UNIVERSITY OF CALIFORNIA
RIVERSIDE

Energy Scheduling for Task Execution on Intermittently-Powered Devices

A Thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Electrical Engineering

by

Mohsen Karimi

March 2021

Thesis Committee:

Dr. Hyoseung Kim, Chairperson
Dr. Daniel Wong
Dr. Shaolei Ren

The Thesis of Mohsen Karimi is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I would like to thank my thesis advisor Dr. Hyoseung Kim for his dedicated guidance, support, and encouragement during the course of my research. I would also like to thank my committee members Dr. Daniel Wong and Dr. Shaolei Ren for their valuable advice and efforts in evaluating my work. I would like to thank my lab mates: Abdulrahman Bukhari, Daniel Enright, Hyunjong Choi, Seyedmehdi Hosseinimotlagh, Yecheng Xiang, and Yidi Wang for their generous help in different aspects. It is an honor for me to thank my parents, especially my mother, for providing me with unconditional love and endless support during the course of my life. Lastly, I would like to give my sincere thank to my dearest wife Fahimeh for her genuine love, endless patience, and great kindness throughout my entire journey.

ABSTRACT OF THE THESIS

Energy Scheduling for Task Execution on Intermittently-Powered Devices

by

Mohsen Karimi

Master of Science, Graduate Program in Electrical Engineering
University of California, Riverside, March 2021
Dr. Hyoseung Kim, Chairperson

Intermittently-powered embedded devices are getting widespread attention these days. However, running real-time tasks on these devices remains a challenging problem due to the lack of support for data freshness guarantee, time keeping, and schedulability analysis. Furthermore, while many sensing applications require low-level sensor readings to be done in an atomic way, meaning that the operations cannot be suspended and resumed later, existing solutions for intermittently-powered devices assume compute-only workloads and disregard such sensor operations. In this research, we provide an energy harvesting model for intermittently-powered devices, and based on that, propose techniques to utilize intermittent power sources efficiently and to schedule real-time periodic tasks that need atomic operations. We present a hardware-software co-design scheme to keep track of time and to ensure the periodic execution of sensing tasks. We provide schedulability analysis to determine if a single task is schedulable in a given charging setup, and extend this idea to scheduling multiple tasks. As a proof-of-concept, we design a custom programmable RFID tag device, called R'tag, and demonstrate the effect of our proposed techniques in a realistic sensing application. We also show the device parameters' effect on energy harvesting performance in simulation. We compare the baseline approach and the proposed method both in simulation and experimental evaluations. Evaluation results, both on simulation and experiment, verify that the proposed method outperforms the baseline approach in terms of task scheduling, time keeping, and periodic sensing.

Contents

List of Figures

Chapter 1

Introduction

Battery-less intermittently-powered devices have gained much interest due to their potential to facilitate wireless sensor networks and Internet of Things (IoT). These devices, powered by an intermittent power source such as sunlight, heat, vibration, radio signal, etc., have widespread applications spanning from smart home and buildings to agriculture and health monitoring[2, 5]. Owing to the fact that they don't have a battery that needs to be changed, they can continue to run more than a decade without so much maintenance, unless they wear out physically. Furthermore, they can be deployed in extreme environments where batteries do not perform well, e.g., hot or cold conditions.

Data freshness (timeliness of data/task) is the foremost requirement of many sensing tasks, and this is also the case for those running on intermittently-powered devices. In most of the sensing applications, sudden change in the environment should be detected and reacted at the right time. If data sensed long after the actual occurrence of the event, the reaction may not be effective or in some cases it can be harmful, e.g., in health monitor-

This work was presented at the Embedded Operating Systems Workshop (EWiLi), 2019. [10]

ing. In many cases, if sensing operation is divided into multiple sub-tasks and executed over multiple charging/discharging cycles of the device, then the data collected earlier may become stale in the middle of processing. In such a case, it would be better to collect new data instead of processing the remaining part using the stale data.

There are multiple challenges and limitations regarding sensing applications running on intermittently-powered devices. One of the challenges is the long indivisible execution time required for many sensor peripherals. In plenty of sensor devices, it takes a long time to drive sensor and collect data out of it. Therefore, the device needs to run for a relatively long time without intermittent power disruptions. In these cases, if the power goes off, all the operation should start over again from the beginning. Prior work, has focused on guaranteeing "forward progress" of intermittently-powered devices, by checkpointing intermediate results in non-volatile memories[3, 15] or by providing a programming language like MayFly[7] to divide the tasks into fragments to guarantee data consistency. However, these methods cannot be used for sensor devices that need continuous execution time.

Another challenge regarding task execution on intermittently powered battery-less devices is the burst execution without considering data freshness. These devices normally can run whenever there is a power source available. In sensor reading applications, when the charging period is small, we would having multiple samples that are alike each other and do not have so much value in sensing applications. In these cases, the device also drains all the energy accumulated by the energy storage and goes off until the next time energy source becomes available again.

Furthermore, in most of the work regarding the battery-less devices [13, 9] there is no notion of time because the device would go off when it exhausts the power and when it wakes up again it cannot keep track of the last time it was turned off. The best work that has considered time is [7]. However, it relies on the decaying effect of capacitors and can measure for less than 20 minutes , which is not sufficient for devices that run for long time and have long periodic sensor tasks.

We propose a scheduling approach to meet the timing requirements of sensor tasks even under intermittent power-source availability. Unlike prior work, our work allows storing more energy than what is required to just turn on the microcontroller. Our work computes the required level of voltage level to complete the given amount of computation, including sensor operations, and makes the device wait until that level is reached. This enables the execution of a long indivisible sensor operation, which was not possibly doable by existing approaches. Furthermore, our work enables time keeping, which is important to check the freshness of obtained data from sensors. We provide analytical guarantees of the schedulability of a single task with a formal proof which can also be extended to a multi-task scenario. We develop a prototype hardware and software system (PCB) and do the the experiment on chemiresistive nanosensor and commercial sensors on the prototype. The experimental results also verifies the theory we provided.

Chapter 2

System Model

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

2.1 Hardware characteristics

Each energy harvesting device has four main units: energy harvester unit, energy storage unit, power management unit, energy conversion unit, and processing unit. An energy harvester unit converts the energy coming from the power source (e.g. light, wind, vibration, and RF signal) to a type of energy, e.g. voltage and electrical current, that can be accumulated in the energy storage unit. A storage unit, usually consists of capacitors, that can store the energy that can be used to power the system. In energy conversion unit, voltage converters, rectifier, and regulator are used to convert the energy source to desired voltage usable for the electrical circuit. Power management unit controls when to store energy and when to use the energy to power up the system. Finally, in the processing unit, microcontrollers and sensors are used to do the desired tasks. Fig. 2.1 shows a general block diagram of a radio frequency identification (RFID) device.

