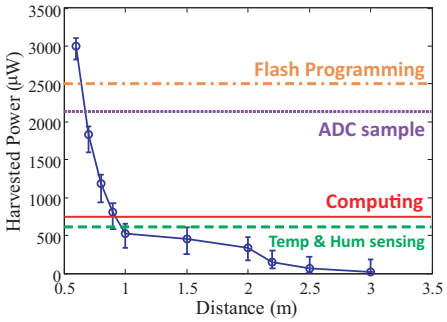


Fig. 4. Harvested power in the WISP. The energy decreases along with the distance increasing, while the energy needing for task executions keeps constant. We also mark some tasks' power consumption to highlight the energy limitation in RF-powered devices.

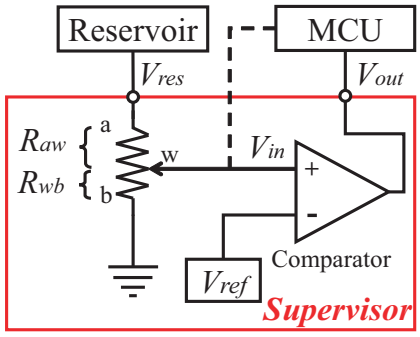


Fig. 5. The proposed design of the supervisor in *Sentinel*. The supervisor mainly contains three components, $i.e.$ a comparator, a voltage reference and a variable resistor. The comparator compares the input voltage V_{in} and the voltage reference V_{ref}. The results are represented by the high/low voltage on the output V_{out}. The variable resistor is digitally controllable by the MCU to divide the input voltage for threshold setting.

We observe that it is inefficient to execute all tasks by a single start threshold, despite some RF-powered devices, like WISP, making use of a fixed single threshold supervisor. We conduct an experiment to explain the reason for the low efficiency. In the experiment, we consider two tasks. The first task operates a humidity and temperature sensor [28] by *BurstExecution* planning, while the second task computes Cyclic Redundancy Check (CRC) over 2 KB data by *Checkpoint* planning. We increase the start threshold of each task using 0.1 V steps and record the task rate (completed tasks per second). The results are normalized for better reading and are shown in Fig. 3.

From the experiment, we observe that every task has a corresponding start threshold for the best performance in terms of throughput (task rate). For instance, the sensing task achieves the highest task rate at a 2.3 V threshold. The CRC task achieves the best performance at a 2.6 V threshold. This brings the requirement of *Sentinel*'s capability on dynamically changing the detected threshold according to diverse threshold demands of tasks.

In existing off-the-shelf supervisors, however, the detected threshold is fixed by the hardware supervising circuit, and is difficult to be adjusted with low energy consumption. Some of the supervisors realize dynamic threshold changing schemes by directly adjusting the voltage reference represented by the threshold. But, their energy overheads are too huge to be employed in RF-powered devices. For example, the kernel circuits of the schemes consume $\sim$200 μJ per second [29], [30], nearly equal to the overhead of energy polling.

Solution. *Sentinel* proposes a low power dynamic threshold scheme to make the thresholds software-tunable by an indirect way. The scheme employs digital potentiometers to indirectly adjust the input (monitored) voltage instead of the voltage reference. Thus, the desired threshold can be set by adjusting the input voltage since the comparison results of the supervisor are affected by both the input energy and the reference. The potentiometers are digitally controlled by the MCU, and consume a small amount of energy [31]. We illustrate the principle of the scheme in section IV-A and further describe the technical detail in section IV-B.

3) Challenge 3: Limited Available Energy: Although the harvested energy is treated as an unlimited resource to RF-powered devices from a long-term point of view, the energy is actually very scarce in many cases. The reason is that the energy consumed is much more than the incoming energy. In order to explain this, we take the WISP as an example to experimentally measure the harvested power across distance. We use an Impinj Reader to continuously charge the WISP with 36 dBm EIRP (Effective Isotropic Radiated Power) and measure the harvested power at some distances, as shown in Fig. 4.

The limited energy introduces a challenge to *Sentinel*, where the proposed design of the supervisor has to be made ultra-low-power. We highlight the concern of the energy overhead, since the supervisor has to be powered all the time to detect the energy. If the design cannot be lightweight, the energy consumption must be higher than the polling overhead.

Solution. We observe that the supervisor can be designed ultra-low-power if leakage current consumption in the circuit can be limited. Thus, we attempt to reduce the leakage consumption by adding an advisable resistor under the condition of minimum disturbance to the supervisor. We illustrate the solution in detail in section IV-C.

IV. Methodology of Sentinel

In this section, we present the proposed design of *Sentinel*. First, we present the principle of *Sentinel*. Then we discuss the key techniques to realize *Sentinel*.

A. Principle of Sentinel

In order to realize low-power *Energy Sensing*, *Sentinel* exploits the trigger strategy to make the device energy-aware. Generally speaking, *Sentinel* delegates a low power software-tunable supervisor to sense energy instead of energy polling. The proposed design of the supervisor is illustrated in Fig. 5. In order to understand this, we interpret the principle of the supervisor as follows.

The supervisor includes two main schemes. First, the *low power energy detection and trigger scheme* realizes the energy

2327-4662 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2854374, IEEE Internet of Things Journal

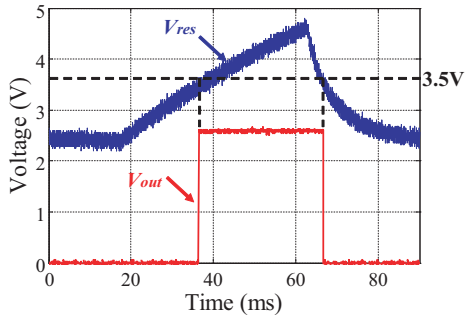


Fig. 6. The example of a comparator with a 3.5 V voltage reference, where V_{res} represents the harvested energy (voltage) from a reservoir (capacitor) and V_{out} is the output signal of the comparator.

sensing function in standby mode, as well as the function of activating the MCU when the energy reaches the threshold. Second, the *low power dynamic threshold scheme* allows the MCU to dynamically set an expected threshold to the supervisor.

