electronics

MDPI

Article

Handling Power Depletion in Energy Harvesting IoT Devices

Young-myoung Kang [1] and Yeon-sup Lim [2,*]

[1] Department of Computer Engineering, Sungkyul University, Anyang 14097, Republic of Korea; ykang@sungkyul.ac.kr
[2] Department of Convergence Security Engineering, Sungshin Women's University, Seoul 02844, Republic of Korea
* Correspondence: ylim@sungshin.ac.kr; Tel.: +82-2-920-7144

Abstract: Efficient energy management is a significant task in Internet-of-Things (IoT) devices because typical IoT devices have the constraint of a limited power supply. In particular, energy harvesting IoT devices must be tolerant of complex and varying temporal/spatial environments for energy availability. Several schemes have been proposed to manage energy usage in IoT devices, such as duty-cycle control, transmission power control, and task scheduling. However, these approaches need to deal with the operating conditions particular to energy harvesting devices, e.g., power depletion according to energy harvesting conditions. In this paper, regarding a wireless sensor network (WSN) as a representative IoT device, we propose an Energy Intelligence Platform Module (EIPM) for energy harvesting WSNs. The EIPM provides harvested energy status prediction, checkpointing, and task execution control to ensure continuous operation according to energy harvesting conditions while minimizing required hardware/software overheads such as additional measurement components and computations. Our experiment results demonstrate that the EIPM successfully enables a device to cope with energy insufficiency under various harvesting conditions.

Keywords: energy harvesting; IoT devices; solar harvesting; duty cycling; WSN

Citation: Kang, Y.-m.; Lim, Y.-s. Handling Power Depletion in Energy Harvesting IoT Devices. *Electronics* **2024**, *13*, 2704. https://doi.org/10.3390/electronics13142704

Academic Editor: Esteban Tlelo-Cuautle

Received: 15 June 2024
Revised: 3 July 2024
Accepted: 9 July 2024
Published: 10 July 2024

1. Introduction

The advancement of the Internet-of-Things (IoT) is paving the way for revolutionizing industries as well as explosive economic growth [1,2]. For example, usage of wireless sensor networks (WSNs), a representative realization of IoT deployments, is increasingly growing, with numerous applications such as monitoring rivers, wildfires, volcanoes, and earthquakes [1,3,4].

One of the important constraints on the performance and lifetime of such devices is their dependence on batteries. The limited capacity of the batteries embedded in IoT devices incurs high maintenance costs for manually replacing batteries when depleted [5,6]. Recent studies have introduced various energy harvesting techniques to resolve this problem and enable perpetual operations without maintenance [1,5,7–9]. Energy harvesting is essentially a technique to obtain energy from ambient sources such as RF radio waves [10], light [11], and mechanical motion [12]. The best energy harvesting method differs depending on the application's operational needs and environment [13].

Among various energy harvesting technologies available today, solar energy harvesting is generally the most effective at providing high power density. It is the optimal energy source for an IoT device like a sensor node operating with relatively small harvesting modules [14,15]. However, due to power fluctuations from the solar panels and the limited storage capacity of an energy buffer, there are several design issues for solar harvesting in IoT devices:

- The amount of harvested energy can exceed the storage capacity from time to time. An energy harvesting IoT device should aggressively utilize such abundant energy.

- Energy harvesting conditions vary temporally and spatially. An energy harvesting IoT device must cope with such uncontrollable variability to extend its lifetime and increase sustainability.
- Energy harvesting without another external power source can cause frequent power depletion. Such power depletion results in shutdowns and restarts, leading to loss of data and severe operational disruptions. An energy harvesting IoT device must respond appropriately to frequent system failures caused by insufficient harvested energy.

To address these design issues, we first build an analytical model for the status of energy harvesting IoT devices that includes power depletion and task rejection probabilities according to the energy harvesting rate and event arrival rate. Then, we propose an energy intelligence platform to control the device behavior according to the energy harvesting conditions. If the device works under a time-varying energy harvesting environment, the EIPM enables it to adjust its task execution rate accordingly and keep its state even with power depletion.

The remainder of this paper is organized as follows: In Section 2, we discuss related work. Section 3 presents our target environment and analytical model. The proposed EIPM is explained in Section 4. Section 5 demonstrates the experiment results using our EIPM implementation in a solar energy harvesting device. Finally, we conclude this paper in Section 6.

2. Related Work

Enormous research efforts have focused on energy-aware management for mobile and IoT devices [16]. Singh et al. [5] provide a systematic taxonomy for energy management schemes, particularly for WSNs, and examine various protocols and algorithms along with different energy management policies. However, these conventional approaches aim to control devices based solely on the battery level and do not consider the energy harvested from the environment.

There have been several research studies for managing scarce energy resources for devices powered by harvested energy [9]. Corke et al. [17] present the hardware design principles for long-term solar-powered wireless sensor networks. Simjee and Chou [18] propose a supercapacitor-operated solar-powered wireless sensor node called Everlast. These studies address initial system architectures for energy harvesting sensor nodes. Noh et al. [19] propose an energy assignment approach for solar-powered sensor systems to satisfy an objective such as ENO. Moser et al. [20] propose a parameterized specification and a corresponding optimal policy to control sensing and communication rates for devices powered by harvested energy instead of solving the optimization problem in real-time. Vigorito et al. [21] seek to address this limitation by formulating duty cycling as an adaptive control problem. Sarang et al. [22] present an energy-neutral operation (ENO)-based adaptive duty cycle MAC protocol that supports each sensor remaining in ENO as long as possible and using the surplus harvested energy to improve network performance. In [23], Bengheni et al. propose a multi-threshold energy regulating algorithm that dynamically adjust its duty cycle through calculating its sleep interval according to the amount of currently remaining energy. Havrlík et al. [24] address how distortion of signals in an energy harvesting environment affects the complexity and performance of the overall system.

However, these approaches assume that the energy harvesting source is stable or that the devices' energy usage and workload profiles are known, although energy sources such as solar power have high variability and the profiles will likely be unavailable. Substantial research has been conducted with the objective of addressing these issues [25]. Zou et al. [26] develop an approach to enable devices to adjust their scheduling plans according to energy production and residual battery levels. Cammarano et al. [27] present an energy prediction model for energy harvesting sensor networks that leverages past energy observations to forecast future energy availability. By integrating energy predictors with time slots of varying lengths, it is possible to enhance the performance of the system by adapting to the fluctuations of the power source. Peng et al. [28] propose a framework for

managing harvested energy in a prediction-free manner. The authors validate the efficacy of their scheme through theoretical analysis and extensive simulations without real-world experiments. In [29], Ashraf et al. employ a queue control model to maintain the energy level at a reference value based on a prediction of the harvestable energy and achieve higher throughput compared with throughput optimal schemes. Li et al. [30] explore task scheduling using predictions based on weather forecasts in energy harvesting systems and demonstrate accurate predictions for medium timescales. For the purpose of minimizing unnecessary errors in solar-based harvesting systems, Sah et al. [3] present a prediction technique based on prior energy measurements to show the future energy status for the respective time slot. In [31], Sah et al. propose a routing awareness scheduling algorithm to address crucial issues such as energy depletion, coverage preservation, routing, clustering, etc. The presented approach dynamically coordinates the role of sensor nodes , i.e., a header or a member of a cluster, based on both the remaining and harvested energy and the number of active nodes.

The research studies discussed above focus on maximizing the utilization of devices that use an ambient power source rather than considering seamless operation after power depletion occurs. Even though these approaches enable devices to optimize power consumption, power depletion events are inevitable. Checkpointing [32] is a typical technique to save the memory state on non-volatile storage in operating systems; restarting from the checkpointed state enables devices to recover their memory state from the last checkpoint. If an IoT node has a checkpoint before it turns off, it can perform subsequent operations after power becomes sufficient. Several studies [33,34] introduce checkpointing techniques to prevent the loss of state information due to power depletion. However, these approaches need to cope carefully with dynamic energy harvesting environments. Considering energy harvesting environments, Ransford et al. [35] suggest checkpointing and rescheduling techniques in computational RFIDs that scavenge energy from an ambient source. However, since RFIDs have severe resource constraints, their proposed approach uses a simple voltage-based policy to determine whether to perform checkpointing and rescheduling.