In the hardware block diagram shown in Fig. 2.1, each unit is specifically designed to harvest the RF energy to be used for the MCU and sensors. In harvester unit, the RF signal is received from the antenna which is designed to have the highest gain at the frequency of standard RF signal transmitted through RFID reader. The voltage received by the antenna is amplified by multi-stage voltage doublers and the energy is stored in the capacitor. When the stored energy, and correspondingly voltage of the capacitor, reaches to some specific amount, the power management unit switches the circuit on so the the regulated voltage can power the MCU and sensors. When the voltage reaches to the minimum voltage needed by the MCU and the rest of the circuit, power management unit turns off the switch and all the processing and data gathering stops until next power cycle. There are multiple types of discharging used in this research that are decaying, processing, and waiting which are discharging while power source is not available before reaching to desired voltage, while running the task, and while the system is in low power mode respectively. Each type of discharging is also explained in more details in Section 4.

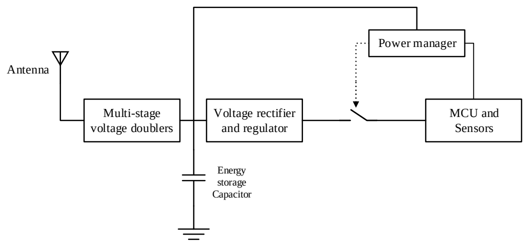

Figure 2.1: Block diagram of an RFID energy harvesting device

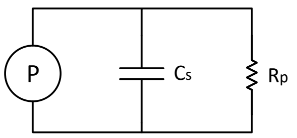

Figure 2.2: Equivalent charging circuit of the device

To find the voltage equations for the circuit, we assumed the energy source to be a fixed power instead of a fixed voltage or current source because what is received from the RFID reader is a fixed power. We also considered a parallel resistor R_p to the capacitor which consists of the equivalent storage capacitor and also the rest of the circuit's resistor in parallel with capacitor. Thus, the power harvesting circuit can be simplified to a fixed power source and a capacitor in parallel with a resistor which is shown in Fig. 2.2.

Therefore, voltage of the capacitor can be calculated by solving the following equation:

$$\frac{P}{V} = C_s \frac{dV}{dt} + \frac{V}{R_p} \tag{2.1}$$

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

$$V = \sqrt{PR_p - e^{\frac{-2t}{C_s R_p}} * \left(PR_p - V_0^2\right)} \tag{2.2}$$

Where V_0 is the voltage of capacitor at $t = 0$, and P is the power received from power source after going through all the voltage douber stages. Based on 2.2 time to reach from voltage V_0 to V, where $V > V_0$, can be calculated as:

$$t_{charging} = \frac{C_s R_p}{2} Ln \left(\frac{P R_p - V_0^2}{P R_p - V^2} \right) \tag{2.3}$$

2.2 Software characteristics

In real-time systems, in general, a task τ_i is a characterized by its worst case execution time C_i, released time r_i, absolute or relative deadline d_i or D_i respectively, and in case of being periodic its period T_i. In this research, we only consider periodic task and for multiple task considerations we assume the arrival or released time of all the task to be at time 0. We also consider the implicit deadline which means $D_i = T_i$. Accordingly, task i can be expressed as $\tau_i : (C_i, T_i)$.

In multi-task scheduling, we considered the the tasked to be scheduled in non-preemptive manner. Hence, higher priority tasks cannot preempt a lower priority task when the lower priority task is running on the device. due to the nature of intermittently-powered devices. This is due to the nature of intermittently powered devices and it is also common scheduling model in prior work such as [4, 21].

Chapter 3

Challenges

To explain the challenges of using an intermittently powered device in sensing applications, we consider a case study on WISP, an RFID programmable tag which is introduced in [17]. Based on [6] the power received by WISP can be calculated as:

$$P_r = \frac{G_s G_r \eta}{L_p} \left(\frac{\lambda}{4\pi(d+\beta)} \right)^2 P_t \tag{3.1}$$

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

Based on WISP specifications, the parameters would become as follows: We use a RFMAX S9028PCL polarized directional antenna which has the transmission gain of $G_s = 8dBi$. WISP has a linear dipole antenna therefore the reception gain is $G_r = 2dBi$. WISP works on 915MHz frequency so wavelength would be about $\lambda = 0.327$. Polarization Loss, noted as L_p; rectifier efficiency, noted as η; and distance compensation parameter, noted as β should be measured in practice. We use WISP values based on [6] which sets $\beta = 0.2316$, $\eta = 0.125$, and $L_p = 2$. Since all the parameters except d are constant I can rewrite the equation to:

$$P_r = \alpha \left(\frac{1}{d + \beta} \right)^2 P_t \tag{3.2}$$

Where α and β are the constant values that should be measure based on experiments. Given the variables mentioned above the equation would become:

$$P_r = 0.001354 \left(\frac{1}{d + 0.2316} \right)^2 P_t \tag{3.3}$$

As it can be inferred from 3.3 the power received by the tag is fixed when it is located in fixed distance to reader. By assuming the distance to be about $60cm$ and the power transmission of $1W$ power reception would be about $1mW$.

We consider that the power reception circuit follows as Fig. 2.2 with $R_p = 1G\Omega$, and $C_s = 100\mu F$.

Fig. 3.1 shows the result of the tasks running on this case study when the tag is always getting charged by the reader. In this figure, red lines are charging cycles and blue lines are the discharging or processing cycles. The power on threshold which is defined by the hardware is 2.2V and power off threshold is set to 1.8V which is the minimum voltage required by MCU.

One of the problems in using intermittently powered device for sensing applications is when the tasks needs to be done atomically and without any power disruptions. As it can be found from Fig. 3.1 the maximum execution of a task is about 40ms. Therefore, any task that needs more than 40ms atomic execution never can finish its job. As an example,

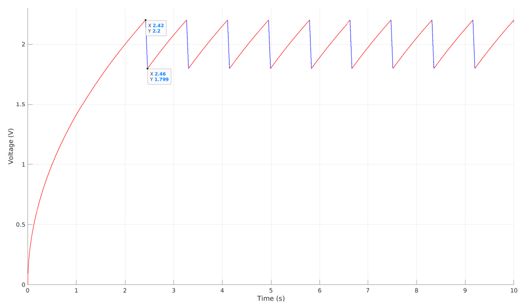

Figure 3.1: Charging and discharging cycles of a regular energy harvesting device

most of the gas sensors are designed to work on specific temperature, since cooling is more difficult, there is a micro-heater inside these sensors to keep the temperature on that specific temperature. These heaters need some time to provide enough heat to attain the desired temperature. As an example, a BME680 commercial gas sensor needs about 100ms to output a valid gas value. Any power loss during sensing would lead the failure of the job and there is not way to pause the sensing procedure. Therefore, these types of applications cannot run on the current intermittently powered devices.