Low power energy detection and trigger scheme. In order to detect energy, the supervisor includes a comparator to compare the harvested energy and the energy reference V_{ref}, as shown in Fig. 5. The output of the comparator is a high/low voltage to represent the result of the comparison. For instance, as shown in Fig. 6, the comparator outputs a high voltage (2.5 V) once the energy is above the 3.5 V threshold; otherwise, the output is a low voltage (0 V). The rising or falling edges of the output can be treated as a trigger signal to wake up and can be detected by the I/O pins of the COTS MCUs. Thus, the MCU can remain in LPM to save energy before the harvested energy reaches the threshold, and does not need to participate in the energy sensing. The MCU in LPM consumes negligible energy (only 0.25 μJ per second in the WISP [32]), as it switches off the CPU (software) and the system clock, and only keeps the kernel circuit on for being awakened by external trigger signals.

In addition, the energy consumption of the scheme has to be designed with as low power as possible, since the supervisor is powered to detect energy all the time during standby mode. In essence, the energy consumption in standby mode mostly stems from the leakage current consumption (from the reservoir to the ground in Fig. 5). The leakage consumption normally takes μA to mA current, which is close to the consumption of a computing task. To reduce the leakage consumption, an intuitive solution is to make use of a resistor with as high resistance as possible. However, the augment of the resistance implies the increasing of threshold deviation, since additional leakage current occurs at the input of the comparator. To address the challenge, we build a model to make a tradeoff between the leakage and the deviation in section IV-C. By this means, we achieve a reduction in the energy overhead without causing an unwanted threshold deviation to the supervisor. We show the leakage consumption is up to 6.7 μJ per second in the evaluation.

Low power dynamic threshold scheme. The dynamic threshold scheme can set desired thresholds in the supervisor according to the demand of a task. To realize this function, the supervisor includes a variable resistor to divide the input voltage of the comparator, as shown in Fig. 5. Specifically, the variable resistor adjusts the ratio of R_{aw} and R_{wb}. Different ratios divide the V_{res} to the corresponding different V_{in}. We can set a desired threshold by carefully designing the ratio to achieve $V_{in} = V_{ref}$ when V_{res} reaches the threshold. By this means, the MCU can set a threshold by only adjusting the variable resistor. This approach is more efficient than directly adjusting the output of the reference V_{ref} as COTS adjustable voltage reference ICs are power-starving to RF-powered devices [29], [30], consuming ~59% of the harvested energy per second in the WISP over 2 m. Furthermore, the resistor can be implemented by a digital potentiometer that the resistance is digitally controlled by the MCU with low energy consumption (the potentiometer in our design only consumes 0.25 μJ per second [31]).

As a consequence, we illustrate the principle of *Sentinel* in this section. However, there are still two questions remaining. First, how to technically achieve the low power dynamic threshold scheme? Second, how to technically reduce the leakage consumption of the energy detection in standby mode? We answer and interpret these two questions in the next two sections, respectively.

B. Designing the Low Power Dynamic Threshold Scheme

The basic approach of the scheme attempts to tune the input voltage of the comparator (V_{in}) by leveraging variable resistors to divide the voltage, as shown in Fig. 5. The threshold can be adjusted according to:

$$V_{th} = \left(\frac{R_{aw}}{R_{wb}} + 1\right) \times V_{ref} \tag{1}$$

where R_{aw} is the resistance of "*aw*" in Fig. 5 and R_{wb} is the corresponding "*wb*". We can employ a digital potentiometer to tune the ratio of R_{aw} and R_{wb}. For example, if $V_{ref} = 1.2\ V$, we can set thresholds $V_{th1} = 2\ V$ and $V_{th2} = 3.5\ V$ through $\frac{R_{aw}}{R_{wb}} = 0.667$ and 1.917, respectively.

In general, a digital potentiometer splits a large resistor into a number of segments. For example, a 1 $M\Omega$ resistor can be divided into 256 segments, with the value of each segment being 1 $M\Omega/256 \approx 3906\ \Omega$. For each segment, a switch is used to control its connectivity so that the value of the resistor can be adjusted by the switches. In order to control a number of switches easily, a decoder is employed to receive serial data from MCUs, then the output parallel signal is used to control the switches. We show the principle diagram of the digital potentiometer used in our supervisor in Fig. 7.

Encoding an expected threshold. The MCU sends digital codes to the potentiometer for threshold setting. Here, we interpret the steps to decide the digital code. First, the expected resistance (*i.e.* R_{aw} or R_{wb}) can be calculated according to Eq. 1. Then, the digital code can be decided to achieve the resistance according to Eq. 2 (here, we simply show the basic relationship. For engineers and programmers, please see the datasheet of the potentiometer you use).

$$R_{digt}(code) = \frac{code}{SEG} \times R_{ab} \tag{2}$$

2327-4662 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2854374, IEEE Internet of Things Journal

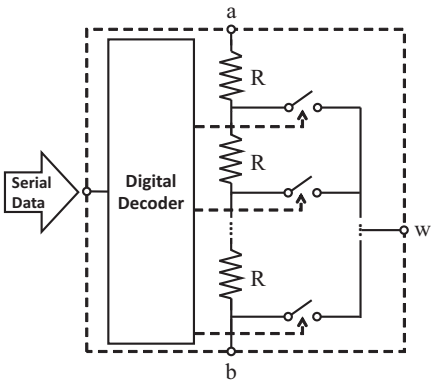


Fig. 7. Schematic of the digital potentiometer.

where $code$ is an unsigned integer in the range from 1 to SEG, SEG is the number of segments in the potentiometer and R_{ab} is the value of the whole large resistor in the potentiometer. In addition, R_{digt} can be R_{aw} or R_{wb} according to the design of the potentiometer. Thus, when an MCU is configuring a new threshold, it can directly send a corresponding $code$ to the digital potentiometer. For example, if V_{ref}=1.18 V, R_{ab}=100 KΩ and SEG=1024, the $code$ should be 550 to set the threshold to 2.56 V.

The scheme gains the merit of low energy consumption as the threshold is adjustable by only controlling the potentiometer. Many COTS potentiometers are already made ultra-low-power, and consume only *nano*Joules per second [31], [33]. Although we cannot gain all the desired values of the resistor with only one potentiometer (limited segments), it is accessible to combine multiple potentiometers to achieve an acceptable accuracy. We discuss this later in section V.

The scheme covers all valuable thresholds needed by the tasks. It seems an intuitive limitation that no threshold can be lower than V_{ref} according to Eq. 1. In practice, however, the expected thresholds will not be too low. This is because reservoirs with low voltage store very little energy, which is barely used by a task in RF-powered devices. Using the WISP as an example, if the voltage is 0.5 V, the energy in the capacitor is 1.25 μJ, which can only support the MCU running for 1.6 ms ideally. It is difficult to find a task that can be completed within such a short time. In addition, the current MCUs all have a minimum energy requirement, like 1.6 V in the MCU MSP430F2132 [32] employed in the WISP. The MCU even cannot boot up if the threshold is below 1.6 V.