3. Problem Statement

3.1. Energy Harvesting IoT Devices

We focus on energy harvesting IoT devices that use capacitors rather than conventional rechargeable batteries as an energy storage/buffer. These two energy storage methods have trade-offs: a capacitor works for many recharge cycles but discharges quickly without a power source. In contrast, a rechargeable battery works for limited recharge cycles and takes a relatively long time to charge up but discharges very slowly. Aiming to sustain the IoT environment without maintenance, we consider capacitors as energy storage for the devices. However, the energy storage capacity of capacitors varies from a few tens of μF to thousands of F. In particular, a typical capacitor has just a few thousand μF, which is too small compared with rechargeable batteries. This results in intolerance to dynamics in energy availability. Also, the intolerance will matter even with a large energy storage capacity if the amount of harvested energy is insufficient. For an energy harvesting source, we choose a solar panel: the most popular and available power source with a dynamic nature for the available amount of harvested energy. Regarding a sensor node as the simplest representative of an IoT device, we use a legacy sensor node [36] equipped with a TI solar panel [37] for the implementation and experiments.

3.2. Microbenchmarks

For the selected device, we conduct a series of microbenchmarks to measure energy consumption for the devices' operations, as presented in Table 1. The device uses a capacitor as energy storage in these experiments. Before each experiment begins, we fully charge the capacitor by connecting the device to an external power supply. After disconnecting the external power supply from the device, we make the device run each operation using

only the energy in the capacitor. Then, we compute the energy consumption based on the output voltage drop of the capacitor while completing each operation.

Table 1. Energy consumption of operations.

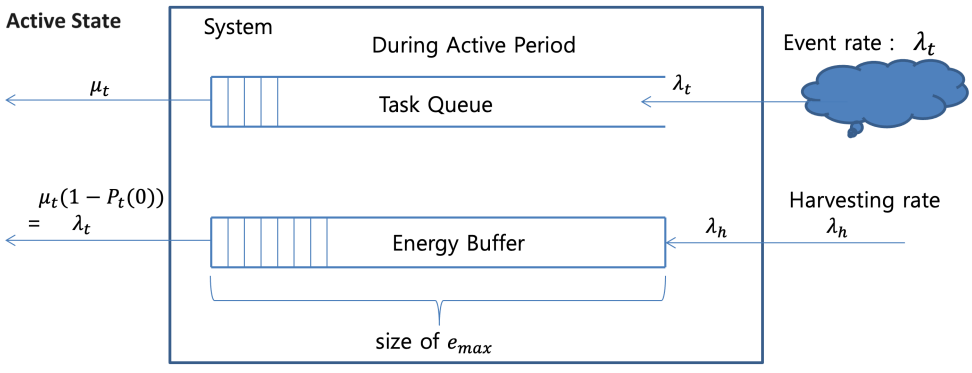
Operation	Measured Energy Consumption
Flash Write (E_{fw})	0.5793 µJ/byte
Flash Read (E_{fr})	0.1588 µJ/byte
Flash Erase (E_{fe})	15,065 µJ
ADC Read (E_{adc})	5.2280 µJ/trial
Memory Operation (E_{mem})	0.0142 µJ/byte
Do Nothing (E_{LPM3})	22.082 µJ/s

3.3. Analytical Model

We build an analytical model to estimate the status of energy harvesting devices, such as the usage of the harvested energy and power depletion, under the following assumptions:

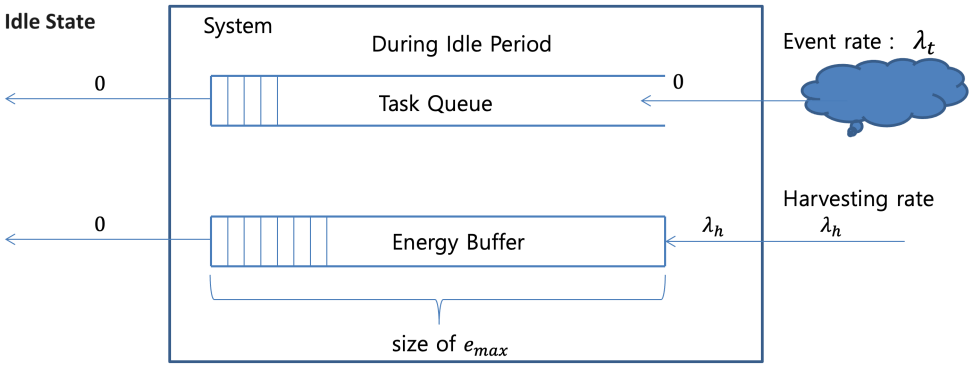
- A device performs a duty cycle with a fixed active period ratio ρ in one cycle of L unit time for processing event tasks.
- The energy buffer of the device is initially empty, and its maximum size is e_{max} units. The device starts its first operation once the energy buffer becomes full.
- A device executes tasks that have arrived with the rate of μ_t, which means the average service time of a task is $\frac{1}{\mu_t}$. The arrival of events is a Poisson process with the rate of λ_t. One task execution consumes one unit of energy from the buffer.
- The arrival of harvested energy is also a Poisson process, with the rate of λ_h.
- Task acceptance and execution are possible only during an active period and with available energy.

Energy is drained from the buffer when the device runs a task, which occurs when the task queue system is not empty during an active period in the duty cycle. Let $P_t(0) = 1 - \frac{\lambda_t}{\mu_t}$ denote the probability that the task queue system is empty. Then the energy drain rate of the buffer μ_d is computed as $\mu_d = \mu_t(1 - P_t(0))$. Figure 1 illustrates the behavior of the task queue and energy buffer under these assumptions.



(**a**)

Figure 1. *Cont.*



(**b**)

Figure 1. System diagram: (**a**) active period; (**b**) idle period.

Let $E[B_0]$ denote the expected energy buffer level at the moment of the first duty cycle of the device. Since we assume the device starts its first operation with a fully filled buffer, $E[B_0]$ is e_{max}. The expected amount of harvested energy during one cycle is $L\lambda_h$, and that of consumed energy for executing tasks during an active period is $L\lambda_t\rho$. We define the expected energy level at cycle k ($E[B_k]$) as follows:

$$\begin{aligned} E[B_0] &= e_{max} \\ E[B_k] &= E[B_0] + kL(\lambda_h - \mu_d\rho) \end{aligned}$$

When $E[B_k]$ becomes zero, the device goes into power depletion status. Therefore, the expected number of duty cycles until power depletion occurs ($E[k]$) is defined as follows:

$$E[k] = \frac{e_{max}}{L(\mu_d\rho - \lambda_h)}$$

After experiencing power depletion, the device starts new cycles once the energy level reaches e_{max} again. Let C_d (power depletion cycle) denote the total length of duty cycles until power depletion happens and the energy harvesting duration for restarting the device. Then the expected value for C_d ($E[C_d]$) is computed as follows:

$$E[C_d] = E[k]L + \frac{e_{max}}{\lambda_h} = e_{max}\left(\frac{1}{\mu_d\rho - \lambda_h} + \frac{1}{\lambda_h}\right)$$

Then, the power depletion probability is defined as $\frac{1}{E[C_d]}$.