Another problem that tasks running on battery-less devices suffer from is that they are unable to capture samples at specific times because the device can only sample data whenever it receives enough power from the power source. In the example shown in Fig. 3.1, the device can only capture samples during the time when the MCU is processing (blue lines). In many applications, capturing samples exactly at specific time is required to guarantee the validity of data and the process. For example, in the aforementioned case study, it is impossible for the device to capture samples every 8 seconds. Even though it wakes up multiple time during time from 0 to 8 seconds, any sample captured in those times is unusable and at the time of 8 second the device is not powered on to capture the sample which would lead to failure of the task.

Another challenge in using battery-less devices is time keeping. The device goes off after each charging and execution cycle and it loses the track of time it is working. Even if a real time clock exists on the device, it would also go off after each power loss. Therefore, any application that needs the notion of time would not be able to run in these type of devices. In this case, it is not possible to check data freshness when using the data which in many applications causes improper or even harmful reactions.

Chapter 4

Proposed Method

In this research, we define three types of discharging: decaying, processing, and waiting. we assume all the discharging rates to be linear which is also confirmed by the experimental results. Decaying is the time when the device is not receiving any power from the power source but the voltage threshold of the storage unit is below the minimum voltage needed for turning on the MCU. In this case, while the circuit in not ideally open circuit and also because of the parallel equivalent resistor exists in the capacitor, the device loses some amount of energy gradually. Processing is the time when voltage of the capacitor becomes above the minimum threshold, power management unit turns on the system, and MCU is running the task. We also added another mode where the power management unit turns on the system, however, since there is not enough energy in the capacitor to keep the MCU continuously running for the time required by the task to be finished, the MCU goes to low power mode. We named this time as waiting time when the MCU waiting for the system to store enough energy to be able to run the task continuously. We consider m_D, m_P, and

m_W to be decreasing slope of voltage while discharging during decaying, processing, and waiting respectively.

During the waiting time, the system is consuming some amount of energy but we consider the power reception from the power source to be always higher than the power consumption in waiting time which causes the voltage to increase in charging cycle. The whole purpose of adding the waiting time is to keep the voltage of the capacitor increasing even when the system is turned on by the power management. When the power management turns on the device based on the execution time of the task, the voltage required for the task to run continuously with the discharging rate of m_P is calculated. According to this voltage and the charging and discharging rates during waiting time, the time required for the system to reach that voltage is calculated. Afterward, a timer is initialized to turn on the MCU after that amount of time and then MCU goes to low power mode.

As mentioned previously, for simplicity of the analysis, all the discharging types are considered to decrease the voltage of the capacitor linearly with different slopes. Charging curve of the device also can be approximated to be linear by slope of m_c where m_c can be calculated based on Eq. (2.3) when V_0 is the minimum voltage threshold of the device and V is the maximum voltage that the capacitor can hold based on its specifications. Therefore, the charging slope can be calculated as:

$$m_c = \frac{V_{min\_th} - V_{max}}{\frac{C_s R_p}{2} Ln\left(\frac{PR_p - V_{min\_th}^2}{PR_p - V_{max}^2}\right)} \tag{4.1}$$

where C_c and T_c are charging time and charging period of each charging cycle respectively.

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

$$\Delta V = \frac{m_c * C_c - m_d * (T_c - C_c)}{T_c} * \Delta t \tag{4.2}$$

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

$$V = m_a * t - \sum_{i=1}^{n} \left(\left\lfloor \frac{t}{T_i} \right\rfloor + s_i \right) * C_i * m_{Pi} + V_0 \tag{4.3}$$

where m_{Pi} is the processing discharging rate for task i. s_i is 1 or 0 if the task i's last released job is serviced or not respectively. It should be noted that Eq. (4.3) is valid before or after each task execution and not while the task is running on the MCU. That is a reasonable criteria since voltage level calculation is needed only after each task execution and before putting the MCU to sleep mode or running another task.

4.1 Single Task Schedulability

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

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

$$\Delta t_w = \max \left\{ \frac{V - (V_{min\_th} + m_{P1} * C_1)}{m_a}, T_1 * \left\lceil \frac{t}{T_i} \right\rceil \right\} \tag{4.4}$$

where V is the current voltage of the capacitor that can be calculated from Eq. (4.3).

For a single task to be schedulable based on aforementioned criteria, first it is needed to check if the accumulation slope m_a is positive or not. Being the m_a positive, complies with the previous assumptions that the voltage level increases during each charging cycle. In this case, the task is guaranteed to be schedulable if the voltage can reach to the desired voltage for the task to run continuously during each period of the task which means the following condition should be met:

$$m_a * T_1 \geqslant m_{P1} * C_1 \tag{4.5}$$

There are multiple ways to implement waiting discharging. During waiting time, the device is still on but it is maintained in low power mode so that it can store more energy. It can be implemented on the MCU itself like the low power mode (LPM3) exists in famous MSP430 low power TI microcontrollers or it can be implemented by adding an external low-power real time clock (RTC) that can be programmed so that it can wake the MCU up by an interrupt after a specific time which can be calculated using Eq. (4.4). In either case, MCU would be kept at low power or sleep mode, however, the depth of the sleep depends on the implementation.

When the m_a is nonpositive the charging rate is not enough to keep the device on during each charging period thus it would cause the device to turn off before start of next charging period and there is no way to avoid that. In this case, we cannot keep track of the time and cannot used the aforementioned methods to find the schedulability of the task.

4.2 Multi-Task Scheduling and Analysis

In order to ensure the schedulability of multiple periodic tasks, we propose to abstract the resource demand of the tasks as a *periodic server* [18, 20, 19] and schedule them by using the budget of the server. The periodic server is characterized by the two parameters, budget and budget replenishment period, which correspond to the execution time and the period of a periodic task, respectively. Hence, once the server parameters are chosen, the schedulability of the server can be analyzed with Eqs. (4.4) and (4.5).