C. Reducing the Leakage Consumption of Energy Detection

It is essential to reduce the leakage consumption since the supervisor stays in standby mode to detect energy all the time. The consumption mainly stems from the energy leakage when the huge current goes through the resistor to the ground (Fig. 5). In this section, we illustrate how to select an appropriate resistor to limit the energy leakage.

From Eq. 1, we can realize different thresholds by setting the ratio of R_{aw} and R_{wb}. By keeping this ratio constant, many resistor values can be selected. An implication of the selection is that a low value of R_{ab} results in a significant amount of energy leakage, which is unacceptable for RF-powered devices.

Intuitively, it seems that a very large R_{ab} can efficiently reduce the leakage. In practice, however, an excessive resistor value incurs a deviation from the expected threshold. As a result, erroneous thresholds may affect task execution and even lead to system failure.

The threshold deviation happens as a comparator also takes some current at the input. Let I_{in} is the input current of the comparator (at V_{in} in Fig. 5). The real threshold can be calculated as:

$$V_{th\_real} = V_{ref}\big(\frac{R_{aw}}{R_{wb}} + 1\big) \pm R_{aw} \cdot I_{in} \tag{3}$$

the sign "$\pm$" denotes that the current I_{in} has two directions, $i.e.$ inflow into the comparator or outflow from it. The $R_{aw}{\cdot}I_{in}$ defines the threshold deviation. We can find that for the same threshold, larger R_{aw} incurs more deviation on the real threshold V_{th_real}, while smaller R_{aw} increases the energy leakage. Thus, a proper resistor value has to be found to achieve both the low energy consumption and the minimum deviation.

To this end, we build a mathematical model to depict the tradeoff among the resistor, the deviation and the current leakage. The modelling process is illustrated as follows.

Modelling: Our model is based on two assumptions. First, the resistor value is ideally precise. Second, no current is presented at the input of V_{ref}.

We build the model by three steps. At a high level, we first formulate the deviation, and then depict the leakage current. Third, we make a tradeoff between the two factors.

Deviation. According to the definition in Eq. 3, the deviation can be derived by Eq. 4.

$$dev = \frac{|V_{th\_real} - V_{th}| + |V_{th} - V_{th\_real}|}{2V_{th}} \times 100\% \tag{4}$$

This formula comprises two directions of comparator's leakage current. For the convenience of next calculations, we assume the current direction is the inflow. Thus, we obtain the relationship between the deviation (dev) and the R_{wb} by combing Eq. 4 and Eq. 3.

$$dev(R_{wb}) = \frac{(\frac{V_{th}}{V_{ref}} - 1)R_{wb}I_{in}}{V_{th}} \times 100\% \tag{5}$$

Leakage Current. The leakage current consists of two parts, $i.e.$ the current through R_{ab} and the current through the comparator. Thus, the relationship between the leakage current (I_{leak}) and the R_{wb} is described in Eq. 6.

$$I_{leak}(R_{wb}) = \frac{V_{res} + I_{in}R_{wb}}{R_{aw} + R_{wb}} = \frac{I_{in}V_{ref} + \frac{V_{res}V_{ref}}{R_{wb}}}{V_{th}} \tag{6}$$

Tradeoff. We aim to determine R_{wb} by finding a $min\{I_{leak}(R_{wb})\}$, such that the energy leakage of our supervisor is reduced. In addition, we also take the deviation $dev(R_{wb})$ into account to guarantee the accuracy of thresholds. Thus, to make tradeoff between the two factors ($i.e.$ the deviation and the leakage), we first select an acceptable deviation depending on applications (e.g. 1% deviation) to gain

2327-4662 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2854374, IEEE Internet of Things Journal

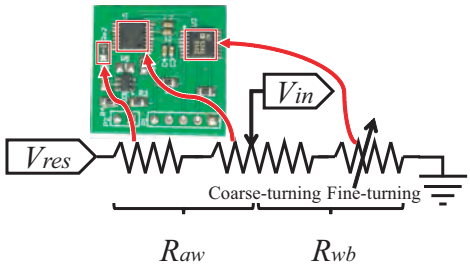


Fig. 8. Implementation of the variable resistor. We use three resistors to act as an ideal variable resistor in our design. The left resistor is used to limit leakage current. The right two variable resistors are exploited to coarse-tuning and fine-tuning, respectively.

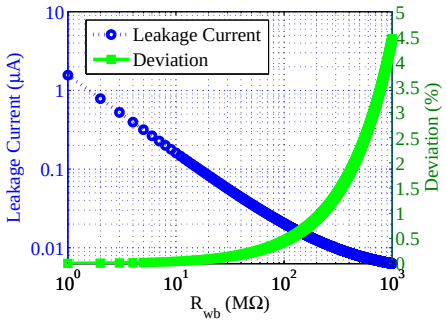


Fig. 9. Simulation of threshold deviation and leakage current for the supervisor. If we expect that the deviation is less than 1%, R_{wb} can be 224 MΩ at maximum and make 0.01μA leakage current. In many applications, we do not have to select such a large R_{wb} to limit leakage current, so that lower R_{wb} can be used to reduce deviation of thresholds.

a $max\{R_{wb}\}$, in which R_{wb} leads to $min\{I_{leak}(R_{wb})\}$ in terms of the energy leakage. Then, R_{wb} can be used to set an expected threshold according to Eq. 3. We simulate the leakage current and the deviation by this model in section V and also evaluate the deviation in section VI.

V. Implementation

The proposed design of *Sentinel* contains three components, including the comparator, the voltage reference and the variable resistor. Among them, the comparator and the reference can be easily implemented by COTS components, while the variable resistor is hardly implemented by a digital potentiometer due to the low accuracy. Existing potentiometers cannot meet the requirement of high accuracy for the variable resistor that can be set to any expected values. In this section, we first demonstrate the solution of coarse-tuning and fine-tuning by combining two potentiometers. Then, we show the components we select and the threshold deviation simulation based on the selection. Finally, we introduce the experimental platform, $i.e.$ Intel WISP.