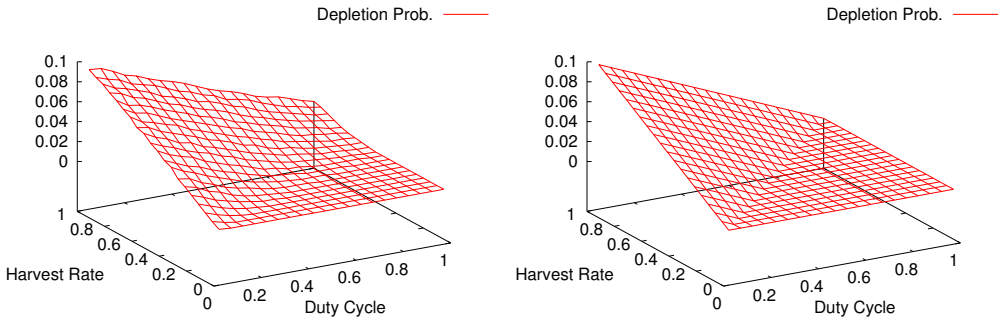
The device cannot accept tasks during an idle period or while fully charging the empty energy buffer in one power depletion cycle. Therefore, the duration for which the device cannot accept tasks in one power depletion cycle is $(1-\rho)E[k]L + \frac{e_{max}}{\lambda_h}$, and then, the probability that tasks are not being queued P_i is defined as follows:

$$P_i = \frac{(1-\rho)E[k]L + \frac{e_{max}}{\lambda_h}}{E[C_d]} = 1 - \frac{\lambda_h}{\mu_d\rho}$$

To guarantee feasible service availability of IoT devices, we must minimize P_i as much as possible. It is straightforward that a larger λ_d results in a higher P_i. If the device enqueues tasks once they occur, the energy drain rate μ_d is λ_t since $P_t(0) = 1 - \frac{\lambda_t}{\mu_t}$. However, if the device accepts a task with some probability $p \in [0,1]$, we can decrease $\mu_d = \lambda_t p$, which means that an appropriate policy to control tasks is required.

We validate our analytical model through event-driven simulations written in C++. We compare the estimated probabilities with the measured ones from simulations for 5000 s assuming backlogged tasks (i.e., a large event rate like $\lambda_t = 50$) and an energy buffer of

size 10. Figure 2 is a 3D plot to present the power depletion probability according to the energy harvesting rate and duty cycle. As shown in Figure 2, our model is likely to yield the same probability as the simulation results. Looking into further detail, we investigate the number of power depletions while fixing the energy harvest rate $\lambda_h = 0.25$, as shown in Figure 3. We observe that our model more correctly estimates the number of power depletion occurrence as the duty cycle ρ becomes larger. With $\rho < 0.2$, our model does not expect any power depletion; our model presumes that the duty cycles ideally prevent the occurrence of power depletion.



(**a**) (**b**)

Figure 2. Power depletion probability: (**a**) simulation; (**b**) model.

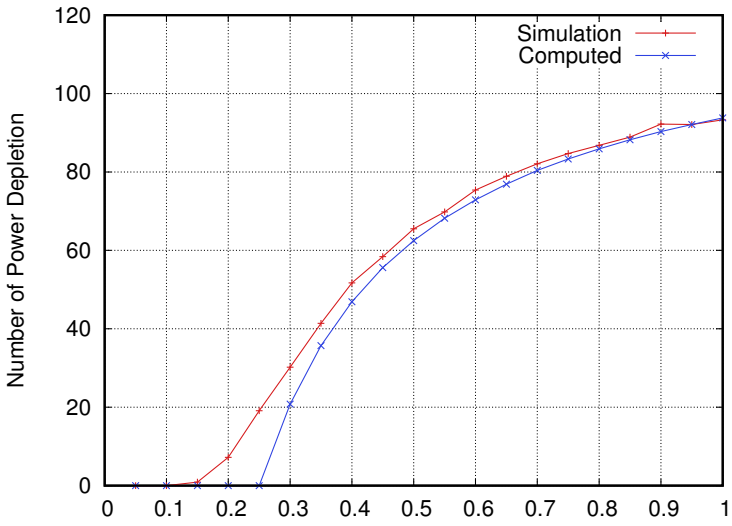


Figure 3. Number of power depletions as a function of duty cycle ($\lambda_h = 0.25$).

4. Energy Intelligence Platform Module

In this section, we present a platform module for implementing energy intelligence in energy harvesting IoT devices called the Energy Intelligence Platform Model (EIPM). The EIPM workflow has four stages: sampling, prediction, scheduling, and action, as shown in Figure 4. We develop the EIPM to process each stage through four components: a Power Manager for the sampling and scheduling stage, an Energy Predictor for the prediction stage, a Task Scheduler for the scheduling stage, and a Checkpoint Manager for one of the actions that the EIPM performs (checkpointing). Figure 5 presents the EIPM structure overview implemented for our legacy device using TinyOS [38].

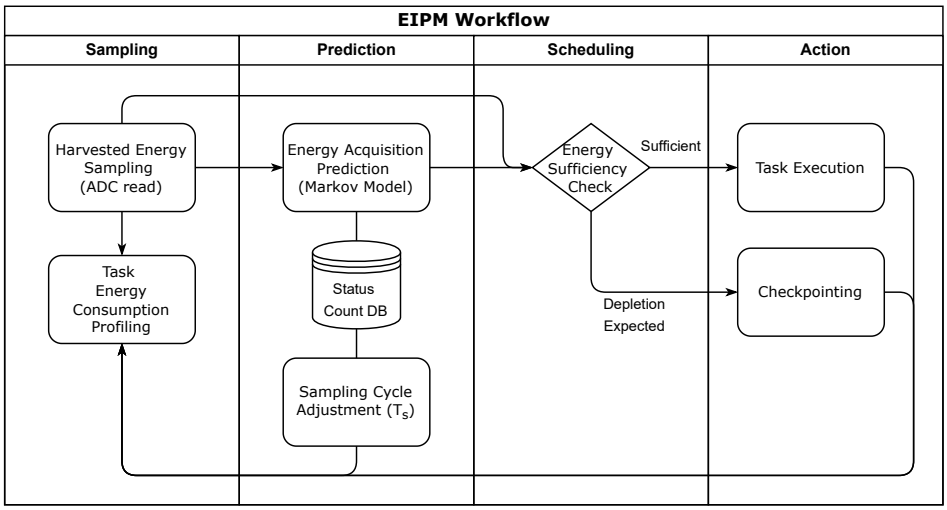


Figure 4. EIPM workflow.

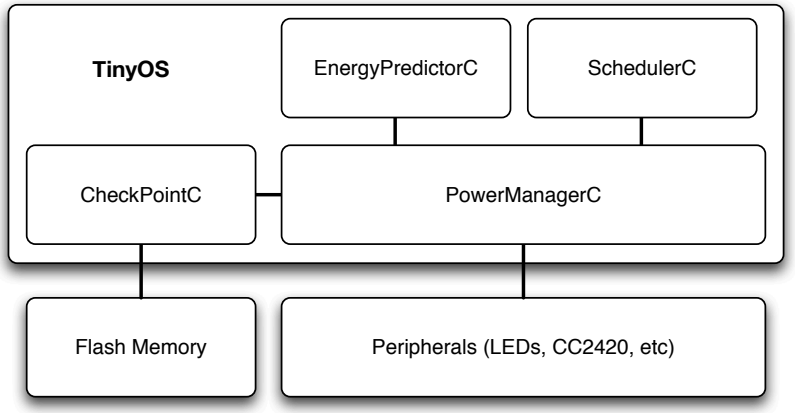


Figure 5. Implemented structure overview.

4.1. Power Manager

The Power Manager takes charge of sampling the collected amount of energy and profiling the energy usage for tasks. As with the energy consumption measurement of device operations in Table 1, the Power Manager computes the amount of harvested energy by measuring the output voltage changes for the capacitor.

Let T_s denote the length of a voltage sampling cycle, as shown in Figure 6. The amount of harvested energy at the moment of the ith sample ($E_h(i)$) is defined as follows:

$$\begin{aligned} E_h(i) &= \frac{C}{2}\left(V(i)^2 - V(i-1)^2\right) \\ &\quad -E_{LPM3} \times T_s - 2 \times E_{adc}\ , \end{aligned}$$

where the capacitance of the capacitor is C μF and the measured voltage at the ith time slot is $V(i)$ volts.