The scheduling of tasks within the periodic server in an intermittently-powered device should be done in a non-preemptive manner and is different from that in conventional real-time systems. Thus, existing hierarchical schedulability analysis for preemptive tasks in periodic servers [16, 11, 12, 8] are inapplicable to our problem. Instead, we formulate this as a variant of the bin-packing problem with additional constraints. The items to be packed are the jobs of tasks, and the amount of budget per period is the size of a bin. The number of bins in this problem is given by T_{gcd}/T_s, where T_{gcd} is the hyperperiod of all periodic tasks and T_s is the server period. Each task i has m items where m is the number of jobs arriving during the hyperperiod, i.e., T_{gcd}/T_i. An important thing here is that no more than one item (job) from the same task can be allocated to the same bin, and the items should be spaced across bins with respect to their task period. This add complexity to the original bin-packing problem which is already NP-hard. Hence, we plan to develop heuristic approaches for this problem.

Another approach that can also be pursued is to define a charging time for each task and define a pseudo-task for each charging that needs to be completed before each non-preemptive task. This can be done by assigning the priorities for pseudo-tasks and execution-tasks properly. Further scheduling analysis can also be expanded from regular preemptive scheduling methods with shared resources such as [1] to model non-preemptive tasks as well as preemptive charging pseudo-tasks.

Chapter 5

Implementation

In this section, we describe the hardware and software setup we use to simulate and implement the methods we discussed in previous sections.

We designed a RFID tag, R'tag, to do the experiment. It is designed based on WISP tag which is introduced in [17]. It features the same MSP430 MCU to do the processing, and the same RF circuit and antenna to receive power and send data to the RFID reader. In addition, we integrated the tag with extra external I/Os that can be used for ADC reading for analog and digital measurements. Furthermore, we added a large super capacitor that can be used to store more energy that enables the device to run for longer time.

For the sensing purpose, we designed a sensor board PCB that can be mounted on the R'tag and measure high resistance values that are generated by resistive sensors that are used for variety of applications. It is also equipped with BME680, a commercial integrated environmental sensor that can measure the temperature, pressure, humidity, and

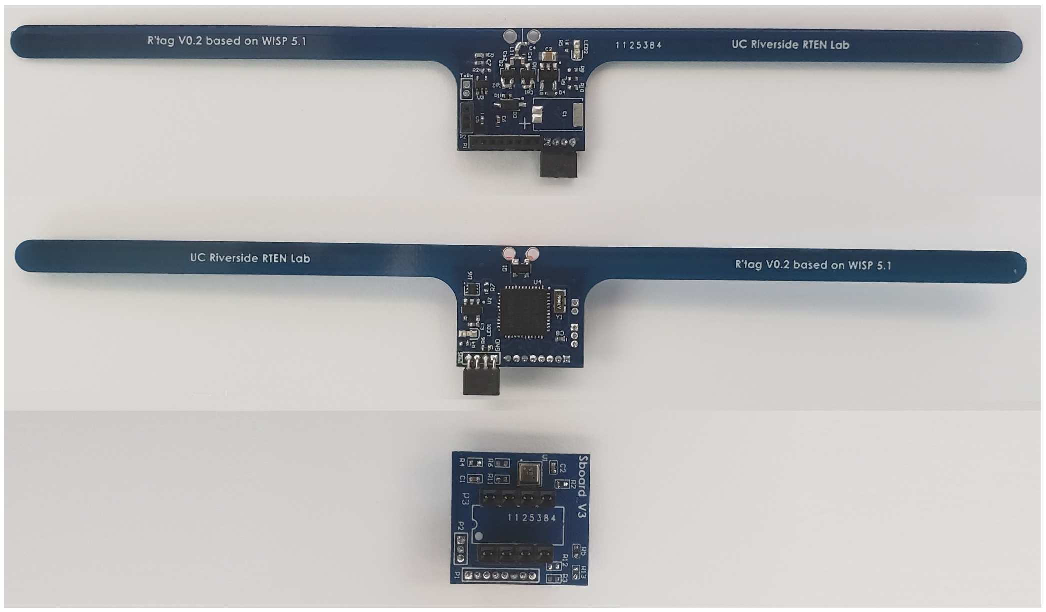

Figure 5.1: R'tag and Sensor board designed for experiment

total volatile organic compounds in the air. Both R'tag board and the designed sensor board are shown in Fig. 5.1

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

On the software side, we use MATLAB for the simulation purpose, Code Composer Studio as the IDE to compile the codes for MSP430 microcontroller, Altium Desginer too design the PCB boards, and Impinj Multitool to manage the RFID reader to charge the tag and send data.

Chapter 6

Evaluation

In this section, we describe the experimental results for one scenario that have been performed on our hardware platform to show that the proposed method works in real applications. We also provide simulation results to show the effect of different parameters based the designed platform and the proposed system model.

6.1 Experimental Results

Our experimental setup consists of the Impinj RFID reader which is connected to the PC via Ethernet cable, a R'tag that is located about 25cm away from the reader's antenna, our sensor board attached to the R'tag, and a $10M\Omega$ resistor connected to the sensor board as an example of a resistive sensor. A $47\mu F$ capacitor is used in the storage unit to store energy during each charging period. The voltage is measured by an Oscilloscope attached to the R'tag via some wired soldered to the R'tag. Fig. 6.1 show the Experimental setup that is used in this project.