Implementation of the variable resistor. We use two digital potentiometers as coarse-tuning and fine-tuning respectively to implement the resistor. The coarse-tuning refers to the use of a large resistance but low accuracy digital potentiometer to roughly select a close value of the expected resistance, while the fine-tuning employs a small resistance but high accuracy potentiometer to fill the gap between the coarse-tuning value and the expected value. Although two digital potentiometers bring more energy consumption to the supervisor, we show that the overhead is acceptable in section VI, as energy consumption of each potentiometer is ultra low (0.25 μJ per second).

There is one thing remaining. If we set the resistances of both two potentiometers to zero accidentally, the leakage current will be huge and even incur a short circuit. To prevent energy wastage, we append a large resistor to limit the leakage current. Altogether, we use three resistors (two digital potentiometers and one resistor) to implement the variable resistor in the supervisor. The block diagram of the circuit is illustrated in Fig. 8. For other parts of the supervisor (the comparator and the reference voltage), there are COTS components available to be directly employed.

Components Selection. According to Eq. 5, the input current I_{in} should be as small as possible to reduce the threshold deviation. To test this, we select LTC1540 [34], a nano-power comparator that has only typical I_{in}=0.01 nA. Also, LTC1540 includes an inside reference voltage V_{ref}=1.18 V. Further, we employ AD5241 [33] (1 MΩ) and AD5165 [31] (100 KΩ) as the coarse-tuning and the fine-tuning digital potentiometers, respectively. Both the potentiometers have 256 segments in consideration of the energy consumption and accuracy, and have a digital interface to be directly controlled by the MCU through a I^2C bus protocol. Furthermore, we use a precise 3 MΩ resistor to limit the leakage current consumption.

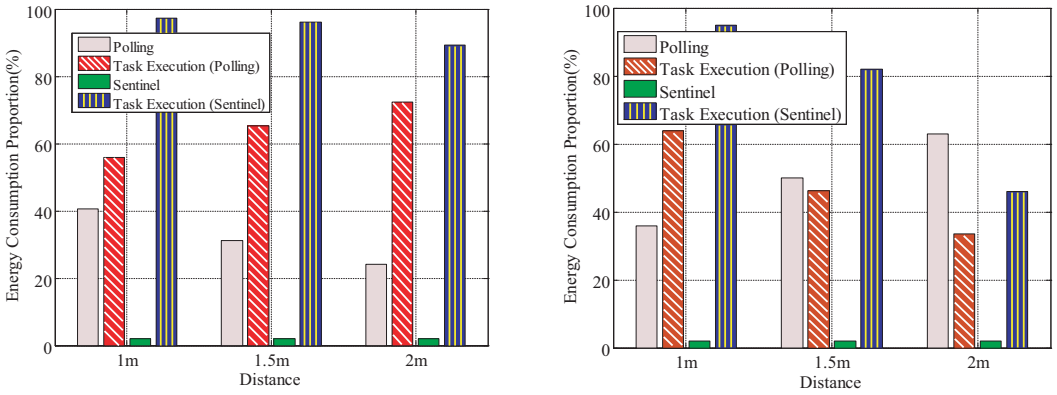
Threshold Deviation Simulation. According to the parameters of the hardware selection above, we simulate the deviation and the leakage current of the supervisor with the model described in section IV-C. The simulation result is shown in Fig. 9. Owing to the very small I_{in} of the comparator, our supervisor can achieve very high accuracy on thresholds and low power consumption simultaneously.

Experiment Platform. We integrate our supervisor on the WISP platform [3]. The WISP is a passive battery-free wireless sensing platform that harvests RF power on 902 MHz $\sim$ 928 MHz band and operates with a commercial RFID reader. We slightly modify the WISP to install the supervisor, in which the original regulator is replaced by a 2.5 V output one to supply the supervisor, as shown in Fig. 11. We also modify the layout and ground plane to improve the performance.

Energy Sensing in the WISP. The WISP mainly exploits the polling method for *Energy Sensing*, while also employing a threshold-fixed energy supervisor. On the one hand, the polling is realized by an inside ADC module in the onboard MCU. The MCU controls the ADC to periodically sense voltage in the storage capacitor and decides task executions accordingly. The polling method is fully programmable but the overhead is heavy (typically 225 μJ per second) for the WISP. On the other hand, the supervisor in the WISP has a prefixed 2 V threshold that will inform the MCU once the harvested voltage is above or below 2 V. The supervisor is ultra-low-power and only consumes 0.63 μJ per second typically. However, the fixed threshold implies that the supervisor can only be used as a low energy warning to protect the WISP against power outage, and cannot be employed as a general method for sensing energy.

2327-4662 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2854374, IEEE Internet of Things Journal



(a) *Sensing* task (b) *Computation* task

Fig. 10. Evaluation results of two popular tasks. **The proportions of task execution are treated as energy utilization efficiency due to the definition in Eq. 7.** (a) The average efficiency of the task execution (polling) is 64.7%, while the average efficiency of the task execution (*Sentinel*) is 94.9%, around a 30% improvement. (b) The average efficiency (polling) is 47.9%, while the average efficiency (*Sentinel*) is 74.3%, about a 26% improvement. In addition, the results are calculated by the data in Tab. I.

TABLE I
EXPERIMENTAL DATA FOR SENSING TASK AND COMPUTATION TASK.

	Distance		1m	1.5m	2m
	Harvested energy per second (μJ)		519.8	449.4	338.6
Sensing task (Polling+BurstExecution)	**Energy Polling**	$N_{polling}$	937	623	363
	Task execution	N_{task}	71	72	60
Sensing task (*Sentinel*+BurstExecution)	**Sentinel**	**Energy supply (μJ)**	10.4	8.2	6.8
	Task execution	N_{task}	124	106	74
Computation task (Polling+Checkpoint)	**Energy Polling**	$N_{polling}$	830	1002	947
	Task execution	t_{exe} **(ms)**	44.86	71.51	130.79
Computation task (*Sentinel*+Checkpoint)	**Sentinel**	**Energy supply (μJ)**	10.5	8.5	6.9
	Task execution	t_{exe} **(ms)**	30.13	40.37	95.53

VI. EVALUATION

In this section, we mainly evaluate two aspects. First, we want to know how much energy utilization efficiency can be improved by *Sentinel* in the WISP. Second, we also evaluate our prototype of the proposed design, including the energy consumption and the threshold deviation.

A. Evaluation of Energy Utilization Efficiency

To clearly describe the evaluation, we first introduce the basic strategy and then show the experimental setup and experiments.

