ICIC Express Letters
Part B: Applications
Volume **10**, Number **9**, September **2019**

ICIC International ©2019 ISSN 2185-2766
pp. **833–840**

ENERGY EFFICIENT SCHEDULING OF TASKS IN IOT ARCHITECTURE

Nasro Min-Allah

College of Computer Science and Information Technology
Imam Abdulrahman Bin Faisal University
P.O. Box 1982, Dammam, Saudi Arabia
nabdullatief@iau.edu.sa

Received February 2019; accepted April 2019

Abstract. *Recent trends in IoT advances pose new challenges to the research community from hardware and software perspectives. Among them, the vitals ones are that these devices should be lightweight, small in size, low memory footprint, running on Internet of Things based operating systems, and long lasting. On the other hand, these devices expected to process increasingly complex problems and hence need more computational power. However, computational power comes at the price of higher energy consumption which is a limiting factor in the context. In this paper, we model an energy efficient system that consists of sensors with no processing capabilities and transmit data at regular intervals to a computing device supported by discrete speed levels. We study the power efficiency of the embedded processing device that ensures all accepted tasks are processed accordingly by their respective deadlines. Our analysis shows that energy savings are significant when the overall system utilization is low.*
Keywords: IoT, Power efficient scheduling, Energy savings, Fixed priority systems, Real-time systems

1. **Introduction.** With increased penetration of Internet of Things (IoT) devices into all fabrics of our daily lives, a fraction of power saving contributes significantly to the reduction in overall energy consumption. As per IoT architecture, several technological layers starting from the sensors, to a middle layer computing device, to sophisticated servers exist. The processing power of these devices increases in aforementioned order as well as the power consumptions of such devices. IoT devices by nature are low power devices and minimizing power consumption of such systems is the key factor [1-3]. It is forecasted that by 2020 the number of IoT devices connected to the Internet is expected to be 50 billion while there were 25 billion devices in 2015 [4]. Today, these devices facilitate industrial automation, smart cities, environment sensing and eco systems, running on various data rates, and handling both types of real-time and non-real-time traffic. IoT has become the de-facto standards in smart homes, wearable, smart infrastructure, street lights, vehicles, and smart shoes, etc. These sensors make the bottom layer of the popular 3-layers structure as per IoT reference architecture [4,5] and are normally battery operated. Such sensors are mainly intended to perform a single specific task like monitoring infrastructure health or temperature of a produce, etc. For instance, IoT based sensors are used in range of domains ranging from monitoring vaccine temperature to food quality to infrastructure health and often installed at locations where power socket might not be an option. These devices are mission oriented or expected to operate for longer duration and hence are battery operated. In many applications that change very slowly such as humidity, sensors are used to transmit data after regular intervals and processed by a system that offer

DOI: 10.24507/icicelb.10.09.833

extra functionality in comparison such as embedded systems. Similarly, IoT data streams provide the opportunities for real-time analytics through high end servers.

A typical embedded system generally includes all of the memory and peripheral interfaces supported by a uni-processor system. Due to simple implementation, many advanced features such as file or memory management are not supported in such system due to memory and power constraints. An embedded control system can be used as a middle layer in IoT architecture but again continuous power supply at such places can also be a challenge. Thus, deploying a higher-level operating system on such devices is not an option. As an alternative, Linux can be tailored to run on such devices but even then cannot run on 8-16 bit MCUs [3]. Even 32-bit systems are unable to run Linux smoothly due to insufficient RAM for Linux kernel. On the other hand, these devices do support many real-time applications and might need java virtual machine, etc. and hence typically need 32-64 bits hardware. Consequently, such devices run on some flavor of a real-time operating system (RTOS). For instance, networked embedded systems require a 32-bit processor that can smoothly run RTOS [4,5].