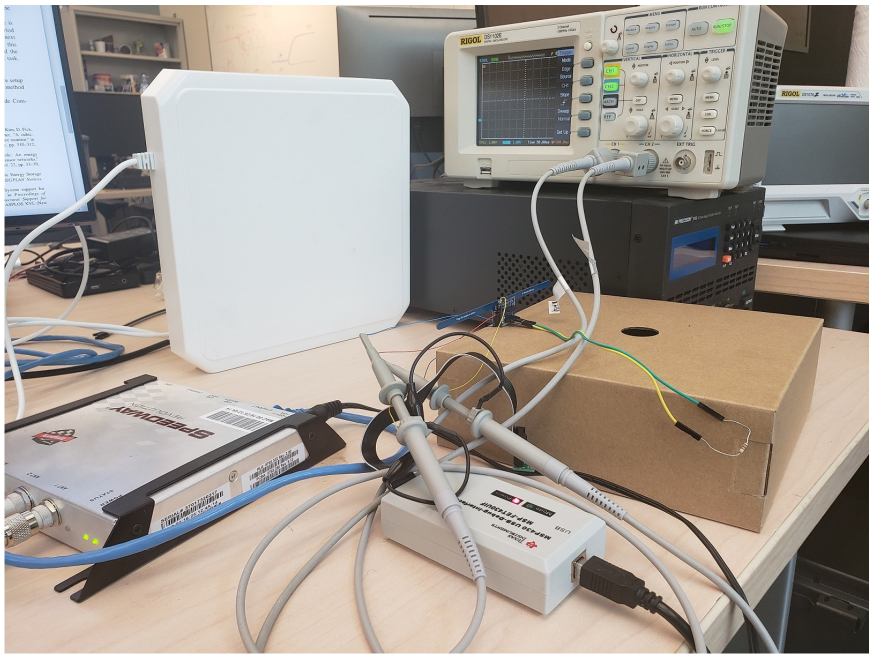

Figure 6.1: Experimental setup used for evaluation

One of the widespread kind of sensors are resistive sensors that their electrical resistance is a function of sensing value. In many of the these types of sensors, the resistance value is so high that cannot be measure by some regular methods like voltage dividers due to current leakage of the I/O and ADCs. For that purpose, we use a capacitor in parallel with the measuring resistance and by measuring the time that it takes to charge the capacitor and based on charging rate of the parallel RC circuit we find the resistance value. In our case, for a $10M\Omega$ it takes about 120ms to charge the capacitor and finish the measurement. We assume this measurement to be done every 1s.

Fig. 6.2 shows when running the resistance measurment task on with baseline approach. Where blue line is the voltage of the storage unit capacitor and yellow line is the voltage of the last regulator that powers the system. As it can be inferred from the figure, by using the baseline approach that allows MCU to run the task whenever the power is available, there is no way to finish such task that has the execution time of 120ms. Since the task is not resumable in the next power sequence, MCU start the task from the beginning in each power cycles and fails to finish the task.

To do the proposed approach we first need to find the charging and discharging rates. Based on the Fig. 6.2 the discharging rate of the task can be approximated as about 12V/s. To find m_c and m_d we put the task in sleep mode and find the charging and discharging rate of the capacitor. Fig. 6.3 shows the charging and discharging rate based on the charging cycles. Then Eq. (4.4) is used to find the next waiting time to run the task. We set one of the I/Os high during the measurement to see the timings of the running task on the tag on oscilloscope. Fig. 6.4 shows that the task can meets the deadline and the voltage never goes down bellow the threshold to turn off the device. Therefore, the device always keeps running, the task meets the deadlines, and the MCU can keep track of time.

6.2 Simulation Results

In this section we describe the effect of some of the parameter on the proposed system model. We also compare the baseline and proposed method based on the simulation. The parameters we change includes the capacitor size, distance from power reception, and execution and period of the task.

In one scenario, we use the same parameters that has been used for the experiments. Fig. 6.5 shows the result when baseline approach is used compared to the proposed

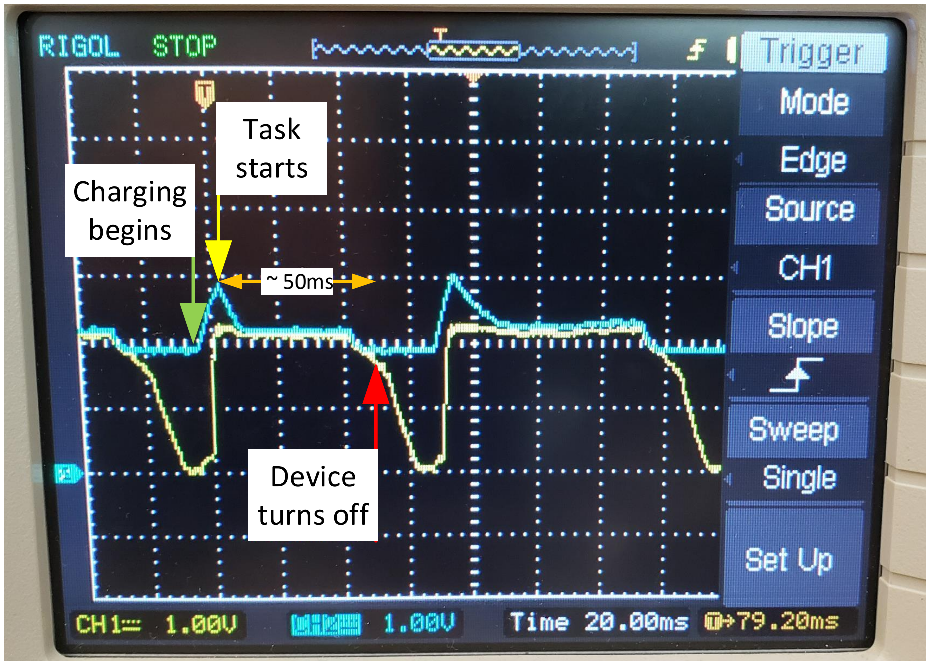

Figure 6.2: Running task fails on baseline approach. Blue line: capacitor's voltage, Yellow line: Regulator's output

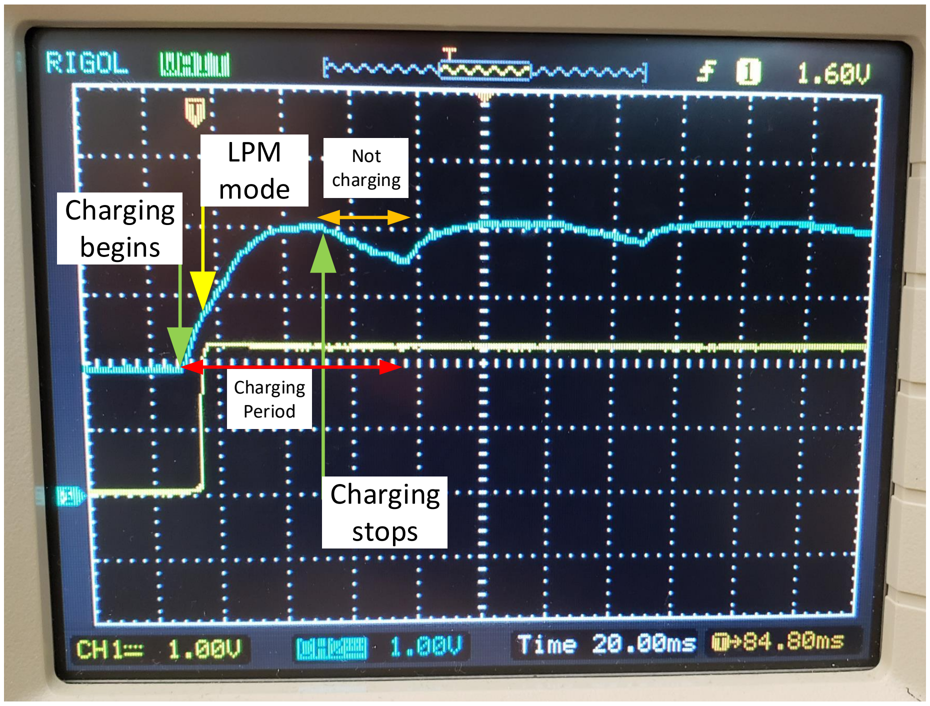

Figure 6.3: Charging curve of the capacitor when the MCU is on LPM3 mode. Blue line: capacitor's voltage, Yellow line: Regulator's output

method. Green lines show the arrival times of the task. As it is shown in the figure, baseline approach fails to run the task continuously for $120ms$ and since the task is not resumable it restarts in every power cycle. However, using the proposed method the task can finish its job before the deadline and is schedulable on the current platform.