1) Basic strategy: The basic method for evaluating the efficiency can be divided into two steps. First, we employ *Sentinel* to support task executions, so that the energy consumption of the task can be measured to calculate the efficiency (the ratio of the energy consumption and the harvested energy). Second, we perform the polling, as well as supporting the same task executions and calculating the efficiency for comparison.

However, it is difficult to accurately and directly measure the energy consumption of a task. To handle this, we have to measure some easily obtained parameters to indirectly calculate the energy consumption, $e.g.$ the number of executed tasks per second (N_{task}). By this means, for *BurstExecution*, the energy consumption of a task can be calculated by N_{task} multiplied by the average energy consumption of each execution (E_{ave}). In addition, for *Checkpoint*, we can measure the computing time (t_{comp}) without reboots, so that the energy consumption of a task is calculated by the power consumption (P_{comp}) multiplied by t_{comp}. Among them, E_{ave}, P_{comp} and t_{comp} can be pre-determined by relevant datasheets or off-line experiments.

Once the energy consumption of the task is obtained, the energy utilization efficiency (η_{EUE}) can be formulated as follows.

$$\eta_{EUE} = \underbrace{\frac{N_{task} \times E_{ave}}{P_{harvested} \times 1s}}_{BurstExecution} = \underbrace{\frac{P_{comp} \times t_{comp}}{P_{harvested} \times t_{exe}}}_{Checkpoint} \quad (7)$$

where t_{exe} is the execution time for a computation task with *Checkpoint* and $P_{harvested}$ is the harvested power in the WISP (we also show the harvested power over distances in Fig. 4). To calculate η_{EUE}, we only need to measure N_{task} and t_{exe} for each task, as well as $P_{harvested}$, by conducting experiments.

2) Experimental setup: We use an Impinj reader that supports EPC protocol on the UHF RFID band to communicate with the WISP. The reader has a 6 dBi gain antenna and can transmit 30 dBm power. The WISP harvests RF power from the reader, and executes tasks according to the results of *Energy Sensing* by the polling or *Sentinel*. Figure 11 shows the experiment environment for the evaluation. We keep this setup the same while only changing the distance between the reader and the WISP (1.0, 1.5 and 2.0 m).

2327-4662 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2854374, IEEE Internet of Things Journal

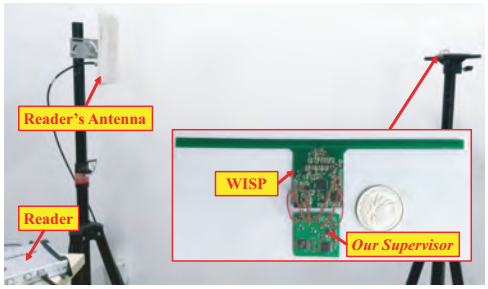


Fig. 11. Experimental environment.

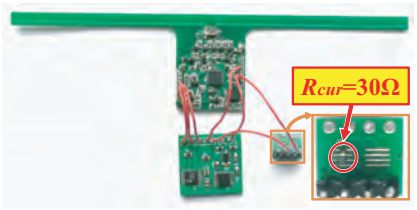


Fig. 12. The approach to measure current consumption of the supervisor by installing the resistor R_{cur} at the port of supply voltage.

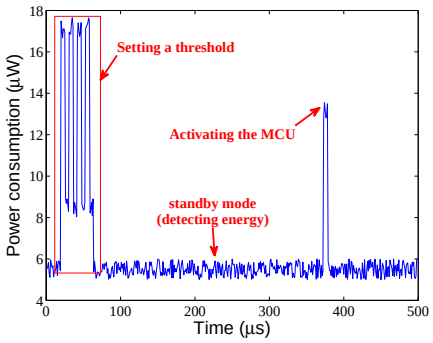


Fig. 13. Evaluation results of the power consumption of the supervisor. In order to set a threshold, the MCU has to operate the coarse-tuning and fine-tuning potentiometers, respectively. The time to set a threshold is ∼50 μs.

We consider a popular task for each execution planning, respectively, in the evaluation. First, the *Sensing* task operates a sensor with the *BurstExecution* planning. Second, the *Computation* task runs a CRC test with the *Checkpoint* planning. We then record N_{task} for the *Sensing* task and t_{exe} for the *Computation* task to calculate the efficiency. The description of each task is illustrated in the following content.

3) Sensing task: The *Sensing* task refers to the on-board MCU operating a humidity and temperature sensor (Si7013 [28]) continuously. Every operation consumes E_{ave}=4.08 μJ on average to sense the humidity and temperature data, which is obtained from the datasheet.

The *Sensing* task is executed by the *BurstExecution* planning. We use one pin of the MCU to represent states of the task execution ($i.e.$ executing or finished) by a high/low output voltage, so that we can count N_{task} using a digital oscilloscope.

We consider both *Sentinel* and energy polling for *Energy Sensing*. We also count the number of energy samplings per second ($N_{polling}$) to calculate the energy overhead of the polling. The experimental results are listed in Tab. I. In addition, the energy consumption of *Sentinel* is evaluated in section VI-B. Consequently, we calculate η_{EUE} for the *Sensing* task by Eq. 7, as shown in Fig. 10(a). *Note that the proportion of task execution in Fig. 10 is treated as the energy utilization efficiency, since both they are calculated by Eq. 7.*

Note that the energy consumption of "Task Execution (Polling)" in Fig. 10(a) increases with distance. This happens because we directly implement *Dewdrop* [19] in the *Sensing* task (*Polling* + *BurstExecution*). The polling in *Dewdrop* will adaptively double or halve the polling frequency according to the speed of energy harvesting. The harvesting is slowed down once the WISP is away from the reader. Therefore, the halved polling frequency reduces the energy overhead and consequently augments the energy efficiency. The average efficiency of *Dewdrop* is 64.7% over the distance, which is currently the highest efficiency in RF-powered devices to the best of our knowledge.

4) Computation task: The *Computation* task evaluates the CRC test computing a CRC16-CCITT checksum over a 2 KB region of on-chip flash memory in the WISP. The power consumption of the task is P_{comp}=750 μW, and the computing time without reboots is t_{comp}=19.84 ms.