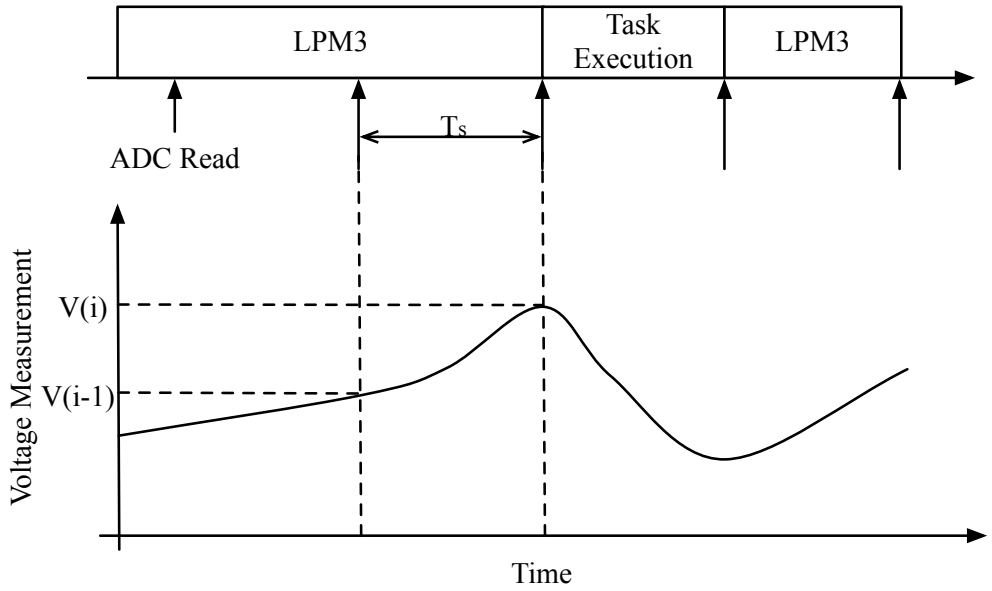


Figure 6. Estimation of harvested energy.

The online task profiles can be easily obtained by measuring the voltage change between task start and end. Assume that a node executed a task k. Denote the voltage levels measured right before and after task execution as V_1 and V_2, respectively. Then the profile of task k, $\pi(k)$, can be defined as follows:

$$\pi(k) = \frac{C}{2}(V_1^2 - V_2^2) \, .$$

The problem with such online task profiling is that it is difficult to discriminate between energy consumption and harvested energy, e.g, if harvested energy is larger than the energy consumption for a user's task, it seems that the user task does not consume any energy. But we focus on such a profile being able to reflect the energy harvesting condition. This can be used for an estimation of remaining energy after a user's task execution under a specific energy harvesting condition.

4.2. Energy Predictor

Several statistical methods such as auto-regression and the Holt–Winters algorithm [39] can be a solution to predict/forecast the future energy status based on current observations. However, these statistical approaches require large amounts of collected data and corresponding computations to build a prediction model, making them inappropriate for IoT devices with limited computing and energy capability.

For IoT devices, we suggest predicting the energy acquisition status (sufficient or insufficient) rather than the exact amount of harvested energy. First, we define an insufficient state as a case in which a device cannot preserve the current level of the energy buffer even without a task to be executed. Therefore, at the moment of the ith sampling, the energy acquisition status is in the insufficient state when the amount of harvested energy is as follows:

$$E_h(i) < E_{LPM3} \times T_s + 2 \times E_{adc} + \gamma \, ,$$

where γ is a margin for an energy-sufficient state.

Then, we build a simple two-state discrete Markov chain, as shown in Figure 7, for which states 0 and 1 mean insufficient and sufficient light, respectively. Based on the series of collected light sufficiency information, we simply compute transition probabilities by

counting the number of transitions. We utilize the corresponding stationary probabilities for each state, $Pr[0]$ and $Pr[1]$, to make decisions on device behavior.

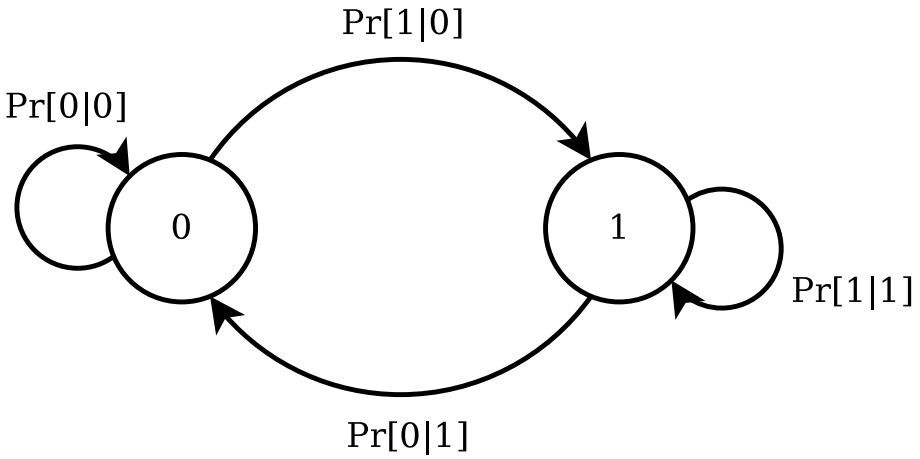


Figure 7. Energy acquisition state transition model.

However, as observations are accumulated for long time, recent transitions between insufficient and sufficient states are likely to be ignored in the model. To resolve this problem, we employ an algorithm to decay the significance of previous observations, as described in Algorithm 1. The accumulated count on each transition is multiplied by $0 < \beta < 1$ every α samplings from the last decay. To ensure the two-state Markov chain is irreducible and aperiodic and has stationary probabilities, we define the threshold $\theta = 1$ to limit the minimum value of the accumulated count of each transition.

Algorithm 1 Energy Acquisition Status Prediction

Init : $Cnt[x \to y] = 0$ for $x, y \in \{0, 1\}$
Perform the following actions every voltage sampling

$CurrState =$ Reading the light sufficiency information
if $CurrState$ is a reading after α samplings from the last decay **then**
  $Cnt[x \to y] = \beta \times Cnt[x \to y]$ for $x, y \in \{0, 1\}$
  if $Cnt[x \to y]$ for $x, y \in \{0, 1\} < \theta$ **then**
    $Cnt[x \to y] = \theta$
  end if
end if
Cnt[$PrevState \to CurrState$] $+= 1$
$PrevState = CurrState$
Calculate $Pr[0]$, $Pr[1]$ from current Cnt[$x \to y$] for $x, y \in \{0, 1\}$

Since the EIPM makes decisions based on voltage samples and the corresponding states, it is important for the EIPM to determine an appropriate sampling cycle in order to prevent unnecessary energy usage while collecting enough information to control a device; a large sampling cycle can result in delayed responses with regard to energy-sufficient conditions, and a short cycle can incur too much energy consumption for ADC readings and corresponding computations. To resolve this issue, the EIPM adjusts the sampling cycle T_s based on the number of consecutive sufficient or insufficient light states. Let us assume that the set of possible lengths for the sampling cycle is $L = \{l_0, l_1, l_2, ..., l_{max}\}$. Algorithm 2 presents the pseudocode for sampling cycle adjustment.

Algorithm 2 Sampling Cycle Adjustment

Init : $T_s = l_{max}$, $idx = max$
Perform the following actions every time slot i

if $E_h(i)$ indicates light insufficiency **then**
  if there are ψ_d consecutive insufficient light states **then**
    $idx = idx - 1$ if $idx > 0$
  end if
else
  if there are ψ_u consecutive sufficient light states **then**
    $idx = idx + 1$ if $idx < max$
  end if
end if
$T_s = l_{idx}$

4.3. Task Scheduler

Suppose that a device executes tasks without any control. In that case, it can result in inefficient energy usage or an unexpected energy depletion due to a power drain that is too large from the energy buffer. So the EIPM needs a component to decide whether a device executes a task based on the energy harvesting status.

The Task Scheduler controls device task execution based on the energy sampling and task profiling from the Power Manager. It decides whether a device executes a task based on the energy harvesting conditions to prevent inefficient energy usage and unexpected energy depletion. Instead of finding the most proper task to be executed, like in [40], the Task Scheduler simply focuses on the decision about whether or not to execute the task at the head of the task queue. It aims to guarantee that the capacitor voltage after task execution is larger than or equal to the minimum required voltage for operating the device. Denote the minimum voltage for a device as V_{dev} (the V_{dev} of our device is 1.8 V). Assume that the Task Scheduler controls the execution of task k at the ith sampling. The expected voltage after task k execution, $V_{after}(k)$, is as follows:

$$V_{after}(k) = \sqrt{V(i)^2 - \frac{2}{C}\pi(k)}\,,$$

where $V(i)$ is a measured voltage level at the ith sampling.