In another scenario, we use different parameters for the task so that it can be finished even with baseline approach. Fig. 6.6 shows the result for task with $\tau : (25ms, 5s)$. Even in the baseline approach the task can finish its job in every power cycle. However, since it goes off in every power cycle it does the sensing over and over in each power cycle with cause redundant measurements. Furthermore, the action regarding the sensing cannot be done at the right time at $t = 5s$ which is the period of sensing and data freshness cannot be satisfied. On the other hand, by using the proposed method, the MCU can keep track of time since it is always on and also does the sensing at the right time which satisfies the data freshness and correctness of the sensed data.

In real applications, sudden power loss is sometimes unavoidable. By deploying a low power RTC like Ambiq Micro RTCs [14] and using the proposed method the device can still keep track of time for a long period even with some sudden power losses. Fig. 6.7 shows the same scenario as shown in Fig. 6.6 with a sudden power loss between 6s to 10.5s.

To find out the effect of the distance of the tag from RFID reader, we used the model proposed on [6] which is explained in Section 3. For voltage calculation we used the model shown in Fig. 2.2 and 2.2 with $R_p = 120M\Omega$, $C_s = 47\mu F$, and P calculated using

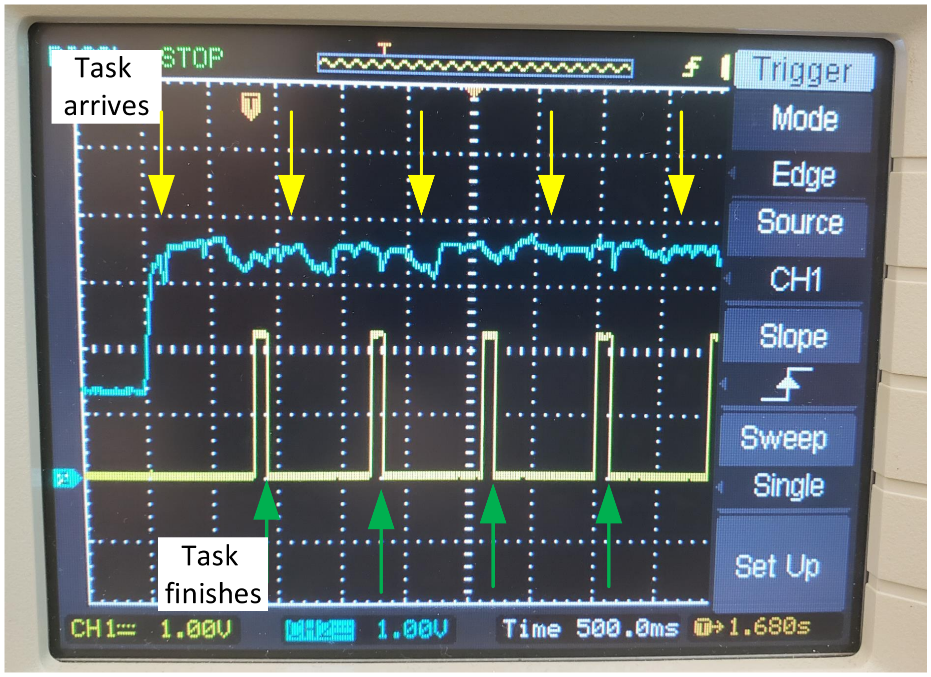

Figure 6.4: Charging curve of the capacitor when using the proposed methond. Blue line: capacitor's voltage, Yellow line: I/O indicating the running task

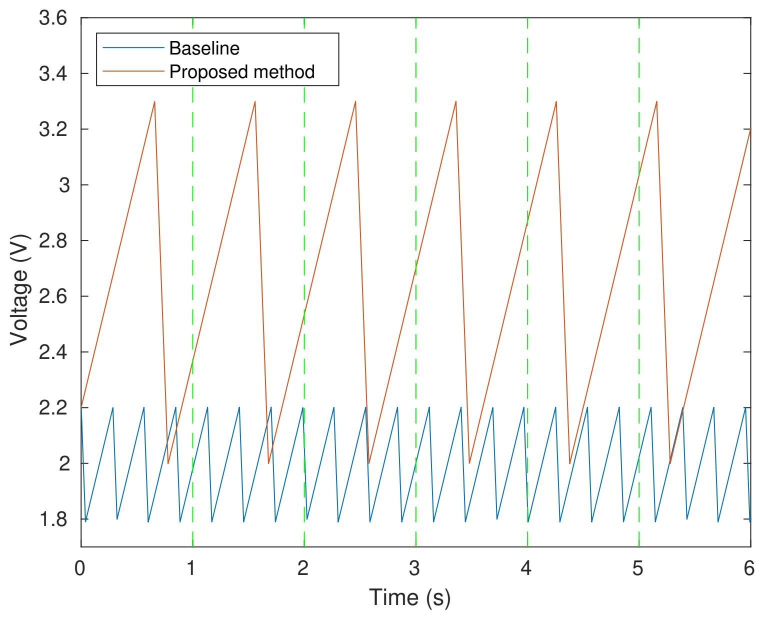

Figure 6.5: Charging and discharging behavior of the tag when running a task with τ : $(120ms, 1s)$