In this evaluation, we implement *Mementos* [24] and measure t_{exe} for the CRC test. *Mementos* uses the polling for *Energy Sensing* and *Checkpoint* for the task execution. Instead of the polling, we also employ *Sentinel* by keeping the other portions the same to record t_{exe} of the task execution. t_{exe} can be recorded by a pin of the MCU to represent states of the WISP (a low voltage indicates in low-power mode to accumulate energy, while a high voltage implies in active-mode for task execution). We show the results in Tab. I and the calculated efficiency in Fig. 10(b).

According to Fig. 10(b), we observe that η_{EUE} (both the polling and *Sentinel*) decrease with distance. This is because the reduction of the harvested energy (along with the increase in distance) incurs the augment of reboots ($i.e.$ protecting tasks when the harvested energy is running up, and recovering it once the device is powered up again). The reboots are not treated as tasks because they only support task executions as a runtime. Thus, the energy consumption of the task is reduced over distances.

B. Evaluation of the Prototype of Sentinel

In this section, we conduct experiments to evaluate the prototype supervisor, including the energy consumption and the threshold deviation.

1) Energy consumption: The main energy consumption (supply consumption) consists of three portions, $i.e.$, detecting the energy in standby mode, setting thresholds and activating the MCU. In addition, the leakage consumption is also taken into account in the evaluation.

Supply consumption: This consumption can be evaluated by measuring the current consumption. Further, the power consumption can be calculated by the current consumption multiplied by the supply voltage. To measure the current, we

2327-4662 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2854374, IEEE Internet of Things Journal

TABLE II
ENERGY CONSUMPTION OF SENTINEL VS. POLLING IN THE WISP

Energy Sensing	Consumption	Value
Sentinel	Standby	$\leq$11.7μJ per second
	Threshold setting	39.7nJ per setting
	Activating the MCU	0.68nJ per activating
Energy Polling	Energy Supply	225μJ per second

append a small resistor R_{cur} at the port of supply voltage in series, as shown in Fig. 12. Then, the voltage created on the resistor can be recorded to calculate the current values by Ohm's law. We choose the value of R_{cur}=30 Ω to minimize the disturbance from modifying the circuit. The results are demonstrated in Fig. 13. The supervisor takes $\sim$5 μW to detect energy in standby mode, and consumes 2.24 and 0.68 nJ to set a threshold and trigger the device, respectively. Note that the energy consumption of the MCU should be taken in account as well, since the MCU is processing during the threshold setting. The power consumption is 750 μW and the duration is 50 μs (Fig. 13). Thus, the total energy consumption of setting a threshold is 39.7 nJ.

Leakage consumption: The leakage consumption constantly changes as the harvested voltage varies. We consider the worst case for the leakage so that the consumption can be easily calculated. According to the implementation, the minimum value of the resistor is 4 MΩ. The maximum harvested voltage in the WISP is 5.2 V. Thus, the maximum leakage power is 6.7 μW.

As a consequence, we combine the two portions to gain the overall energy consumption of *Sentinel*, as illustrated in Tab. II. The energy consumption of the standby mode comprises the supply consumption and the leakage consumption. Normally, the standby mode takes 11.7 μJ per second at maximum, which is 5.2% of the energy consumption of the polling (225 μJ per second) in the WISP. In addition, we also consider the energy consumption of the threshold setting and the MCU activation. This part of the consumption depends on the amount of executed tasks per second. Normally, the number is $\sim$100 for a lightweight task, for example, N_{task}=124, 106 and 74 over distances in Tab. I. If the task has only one threshold demand, the energy consumption is 4 μJ per second.

2) Threshold deviation: In practice, the deviation incurs an output delay of the trigger signal. For example, assuming that the expected threshold is 2 V, while the actual threshold is 2.2 V, the supervisor activates the MCU when the voltage is above 2.2 V, not 2 V. Thus, we measure the output delay to evaluate the deviation. We use an oscilloscope to record the voltage and monitor the trigger signal so that the delay can be measured. The results are shown in Fig. 14.

We observe that the maximum output delay is 60 μs, which is much shorter than the interval of the polling (typically 1 ms). We believe that the delay is acceptable in RFID-based platforms [3], [35]–[37]. This is because the harvested energy changes in milliseconds in RFID-based platforms. For other platforms [22], [38] or power-heavy applications [39]–[41], they often use larger capacitors ($e.g.$ 1 F) to store more harvested energy. This means that the trigger delay must also be acceptable because the higher capacitance indicates the slower charging/discharging speed.

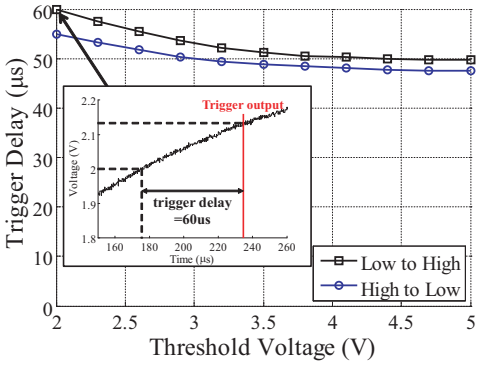


Fig. 14. We evaluate the threshold deviation by measuring the trigger delay. The two cases ("Low to High" and "High to Low") refer to the rising edge and the falling edge of the output voltage, respectively. The sub-figure shows the method to measure the trigger delay as an example at 2 V threshold voltage in "Low to High" case. We use the same method to measure the delay for all the thresholds and the cases.

VII. RELATED WORK

We have discussed the existing execution plans in section II, so we focus on the related works of *Energy Sensing* in this section. The existing works can be divided into two parts, $i.e.$ energy polling and energy trigger.

Energy Polling. Polling is currently the common method to sense the harvested energy in RF-powered devices. Many works exploit polling to be aware of the energy, like [19]–[22], [24]–[26], [42]. Furthermore, the polling is also widely used in battery-assisted devices ($e.g.$ mobile phones and wristbands). For a battery-assisted device, the energy overhead of the polling may be negligible as batteries possess far more energy (4-6 orders of magnitude) than the energy overhead. However, in RF-powered devices, the energy overhead becomes significant and reduces the energy exploited for task executions.

Energy Trigger. Some works, like [18], [43]–[45], employ a hardware supervisor to sense energy statically and passively. Once the harvested energy reaches a threshold, the supervisor will trigger the device to execute a task. This method is lightweight as the device does not need to periodicity sample the resident energy and stays in low-power mode to save energy. However, the thresholds in existing supervisors are pre-fixed by hardware. This method can only provision for the worst case scenario (mainly protection against power outage) and cannot become a common approach for *Energy Sensing* in RF-powered devices.