Task k can be executed only if V_{after} is larger than or equal to V_{dev}; thus, V_{dev} can be a minimum threshold for the Task Scheduler to compare to V_{after}. Note that the EIPM also seeks to make a checkpoint before power depletion happens. Therefore, the threshold to execute the task k should consider the amount of energy for checkpointing if necessary (i.e., if light is continuously insufficient). To reflect this, the Task Scheduler adjusts the threshold to be close to the voltage level required for checkpointing (V_{chk}) as the stationary probability of the insufficient light state $Pr[0]$ increases. Then, it executes task k if $V_{after}(k) \geq V_{threshold}$.

$$V_{threshold} = V_{dev} + (V_{chk} - V_{dev}) \times Pr[0]$$

However, several system tasks are critical for device operation, such as timer expiration and interrupts. If the Task Scheduler blocks such tasks, a node cannot work correctly. To avoid blocking this type of task, the Task Scheduler divides tasks into two categories: system tasks and user tasks. As shown in Figure 8, the Task Scheduler manages two queues for each type of task. Tasks in the system task queue are under the default queue management policy, e.g., they are preemptive against user tasks, and the Task Scheduler does not control them at all. In contrast, a device executes user tasks only when the Task Scheduler checks that the energy condition satisfies the above constraints with the help of the Power Manager.

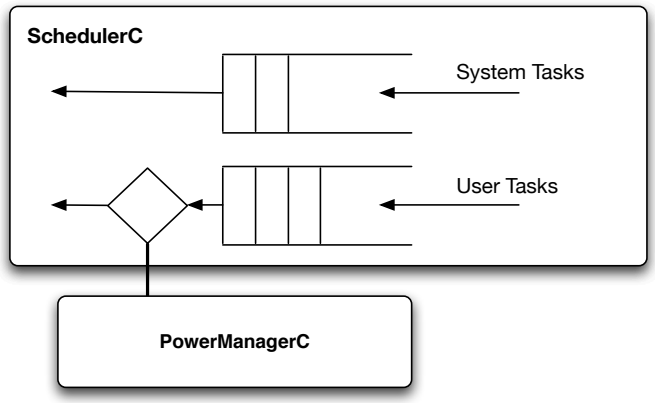


Figure 8. Task queues.

4.4. Checkpoint Manager

Checkpointing is a typical technique to save the memory state on non-volatile storage in operating systems; restarting from the checkpointed state allows applications to recover their memory state to the last checkpoint. In the EIPM, the Checkpoint Manager creates such a rollback point to keep the device state information regardless of power depletion. It is straightforward that deferring a checkpoint to just before power depletion is energy efficient since saving a checkpoint triggers additional energy consumption. However, to be tolerant of unexpected failures, it can be better to do an eager checkpoint, which saves the memory state periodically or at certain points in the application's execution. To decide an appropriate time for a checkpoint, the Checkpoint Manager collaborates with the Energy Predictor; it makes a checkpoint when the stationary probability for the insufficient light state is larger than a particular threshold, e.g., we use $Pr[0] > 0.8$ as a triggering condition for a checkpoint in our implementation. Also, as discussed for the Task Scheduler, the Checkpoint Manager also check whether the capacitor voltage level is larger than V_{chk}, which satisfies the following condition:

$$\frac{C}{2}\left(V_{chk}^2 - {V_f}^2\right) \geq (E_{fw} + E_{mem}) \times S\,,$$

where V_f is the minimum voltage level for operating flash memory, and S is the size of a checkpoint.

If checkpoints are created too frequently, most of them are useless since they do not contain any new information compared with the previous one. To avoid unnecessary checkpoints, the Checkpoint Manager sets an indicating value for each checkpoint (*checkpoint TTL*), which specifies how long the corresponding checkpoint is worthwhile. Assuming more task executions make a checkpoint obsolete, the Checkpoint Manager decreases the value of the *checkpoint TTL* (reduces it by one) every task execution. Then, if the *checkpoint TTL* becomes zero, the Checkpoint Manger makes a new one if necessary.

5. Evaluation

5.1. Real Device Experiments

To evaluate the performance of the EIPM on our device, we build a simple application that calculates the average values from sensed data and transmits them to a basestation. The communication component is turned on only when a node tries to transmit a packet, and each transmitted packet contains a sequence number to identify how many values are used for computing the average. We compare the EIPM with a naive threshold-based approach to control task execution. In the case of the threshold-based scheme, the device periodically samples its voltage level and determines whether to execute a task based on the measured voltage level.

We run outdoor experiments using the simple applications compiled with each approach for 24 h. We select parameter values for the EIPM and the threshold-based approach as shown in Tables 2 and 3, which yield feasible and comparable performance from preliminary experiments. Recall that the EIPM adjusts the sampling cycle according to the energy harvesting conditions. In contrast, the threshold-based approach uses a fixed value for the sample cycle; we choose the medium value of L for the EIPM. In the case of the threshold for task execution, the naive approach uses the voltage value for operating flash memory as a threshold.

Table 2. EIPM parameters.

Parameter	Description	Value
α	Threshold for decaying	30
β	Decaying factor	0.5
γ	Margin for light sufficiency	300 µJ
ψ_u	Threshold for increasing T_s	10
ψ_d	Threshold for decreasing T_s	2
S	Set of possible T_s	$\{50\text{ ms}, 100\text{ ms}, ..., 450\text{ ms}, 500\text{ ms}\}$
TTL	Checkpoint lifetime	15 task executions

Table 3. Threshold-based approach parameters.

Parameter	Value
Sampling cycle T_s	300 ms
Threshold for task execution	2750 mV

5.1.1. Task Execution Rate

To investigate whether the EIPM successfully manages energy usage while executing a feasible number of tasks, we explore the task execution rate and the voltage level before task execution. Figure 9 depicts the average task execution rate during 24 h. The application compiled with the EIPM executes tasks more aggressively than the one with the threshold-based approach while successfully decreasing the execution rate as the evening goes on (light becomes insufficient). In contrast, the threshold-based approach yields about 12,000 task executions per hour, which is much lower than that with the EIPM. Recall that both approaches execute a task after measuring the voltage level. Therefore, in the case of the threshold-based approach without sampling cycle adjustments, its maximum task execution rate is fixed just by the pre-determined sampling cycle even under sufficient light condition (12:00 to 19:00). Therefore, the naive approach fails to fully utilize harvested energy for processing tasks, while the EIPM makes a device more aggressive under sufficient light.

Figure 10 shows the average voltage level before task execution, which represents the amount of energy stored in the capacitor before task execution. As discussed with regard to the task execution rate, the EIPM more aggressively utilizes harvested energy than the threshold-based approach under the sufficient light state. Also, in the case of the EIPM, the voltage level before task execution increases as the evening goes on. This is because the EIPM increases the voltage threshold for task execution when it needs to make a checkpoint due to potential power depletion (light becomes insufficient).

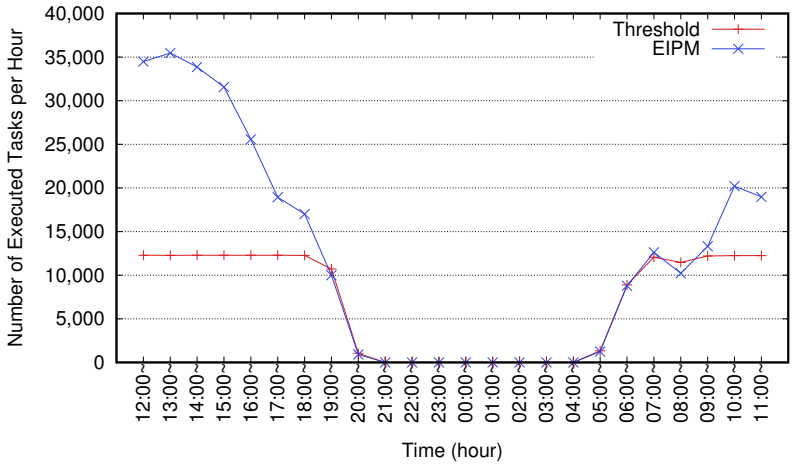


Figure 9. Average task execution rates.