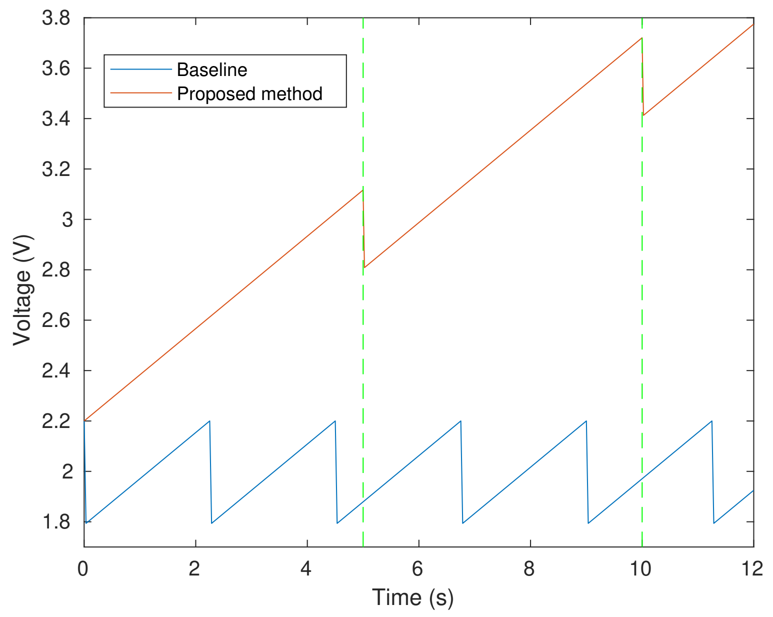

Figure 6.6: Charging and discharging behavior of the tag when running a task with $\tau : (25ms, 5s)$

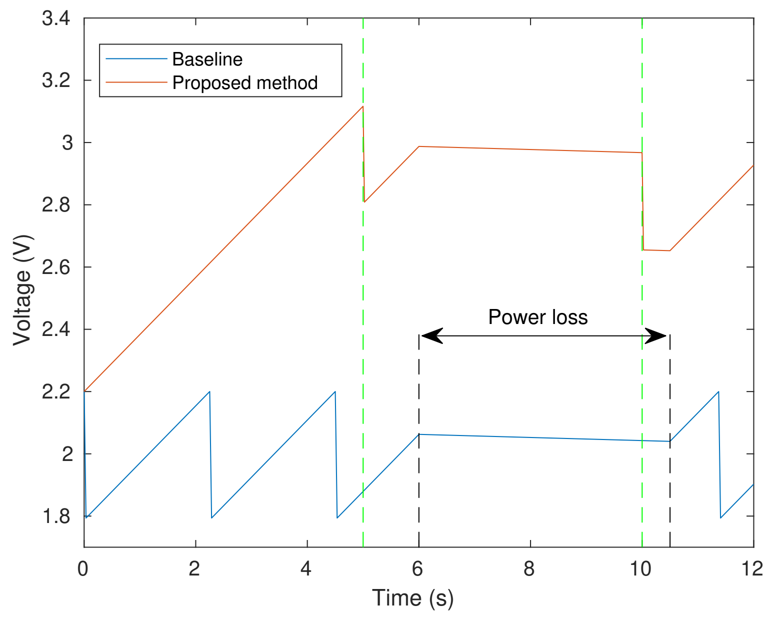

Figure 6.7: Charging and discharging behavior of the tag when running a task with $\tau : (25ms, 5s)$ in presence of power loss

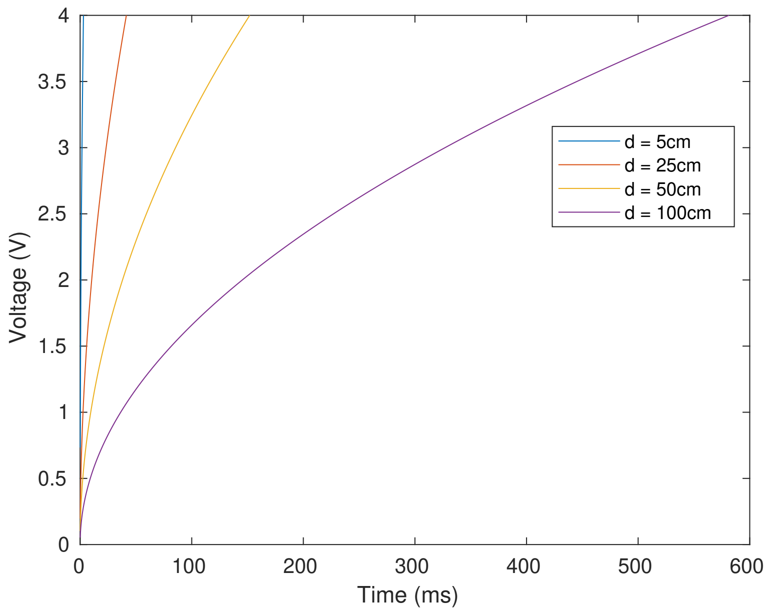

Figure 6.8: Power reception of the tag on different distance from RFID reader

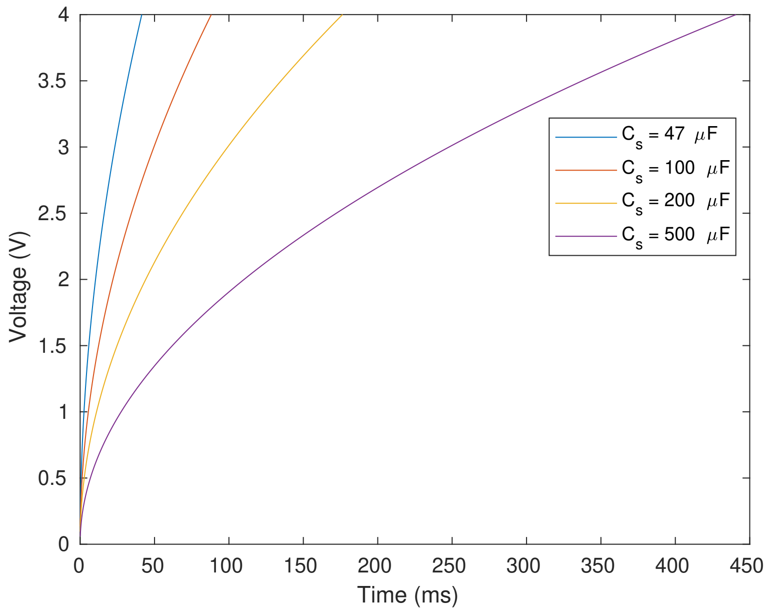

Figure 6.9: Power reception of the tag with different storage capacitors

3.3. As it can be inferred from 3.3, increasing the distance would lead to reduction in the power reception and respectively voltage increase rate. Fig. 6.8 shows this effect in multiple distances of the tag from RFID reader.

Another parameter that affects the charging rate is the storage capacitor size. The larger the capacitor, the longer it takes to charge and respectively the longer it can last. Fig. 6.9 shows the effect of capacitor size on charging rate.