To reduce the overhead of energy sensing, we propose *Sentinel* and present the proposed design to realize the software-tunable supervisor. The supervisor enables *Sentinel* to detect any expected thresholds, so that *Sentinel* becomes a new common method for *Energy Sensing*. Compared to the polling, *Sentinel* is low energy consumption and suitable for RF-powered devices. With *Sentinel*, more energy is saved to execute tasks so that the energy utilization efficiency is improved.

2327-4662 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2854374, IEEE Internet of Things Journal

VIII. Conclusion

To improve the energy utilization efficiency, RF-powered devices plan task executions according to the harvested energy. To sense the energy, the only current general approach is energy polling. However, the energy overhead of the polling is huge to RF-powered devices and thus becomes the bottleneck of energy utilization efficiency.

In this study, we propose *Sentinel*, a novel general approach to sense energy with low energy consumption. *Sentinel* employs a trigger strategy to avoid the polling. Consequently, more saved energy can be utilized to execute tasks so that the utilization efficiency can be improved up to 94.9% in the WISP (combined with existing strategies of task executions).

Sentinel needs a software-tunable supervisor which the threshold can be directly tuned by software. We present the proposed design by proposing the low power energy detection and trigger scheme, as well as the low power dynamic threshold scheme. Finally, we implement and evaluate the supervisor by COTS components. The results show the supervisor only consumes up to 11.7 μJ per second to detect energy in standby mode and fits for RF-powered devices.

Acknowledgement

The authors would like to thank the editors and anonymous reviewers for their insightful comments and constructive feedback. This work is supported in part by the National Key R&D Program of China (2017YFB1003003), and in part by National Natural Science Foundation of China (61472068).

References

[1] K. Mok, "Delivering power with wi-fi signals to the next billion devices, no batteries required," https://thenewstack.io/delivering-power-with-wi-fi-signals-to-the-next-billion-devices-no-batteries-required/, 2015, accessed: 2017-10-29.

[2] A. Nimo, T. Beckedahl, T. Ostertag, and L. Reindl, "Analysis of passive rf-dc power rectification and harvesting wireless rf energy for micro-watt sensors," vol. 3, pp. 184–200, 04 2015.

[3] A. P. Sample, D. J. Yeager, P. S. Powledge, A. V. Mamishev, and J. R. Smith, "Design of an rfid-based battery-free programmable sensing platform," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 11, pp. 2608–2615, 2008.

[4] B. Kellogg, A. Parks, S. Gollakota, J. R. Smith, and D. Wetherall, "Wi-fi backscatter: Internet connectivity for rf-powered devices," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 607–618.

[5] D. Bharadia, K. R. Joshi, M. Kotaru, and S. Katti, "Backfi: High throughput wifi backscatter," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 283–296, 2015.

[6] J. F. Ensworth and M. S. Reynolds, "Every smart phone is a backscatter reader: Modulated backscatter compatibility with bluetooth 4.0 low energy (ble) devices," in *RFID (RFID), 2015 IEEE International Conference on*. IEEE, 2015, pp. 78–85.

[7] B. Kellogg, V. Talla, S. Gollakota, and J. R. Smith, "Passive wi-fi: Bringing low power to wi-fi transmissions." in *NSDI*, vol. 16, 2016, pp. 151–164.

[8] P. Zhang, D. Bharadia, K. R. Joshi, and S. Katti, "Hitchhike: Practical backscatter using commodity wifi." in *SenSys*, 2016, pp. 259–271.

[9] P. Zhang, M. Rostami, P. Hu, and D. Ganesan, "Enabling practical backscatter communication for on-body sensors," in *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*. ACM, 2016, pp. 370–383.

[10] V. Iyer, V. Talla, B. Kellogg, S. Gollakota, and J. Smith, "Inter-technology backscatter: Towards internet connectivity for implanted devices," in *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*. ACM, 2016, pp. 356–369.

[11] A. Yakovlev, S. Kim, and A. Poon, "Implantable biomedical devices: Wireless powering and communication," *IEEE Communications Magazine*, vol. 50, no. 4, 2012.

[12] M. Buettner, B. Greenstein, A. Sample, J. R. Smith, D. Wetherall *et al.*, "Revisiting smart dust with rfid sensor networks," in *Proceedings of the 7th ACM Workshop on Hot Topics in Networks (HotNets-VII)*, 2008.

[13] F. Gasco, P. Feraboli, J. Braun, J. Smith, P. Stickler, and L. DeOto, "Wireless strain measurement for structural testing and health monitoring of carbon fiber composites," *Composites Part A: Applied Science and Manufacturing*, vol. 42, no. 9, pp. 1263–1274, 2011.

[14] C. A. Tokognon, B. Gao, G. Y. Tian, and Y. Yan, "Structural health monitoring framework based on internet of things: A survey," *IEEE Internet of Things Journal*, vol. 4, no. 3, pp. 619–635, June 2017.

[15] W. Anran, V. Iyer, V. Talla, J. Smith, and S. Gollakota, "Fm backscatter: Enabling connected cities and smart fabrics," in *Usenix NSDI*, 2017.

[16] S. Amendola, R. Lodato, S. Manzari, C. Occhiuzzi, and G. Marrocco, "Rfid technology for iot-based personal healthcare in smart spaces," *IEEE Internet of Things Journal*, vol. 1, no. 2, pp. 144–152, April 2014.

[17] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, Feb 2014.

[18] P. Zhang and D. Ganesan, "Enabling bit-by-bit backscatter communication in severe energy harvesting environments," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, Conference Proceedings, pp. 345–357.

[19] M. Buettner, B. Greenstein, and D. Wetherall, "Dewdrop: an energy-aware runtime for computational rfid," in *Proc. USENIX NSDI*, 2011, pp. 197–210.

[20] T. Zhu, A. Mohaisen, Y. Ping, and D. Towsley, "Deos: Dynamic energy-oriented scheduling for sustainable wireless sensor networks," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 2363–2371.

[21] J. Hester, T. Scott, and J. Sorber, "Ekho: Realistic and repeatable experimentation for tiny energy-harvesting sensors," in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. ACM, 2014, pp. 330–331.

[22] J. Hester, L. Sitanayah, and J. Sorber, "Tragedy of the coulombs: Federating energy storage for tiny, intermittently-powered sensors," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2015, pp. 5–16.