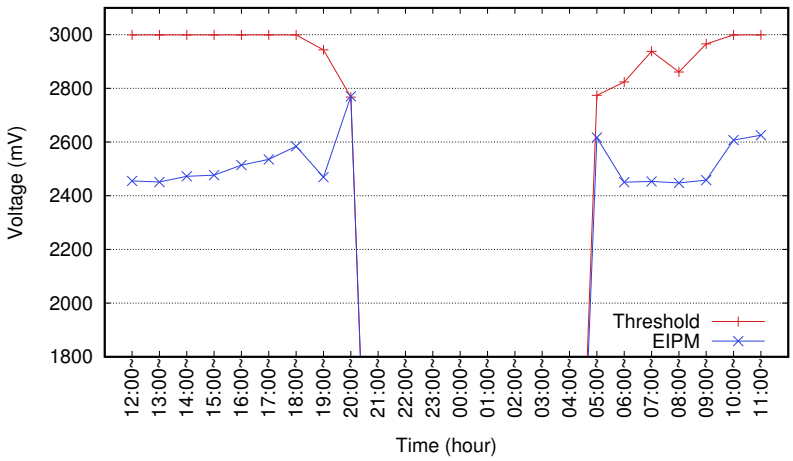


Figure 10. Average voltage levels before task execution.

5.1.2. Checkpoints

To examine how efficiently the EIPM creates checkpoints while avoiding unnecessary ones, we investigate the number of checkpoints and the amount of lost information due to power depletion. Since the threshold-based approach does not have a mechanism to create checkpoints, we assume that it makes a checkpoint when the voltage level becomes the minimum threshold to operate flash memory. Therefore, we count the number of expected checkpoints based on the threshold during the last hour before power depletion.

Table 4 presents the average number of checkpoints and lost information (the number of executed tasks from the last checkpoint to power depletion). The EIPM creates 13 checkpoints while losing 9 states on average. The amount of lost information with 13 checkpoints is smaller than that of the predetermined *checkpoint TTL = 15*, which means that the last checkpoint is still relevant. These results show that the EIPM efficiently mitigates unnecessary checkpoints while preserving a valid checkpoint. In contrast, since the threshold-based approach uses the voltage level to determine whether to make a checkpoint, it yields 84 checkpoints on average, which is much more frequent than the EIPM. Also, the threshold-based approach does not guarantee successful checkpoint creation because it does not check energy availability for completing a checkpoint.

Table 4. Checkpoint statistics.

Scheme	Description	Count
Threshold-based	Number of expected checkpoints	84
EIPM	Number of checkpoints	13
	Amount of lost information	9

5.2. Simulations

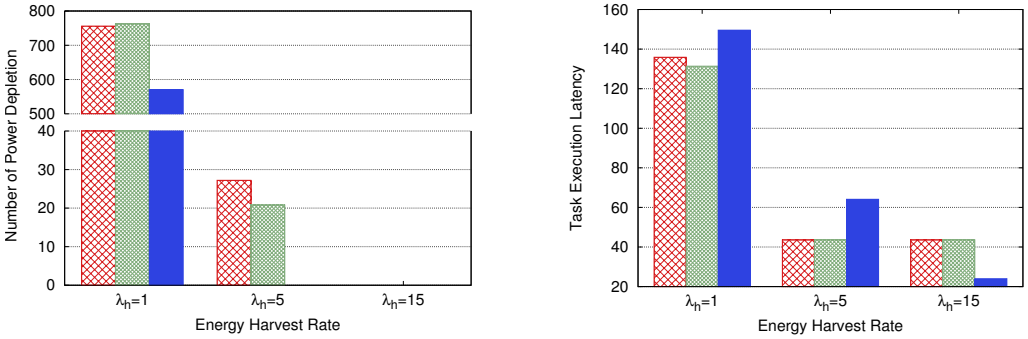
To compare the EIPM with other approaches in more detail, we conduct event-driven simulation based on the assumptions of the analytical model in Section 3.3. For the comparison study, we select the threshold-based approach in the implementation experiments and a control method based on statistical time series prediction using the Holt–Winters algorithm [39]. Note that in the simulations, we simplify the EIPM to utilize energy level measurements instead of voltage. In the case of the Holt–Winters algorithm, we use a smoothing parameter of 0.25 and utilize a series of 20 samples for the algorithm to predict the one-step-ahead energy level.

5.2.1. Static Environments

First, we explore the simulations for 10,000 units of time with the static environment parameter settings in Table 5. Each setting of the energy harvesting rate represents scarce ($\lambda_h = 1$), moderate ($\lambda_h = 5$), and sufficient ($\lambda_h = 15$) energy harvesting conditions. Figure 11 and Table 6 present the average number of power depletions, the average task execution latencies (i.e., the interval between task enqueuing and execution), and device turn-on durations collected from five simulation runs. As shown in Figure 11a, the EIPM successfully prevents power depletions under moderate and sufficient conditions. In the case of scarce conditions, the EIPM experiences inevitable power depletions but mitigates them as much as possible. By doing so, the EIPM guarantees longer device turn-on duration than the others, as shown in Table 6. In contrast, the other approaches suffer from power depletions even when the energy harvesting condition is moderate. Compared with the threshold-based approach, the Holt–Winters-algorithm-based one yields slightly fewer power depletions but still yields many more than the EIPM. Figure 11b shows that task execution latency decreases as the energy harvest rate becomes larger because enough energy in the energy buffer enables a device to execute tasks more frequently. Since the EIPM regulates task executions to avoid power depletions, it yields longer task execution latency than the others under scarce and moderate conditions. However, once the EIPM detects sufficient conditions, it more aggressively executes tasks; thus, its average task execution latency becomes much shorter than the others.

Table 5. Simulation parameters for static environments.

Parameter	Description	Value
ρ	Ratio of active period in one cycle	0.7
L	Length of cycle	10
K	Size of energy buffer	10
Q	Size of task queue	100
λ_t	Task arrival rate	10
μ_t	Task service rate	15
λ_h	Energy harvesting rate	1, 5, 10



(**a**) (**b**)

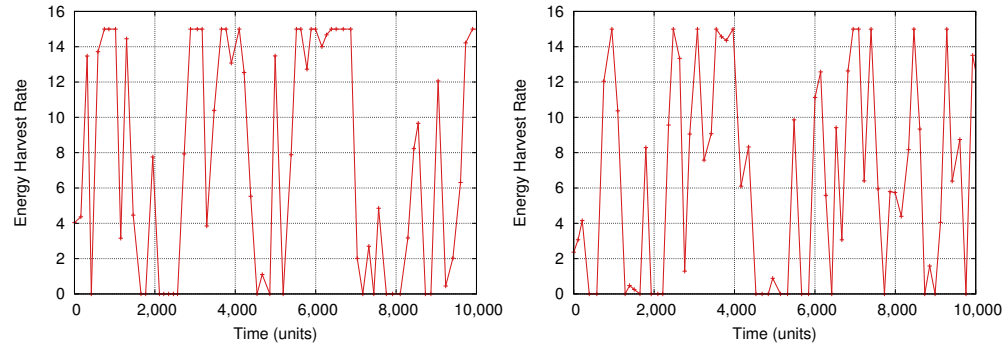
Figure 11. Power depletion and task execution latency in static environments: (**a**) number of power depletions; (**b**) task execution latency.

Table 6. Device turn-on duration (units of time) in static environments.