Chapter 7

Conclusion

In this thesis, we provided an energy harvesting model that can be used to analyze the charging and discharging behavior of intermittently powered devices. This model was used to derive the voltage of the energy harvesting unit and schedulability analysis of a task on the device. To schedule the task on the device, overcharging and using the low power mode on the device was considered to store the required energy for the task to run continuously for a relatively large amount of time which was required by the task to satisfy its correctness and functionality of the system. Time keeping and periodic sampling was also addressed by keeping the device on using the same method and finding the right time to run the task by keeping the notion of time on the device.

R'tag board, a custom designed programmable RFID tag, and a sensor board was designed and used as a case study in the evaluation to show the functionality of the method provided in the thesis. We showed in the experimental results that with the same charging rate from RFID reader, the baseline method failed to finish the task that had a relatively long atomic execution time and lost the notion of time while by using the proposed method the device could schedule the task while keeping the notion of time.

We also provided the simulation to analyze the same scenario on simulation and also to see the effect of each parameter on the behavior of the device. Simulation results was also matched with the experimental results showing that the failure of baseline method and the success of the proposed method. Furthermore, we showed that even with some occasional power loss the device can keep track of time and execute the task at the right time.

The proposed method has multiple interesting direction for the future work. First, multi-task scheduling using server based approach can be done by designing the server based on the charging and discharging rate and also task set. Second, a chemiresistive nanosensor measurement is going to be used as real application for health monitoring. Some filtering should also be applied to filter the out of order fluctuations on the sensor value readings. Third, better antenna design and more efficient power harvesting circuit that can harvest energy from other sources than RFID as a backup energy source and also a long range and more efficient communication channel than RFID communication to ensure the success of transmitted data.

Bibliography

[1] Theodore P. Baker. Stack-based scheduling of realtime processes. *Real-Time Systems*, 3(1):67–99, 1991.

[2] Gregory Chen, Hassan Ghaed, Razi ul Haque, Michael Wieckowski, Yejoong Kim, Gyouho Kim, David Fick, Daeyeon Kim, Mingoo Seok, Kensall Wise, David Blaauw, and Dennis Sylvester. A cubic-millimeter energy-autonomous wireless intraocular pressure monitor. In *2011 IEEE International Solid-State Circuits Conference*. IEEE, feb 2011.

[3] Alexei Colin, Emily Ruppel, and Brandon Lucia. A Reconfigurable Energy Storage Architecture for Energy-harvesting Devices. *ACM SIGPLAN Notices*, 53(2):767–781, 2018.

[4] Zheng Dong, Yu Gu, Jiming Chen, Shaojie Tang, Tian He, and Cong Liu. Enabling Predictable Wireless Data Collection in Severe Energy Harvesting Environments. In *Proceedings - Real-Time Systems Symposium*, pages 157–166, 2017.

[5] D. Fan, L. Lopez Ruiz, J. Gong, and J. Lach. Ehdc: An energy harvesting modeling and profiling platform for body sensor networks. *IEEE Journal of Biomedical and Health Informatics*, 22(1):33–39, Jan 2018.

[6] Shibo He, Jiming Chen, Fachang Jiang, David K.Y. Yau, Guoliang Xing, and Youxian Sun. Energy provisioning in wireless rechargeable sensor networks. *IEEE Transactions on Mobile Computing*, 2013.

[7] Josiah Hester, Kevin Storer, and Jacob Sorber. Timely Execution on Intermittently Powered Batteryless Sensors. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems - SenSys '17*, pages 1–13, 2018.

[8] Seyedmehdi Hosseinimotlagh and Hyoseung Kim. Thermal-aware servers for real-time tasks on multi-core GPU-integrated embedded systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2019.

[9] Hrishikesh Jayakumar, Arnab Raha, and Vijay Raghunathan. QUICKRECALL: A low overhead HW/SW approach for enabling computations across power cycles in transiently powered computers. *Proceedings of the IEEE International Conference on VLSI Design*, pages 330–335, 2014.

[10] Mohsen Karimi and Hyosueng Kim. Energy scheduling for task execution on intermittently-powered devices. In *Embedded Operating Systems Workshop (EWiLi)*, 2019.

[11] Hyoseung Kim and Ragunathan Rajkumar. Real-time cache management for multi-core virtualization. In *International Conference on Embedded Software (EMSOFT)*, pages 1–10. IEEE, 2016.

[12] Hyoseung Kim, Shige Wang, and Ragunathan Rajkumar. vMPCP: A synchronization framework for multi-core virtual machines. In *IEEE Real-Time Systems Symposium (RTSS)*, 2014.

[13] Kiwan Maeng and Brandon Lucia. Adaptive dynamic checkpointing for safe efficient intermittent computing. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'18, pages 129–144, Berkeley, CA, USA, 2018. USENIX Association.

[14] Ambiq Micro. Ultra-low power rtcs. `https://ambiqmicro.com/rtc/`, June 2019.

[15] Benjamin Ransford, Jacob Sorber, and Kevin Fu. Mementos: System support for long-running computation on rfid-scale devices. In *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVI, pages 159–170, New York, NY, USA, 2011. ACM.

[16] Saowanee Saewong, Ragunathan Rajkumar, John P Lehoczky, and Mark H Klein. Analysis of hierarchical fixed-priority scheduling. In *ECRTS*, volume 2, page 173. Citeseer, 2002.

[17] Alanson P. Sample, Daniel J. Yeager, Pauline S. Powledge, Alexander V. Mamishev, and Joshua R. Smith. Design of an RFID-based battery-free programmable sensing platform. *IEEE Transactions on Instrumentation and Measurement*, 2008.

[18] Lui Sha, John Lehoczky, and Ragunathan Rajkumar. Solutions for some practical problems in prioritized preemptive scheduling. In *IEEE Real-Time Systems Symposium*. IEEE Computer Society Press, 1986.

[19] Brinkley Sprunt, Lui Sha, and John Lehoczky. Aperiodic task scheduling for hard-real-time systems. *Real-Time Systems*, 1(1):27–60, 1989.

[20] Jay K. Strosnider, John P. Lehoczky, and Lui Sha. The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments. *IEEE Transactions on Computers*, 44(1):73–91, 1995.

[21] Kasım Sinan Yıldırım, Amjad Yousef Majid, Dimitris Patoukas, Koen Schaper, Przemyslaw Pawelczak, and Josiah Hester. InK: Reactive Kernel for Tiny Batteryless Sensors. *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems - SenSys '18*, pages 41–53, 2018.