[23] B. Lucia and B. Ransford, "A simpler, safer programming and execution model for intermittent systems," *ACM SIGPLAN Notices*, vol. 50, no. 6, pp. 575–585, 2015.

[24] B. Ransford, J. Sorber, and K. Fu, "Mementos: System support for long-running computation on rfid-scale devices," *Acm Sigplan Notices*, vol. 47, no. 4, pp. 159–170, 2012.

[25] D. Balsamo, A. S. Weddell, G. V. Merrett, B. M. Al-Hashimi, D. Brunelli, and L. Benini, "Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems," *IEEE Embedded Systems Letters*, vol. 7, no. 1, pp. 15–18, 2015.

[26] A. Mirhoseini, E. M. Songhori, and F. Koushanfar, "Idetic: A high-level synthesis approach for enabling long computations on transiently-powered asics," in *Pervasive Computing and Communications (PerCom), 2013 IEEE International Conference on*. IEEE, 2013, pp. 216–224.

[27] F. Azmat, Y. Chen, and N. Stocks, "Predictive modelling of rf energy for wireless powered communications," *IEEE Communications Letters*, vol. 20, no. 1, pp. 173–176, Jan 2016.

[28] S. Labs, "Humidity and temperature sensor," https://www.silabs.com/documents/public/data-sheets/Si7013-A20.pdf, 2016.

[29] T. Instruments, "Lm285 - adjustable micropower voltage reference," http://www.ti.com/lit/ds/symlink/lm285-adj.pdf, 2013.

[30] M. Integrated, "Max6160 - sot23, low-cost, low-dropout, 3-terminal voltage references," https://datasheets.maximintegrated.com/en/ds/MAX6125-MAX6160.pdf, 2016.

[31] A. Devices, "Ultralow power 1.8 v logic-level digital potentiometer," http://www.analog.com/media/en/technical-documentation/data-sheets/AD5165.pdf, 2017.

[32] T. Instruments, "Msp430f2132 datasheet," http://www.ti.com.cn/cn/lit/ds/symlink/msp430f2132.pdf, 2012.

[33] A. Devices, "256-position digital potentiometers," http://www.analog.com/media/en/technical-documentation/data-sheets/AD5241_5242.pdf, Rev. D.

[34] L. Technology, "Nanopower comparator with reference," http://cds.linear.com/docs/en/datasheet/1540fas.pdf, 1997.

2327-4662 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2854374, IEEE Internet of Things Journal

[35] R. Mirjalili and B. A. Parviz, "Ubitag: A ubiquitous optical wireless tag for identifying objects in the physical space," *IEEE Internet of Things Journal*, vol. 2, no. 3, pp. 259–264, June 2015.
[36] M. S. Khan, M. S. Islam, and H. Deng, "Design of a reconfigurable rfid sensing tag as a generic sensing platform toward the future internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 300–310, Aug 2014.
[37] H. Zhang, J. Gummeson, B. Ransford, and K. Fu, "Moo: A batteryless computational rfid and sensing platform," 11 2017.
[38] F. Simjee and P. H. Chou, "Everlast: long-life, supercapacitor-operated wireless sensor node," in *Low Power Electronics and Design, 2006. ISLPED'06. Proceedings of the 2006 International Symposium on*. IEEE, 2006, pp. 197–202.
[39] S. Naderiparizi, A. N. Parks, Z. Kapetanovic, B. Ransford, and J. R. Smith, "Wispcam: A battery-free rfid camera," in *RFID (RFID), 2015 IEEE International Conference on*. IEEE, 2015, pp. 166–173.
[40] J. Chen, K. Hu, Q. Wang, Y. Sun, Z. Shi, and S. He, "Narrow-band internet of things: Implementations and applications," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2017.
[41] D. Wu, L. Lu, M. J. Hussain, S. Li, M. Li, and F. Zhang, "R3: Reliable over-the-air reprogramming on computational rfids," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 17, no. 1, p. 9, 2017.
[42] B. Ransford, S. S. Clark, M. Salajegheh, and K. Fu, "Getting things done on computational rfids with energy-aware checkpointing and voltage-aware scheduling." *HotPower*, vol. 8, pp. 5–5, 2008.
[43] P. Zhang, D. Ganesan, and B. Lu, "Quarkos: Pushing the operating limits of micro-powered sensors," in *Proceedings of the 14th USENIX conference on Hot Topics in Operating Systems*. USENIX Association, 2013, pp. 7–7.
[44] H. Jayakumar, A. Raha, W. S. Lee, and V. Raghunathan, "Quickrecall: A hw/sw approach for computing across power cycles in transiently powered computers," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 12, no. 1, p. 8, 2015.
[45] D. Balsamo, A. Das, A. S. Weddell, D. Brunelli, B. M. Al-Hashimi, G. V. Merrett, and L. Benini, "Graceful performance modulation for power-neutral transient computing systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 5, pp. 738–749, 2016.

Songfan Li is currently pursuing the Ph.D. degree at the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC). His research interests include energy harvesting systems, embedded wireless systems and low power devices.

Li Lu received his B.E. and M.S degrees in 2000 and 2003, respectively, all in Automation Control. He received PhD in the Key Lab of Information Security, Chinese Academy of Science, in 2007. He is a Professor with School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include Battery-free systems and RFID technology, wireless network and network security. He is a member of IEEE Communication Society and ACM.

Muhammad Jawad Hussain received his B.E. degree in Avionics from College of Aeronautical Engineering (CAE), National University of Science and Technology (NUST), Pakistan in 2005. For next 6 years, he worked as a design and field engineer for Electronic Warfare systems. He received his M.S. and Ph.D degrees in Information Security from University of Electronic Science and Technology of China in 2013 and 2017 respectively. His current research interests include Security in Backscatter tokens and Computational RFID systems.

Yalan Ye received her Ph.D. degree in the University of Electronic Science and Technology of China(UESTC), in 2008. She is currently an associate professor with the School of Computer Science and Engineering in UESTC. Her research interests include biomedical signal processing, information processing, machine learning.

Hongzi Zhu received his PhD in the Department of Computer Science and Engineering at Shanghai Jiao Tong University in 2009. He is an Associate Professor working at Shanghai Jiao Tong University. His research interests include mobile computing, wireless networks, vehicular ad hoc Networks and network security. He is a member of the IEEE Computer and the IEEE Communication Society.

2327-4662 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.