	Energy Harvesting Condition		
Method	**Scarce ($\lambda_h = 1$)**	**Moderate ($\lambda_h = 5$)**	**Sufficient ($\lambda_h = 15$)**
Threshold	12.21	348.06	always
Holt–Winters	12.09	451.60	always
EIPM	16.05	always	always

5.2.2. Dynamic Environments

Next, we investigate the behavior of the EIPM and the other approaches under dynamic environments wherein the energy harvesting rate changes from time to time. To this end, we artificially generate five time-series traces by continuously changing the energy harvesting rates based on the previous ones for 10,000 units of time. We set the minimum and maximum values of the energy harvesting rate to 0 and 15, respectively. Figure 12 presents the energy harvesting rate traces of the selected scenarios.



(**a**) (**b**)

Figure 12. Example scenarios using dynamic energy harvesting rate: (**a**) scenario 2; (**b**) scenario 5.

Table 7 presents the number of power depletions, device turn-on duration, and task execution latency on average. Note that we sort the scenarios by the increasing order of the number of power depletions. We expect that scenario 1 results in similar behavior to the sufficient scenario in the static environment, scenarios 2–4 relates to the moderate one, and scenario 5 relates to the scarce one. Similar to the result for the static environments, the EIPM entirely prevents power depletions in scenarios 1–4 and suffers from a relatively small number of inevitable power depletions in scenario 5. In contrast, the other approaches severely suffer from power depletions, except for in scenario 1. The other metrics also correspondingly behave similarly to those in the static environments. The EIPM yields the

largest average device turn-on time for all scenarios, followed by the Holt–Winters and threshold-based approaches. For example, Figure 13 presents the cumulative distribution function of the device turn-on duration in scenario 5. As shown in Figure 13, the threshold and Holt–Winters approaches exhibit device turn-on times of up to 33 units while the EIPM does up to 681. In the case of task execution latency, the EIPM always yields longer latencies. As with sufficient conditions in static environments, the EIPM successfully shortens task execution latency as the energy harvesting condition becomes good, as in scenario 1, although it is still slightly longer than the others to prevent power depletions. These results demonstrate that the EIPM successfully manages device energy consumption to mitigate power depletions while preserving feasible task execution performance.

Table 7. Results with dynamic environments.

a. Number of Power Depletions					
	Scenario				
Method	**1**	**2**	**3**	**4**	**5**
Threshold	8	191	313	409	1015
Holt–Winters	6	155	254	365	1001
EIPM	0	0	0	0	66
b. Device Turn-on Duration (Units of Time)					
	Scenario				
Method	**1**	**2**	**3**	**4**	**5**
Threshold	1078.32	51.33	31.50	24.05	9.42
Holt–Winters	1319.42	63.53	38.23	26.91	9.57
EIPM	always	always	always	always	147.08
c. Task Execution Latency (Units of Time)					
	Scenario				
Method	**1**	**2**	**3**	**4**	**5**
Threshold	43.61	43.78	44.27	44.71	58.36
Holt-Winters	43.58	44.01	44.62	45.12	58.64
EIPM	59.11	71.12	72.55	73.59	77.16

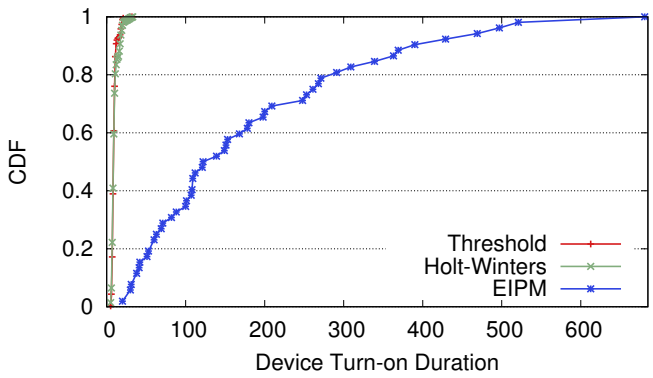


Figure 13. CDF of device turn-on duration (scenario 5).

6. Conclusions

This paper proposes the EIPM: a platform module to implement energy intelligence for energy harvesting IoT devices. First, we devise a simple energy measurement scheme and prediction model to realize the EIPM in IoT devices with limited capabilities. Using lightweight energy sampling and prediction models, the EIPM enables devices to adjust their task execution rates according to energy harvesting conditions. Also, the EIPM provides a checkpoint mechanism based on predictions of energy conditions, allowing devices to cope with inevitable power depletion events. Our simulation results prove that under various environments, the EIPM almost perfectly prevents power depletions and enables devices to turn on for a longer time; e.g., compared to the other approaches, the EIPM reduces the number of power depletions by 93.4% if those are inevitable and provides 15.6× longer device turn-on duration while yielding 32.6% increased task execution latency as a tradeoff, which is feasible. Also, we implement a proof of concept for the EIPM in a wireless sensor node as a representative energy harvesting IoT device. Our experiment results demonstrate that the EIPM successfully achieves the design goal of aggressively processing tasks with the consumption of abundant energy under good harvesting conditions and preparing a checkpoint with restricted task execution otherwise.

Author Contributions: Conceptualization, Y.-m.K. and Y.-s.L.; methodology, Y.-m.K. and Y.-s.L.; software, Y.-m.K.; validation, Y.-s.L.; formal analysis, Y.-m.K.; writing—original draft preparation, Y.-m.K.; writing—review and editing, Y.-s.L.; visualization, Y.-m.K.; supervision, Y.-s.L.; project administration, Y.-s.L.; funding acquisition, Y.-s.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Sungshin Women's University Research grant H20240066.

Data Availability Statement: Data is available upon request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Jamshed, M.A.; Ali, K.; Abbasi, Q.H.; Imran, M.A.; Ur-Rehman, M. Challenges, Applications, and Future of Wireless Sensors in Internet of Things: A Review. *IEEE Sens. J.* **2022**, *22*, 5482–5494. [CrossRef]
2. Rahmani, H.; Shetty, D.; Wagih, M.; Ghasempour, Y.; Palazzi, V.; Carvalho, N.B.; Correia, R.; Costanzo, A.; Vital, D.; Alimenti, F.; et al. Next-Generation IoT Devices: Sustainable Eco-Friendly Manufacturing, Energy Harvesting, and Wireless Connectivity. *IEEE J. Microwaves* **2023**, *3*, 237–255. [CrossRef]
3. Sah, D.K.; Hazra, A.; Kumar, R.; Amgoth, T. Harvested Energy Prediction Technique for Solar-Powered Wireless Sensor Networks. *IEEE Sens. J.* **2023**, *23*, 8932–8940. [CrossRef]
4. Hazra, A.; Donta, P.K.; Amgoth, T.; Dustdar, S. Cooperative Transmission Scheduling and Computation Offloading with Collaboration of Fog and Cloud for Industrial IoT Applications. *IEEE Internet Things J.* **2023**, *10*, 3944–3953. [CrossRef]
5. Singh, J.; Kaur, R.; Singh, D. A survey and taxonomy on energy management schemes in wireless sensor networks. *J. Syst. Archit.* **2020**, *111*, 101782. [CrossRef]
6. Donta, P.K.; Rao, B.S.P.; Amgoth, T.; Annavarapu, C.S.R.; Swain, S. Data Collection and Path Determination Strategies for Mobile Sink in 3D WSNs. *IEEE Sens. J.* **2020**, *20*, 2224–2233. [CrossRef]
7. Ma, D.; Lan, G.; Hassan, M.; Hu, W.; Das, S.K. Sensing, Computing, and Communications for Energy Harvesting IoTs: A Survey. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1222–1250. [CrossRef]
8. Ruan, T.; Chew, Z.J.; Zhu, M. Energy-Aware Approaches for Energy Harvesting Powered Wireless Sensor Nodes. *IEEE Sens. J.* **2017**, *17*, 2165–2173. [CrossRef]
9. Sah, D.K.; Amgoth, T. Renewable energy harvesting schemes in wireless sensor networks: A Survey. *Inf. Fusion* **2020**, *63*, 223–247. [CrossRef]
10. Ibrahim, H.H.; Singh, M.J.; Al-Bawri, S.S.; Ibrahim, S.K.; Islam, M.T.; Alzamil, A.; Islam, M.S. Radio Frequency Energy Harvesting Technologies: A Comprehensive Review on Designing, Methodologies, and Potential Applications. *Sensors* **2022**, *22*, 4144 . [CrossRef]
11. Costa, M.S.; Manera, L.T.; Moreira, H.S. Study of the light energy harvesting capacity in indoor environments. In Proceedings of the 2019 4th International Symposium on Instrumentation Systems, Circuits and Transducers (INSCIT), Sao Paulo, Brazil, 26–30 August 2019; pp. 1–4. [CrossRef]
12. Bairagi, S.; ul Islam, S.; Shahadat, M.; Mulvihill, D.M.; Ali, W. Mechanical energy harvesting and self-powered electronic applications of textile-based piezoelectric nanogenerators: A systematic review. *Nano Energy* **2023**, *111*, 108414. [CrossRef]

13. Becker, T.; Kiziroglou, M.E.E. *Energy Harvesting for a Green Internet of Things*; PSMA Energy Harvesting Technical Committee: Long Beach, CA, USA, 2021.
14. Dondi, D.; Bertacchini, A.; Brunelli, D.; Larcher, L.; Benini, L. Modeling and Optimization of a Solar Energy Harvester System for Self-Powered Wireless Sensor Networks. *IEEE Trans. Ind. Electron.* **2008**, *55*, 2759–2766. [CrossRef]
15. Sharma, H.; Haque, A.; Jaffery, Z.A. Modeling and Optimisation of a Solar Energy Harvesting System for Wireless Sensor Network Nodes. *J. Sens. Actuator Netw.* **2018**, *7*, 40. [CrossRef]
16. Pasricha, S.; Ayoub, R.; Kishinevsky, M.; Mandal, S.K.; Ogras, U.Y. A Survey on Energy Management for Mobile and IoT Devices. *IEEE Des. Test* **2020**, *37*, 7–24. [CrossRef]
17. Corke, P.; Valencia, P.; Sikka, P.; Wark, T.; Overs, L. Long-duration solar-powered wireless sensor networks. In Proceedings of the 4th Workshop on Embedded Networked Sensors, EmNets '07, New York, NY, USA, 25–26 June 2007; pp. 33–37. [CrossRef]
18. Simjee, F.; Sharma, D.; Chou, P.H. Everlast: Long-life, supercapacitor-operated wireless sensor node. In Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, SenSys '05, New York, NY, USA, 2–4 November 2005; pp. 315–315. [CrossRef]
19. Noh, D.K.; Wang, L.; Yang, Y.; Le, H.K.; Abdelzaher, T. Minimum Variance Energy Allocation for a Solar-Powered Sensor System. In Proceedings of the 5th IEEE International Conference on Distributed Computing in Sensor Systems, DCOSS '09, Berlin/Heidelberg, Germany, 8–10 June 2009; pp. 44–57. [CrossRef]
20. Moser, C.; Thiele, L.; Brunelli, D.; Benini, L. Adaptive power management in energy harvesting systems. In Proceedings of the Conference on Design, Automation and Test in Europe, DATE '07, San Jose, CA, USA, 16–20 April 2007; pp. 773–778.
21. Vigorito, C.M.; Ganesan, D.; Barto, A.G. Adaptive control of duty cycling in energy-harvesting wireless sensor networks. In Proceedings of the 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON 2007, San Diego, CA, USA, 18–21 June 2007; pp. 21–30.
22. Sarang, S.; Stojanović, G.M.; Drieberg, M.; Stankovski, S.; Jeoti, V. Energy Neutral Operation based Adaptive Duty Cycle MAC Protocol for Solar Energy Harvesting Wireless Sensor Networks. In Proceedings of the 2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring), Helsinki, Finland, 19–22 June 2022 ; pp. 1–6. [CrossRef]
23. Bengheni, A. A Multi-Threshold Energy approach for Energy Harvesting WSN. In Proceedings of the 2022 2nd International Conference on New Technologies of Information and Communication (NTIC), Mila, Algeria, 21–22 December 2022; pp. 1–5. [CrossRef]
24. Havrlík, M.; Libra, M.; Poulek, V.; Kouřím, P. Analysis of Output Signal Distortion of Galvanic Isolation Circuits for Monitoring the Mains Voltage Waveform. *Sensors* **2022**, *22*, 7769. [CrossRef] [PubMed]
25. Sandhu, M.M.; Khalifa, S.; Jurdak, R.; Portmann, M. Task Scheduling for Energy-Harvesting-Based IoT: A Survey and Critical Analysis. *IEEE Internet Things J.* **2021**, *8*, 13825–13848. [CrossRef]
26. Zou, T.; Lin, S.; Feng, Q.; Chen, Y. Energy-Efficient Control with Harvesting Predictions for Solar-Powered Wireless Sensor Networks. *Sensors* **2016**, *16*, 53. [CrossRef] [PubMed]
27. Cammarano, A.; Petrioli, C.; Spenza, D. Online Energy Harvesting Prediction in Environmentally Powered Wireless Sensor Networks. *IEEE Sens. J.* **2016**, *16*, 6793–6804. [CrossRef]
28. Peng, S.; Low, C. Prediction free energy neutral power management for energy harvesting wireless sensor nodes. *Ad Hoc Netw.* **2014**, *13*, 351–367. [CrossRef]
29. Ashraf, N.; Faizan, M.; Asif, W.; Qureshi, H.K.; Iqbal, A.; Lestas, M. Energy management in harvesting enabled sensing nodes: Prediction and control. *J. Netw. Comput. Appl.* **2019**, *132*, 104–117. [CrossRef]
30. Li, Y.; Jia, Z.; Li, X. Task Scheduling Based on Weather Forecast in Energy Harvesting Sensor Systems. *IEEE Sens. J.* **2014**, *14*, 3763–3765. [CrossRef]
31. Sah, D.K.; Hazra, A.; Mazumdar, N.; Amgoth, T. An Efficient Routing Awareness Based Scheduling Approach in Energy Harvesting Wireless Sensor Networks. *IEEE Sens. J.* **2023**, *23*, 17638–17647. [CrossRef]
32. Singla, P.; Sarangi, S.R. A survey and experimental analysis of checkpointing techniques for energy harvesting devices. *J. Syst. Archit.* **2022**, *126*, 102464. [CrossRef]
33. Gummadi, R.; Kothari, N.; Millstein, T.; Govindan, R. Declarative failure recovery for sensor networks. In Proceedings of the 6th International Conference on Aspect-Oriented Software Development, AOSD '07, New York, NY, USA, 12–16 March 2007; pp. 173–184. [CrossRef]
34. Österlind, F.; Dunkels, A.; Voigt, T.; Tsiftes, N.; Eriksson, J.; Finne, N. Sensornet Checkpointing: Enabling Repeatability in Testbeds and Realism in Simulations. In Proceedings of the 6th European Conference on Wireless Sensor Networks, EWSN '09, Berlin/Heidelberg, Germany, 11–13 February 2009; pp. 343–357. [CrossRef]
35. Ransford, B.; Clark, S.S.; Salajegheh, M.; Fu, K. Getting Things Done on Computational RFIDs with Energy-Aware Checkpointing and Voltage-Aware Scheduling. In Proceedings of the Workshop on Power Aware Computing and Systems, HotPower, USENIX Association, San Diego, CA, USA, 8–10 December 2008.
36. Lajara, R.; Pelegrí-Sebastiá, J.; Solano, J.J.P. Power Consumption Analysis of Operating Systems for Wireless Sensor Networks. *Sensors* **2010**, *10*, 5809–5826. [CrossRef]
37. TI eZ430-RF2500-SEH Development Tool User Guide. Available online: http://focus.ti.com/lit/ug/slau273c/slau273c.pdf (accessed on 14 June 2024).
38. TinyOS. Available online: http://www.tinyos.net/ (accessed on 14 June 2024).

39. Brockwell, P.J.; Davis, R.A. *Introduction to Time Series and Forecasting*; Springer: Berlin/Heidelberg, Germany, 2002.
40. Moser, C.; Brunelli, D.; Thiele, L.; Benini, L. Real-time scheduling for energy harvesting sensor nodes. *Real-Time Syst.* **2007**, *37*, 233–260. [CrossRef]