On the one hand, due to ever increasing complexity of applications, these devices are expected to handle a larger number of tasks, while on the other hand, these devices need to operate for longer hours and hence trades performance for energy consumption. Considering these constraints and the rise of IoT sensors, embedded system designers focus on reducing power consumptions of the overall system. Today, latest microprocessors installed in such devices are capable of operating at various speed level to maintain a balance between energy and system responsiveness. An ideal embedded system should support continuous levels of speed levels but due to practical considerations, only a set of discrete levels is supported. Consequently, with minor adjustments, theoretical speed levels are mapped to the corresponding discrete speed level supported by the hardware. For real-time systems, a higher discrete speed is desirable from responsiveness perspectives, but even lower speed is equally good as long as task deadlines are respected.

With growing popularity of IoT, many embedded control systems follow cloud first architecture, where these device nodes perform data collection or respond to remote control commands and hence connected to the Internet (using Google's Cloud IoT Core, etc.), where complex analysis is made using cloud/fog or edge computing etc. In this study, primary focus is made on the embedded control system for reducing their energy consumptions. Our system supports real-time tasks where tasks have strict deadlines associated and hence task deadline must be respected under any possible scenario [6-8].

We apply real-time scheduling theory, model a system that operates at 9 discrete levels, and handle n number of independent period tasks where each task demands required CPU slots for task completion before its next invocation. Priority assigned to a task remains fixed throughout the operation of a given task set. For power efficiency, we use the formulation derived in [9-11] between power consumption and system speed. To guarantee the timing requirements of individual tasks, we apply real-time scheduling theory [6-8,12-24] for static priority system under rate-monotonic scheduling algorithm. The only difference in our work and previous solutions is that we consider the output of a sensor as a periodic event. We then extend an existing feasibility test to determine a suitable CPU speed such that the energy consumption of the embedded system is reduced while the deadlines set to be sensors are respected as well. We study the hypothetical scenario of a hardware system operating on nine speed levels and integrate with schedulability theory for real-time systems. The lowest speed of our system in 200 MHz needs 1.18 volts, tentatively. To lower the overall power consumptions of such systems, we establish an algorithm based on latest scheduling technique and dynamic voltage scaling which enables adjusting system speed on the fly and applicable to real-time systems [12,13].

Remaining part of this paper has been divided into 5 sections. In Section 2, we provide related work, while Section 3 describes the hardware model supporting discrete speed

levels as well the nature of the tasks to be processed. Details of the algorithm are sketched in Section 4. Experimental results are discussed in Section 5 and conclusions are drawn in Section 6.

2. **Related Work.** As per IoT reference architecture [1-3], the lowest layer comprises wireless sensor networks (WSNs) with an assortment of sensor/actuator nodes [3,4,26-29]. In plant automation scenario, sensors send data every few seconds for further processing. In many domains, sensors record various types of data (task) and transmit to the middle layer that process these tasks. For analysis purposes, the results of middle layers are communicated to the cloud/fog for further processing. Due to size and power constraints, these devices run on simple operating systems such as TinyOS, FreeRTOS, or LiteOS that provided limited functionality.

Attempts have been made in lowering the power consumption of micro-processors by exploiting the feature of adjusting system speed at run time. These applications include both non-real-time [25] and real-time systems [5-20]. Weiser et al. [25] used the approach of busy system to increase system speed and reduce it when the system is idle. The work in [25] is based on dividing time into intervals to set speed for upcoming interval based on recent CPU utilization. Though promising for non-real-time applications, recent usage of the systems cannot predict the actual performance requirements of the task and hence invalid for real-time systems. A voltage scaling technique was adopted for real-time systems under both static and dynamic priority systems in [9,21]. Later, authors in [18] extended the work for multi-core systems.

In [1], energy efficient scheduling for IoT devices was discussed in detail and exchanges of local information between neighboring nodes as well as global server that maintain the overall network state were highlighted in [2]. In this paper, we model sensors as periodic task where each sensor (task) sends data to be processed by an embedded system that has the ability to adjust system speed in nine discrete levels. Our solution ensures the each task (sensor data) is processed before the same sensor resends another set of data for processing. The classic real-time scheduling theory has been applied while assigning priorities to individual tasks.

3. **System Model.** Currently, one of the most effective means for reducing power consumption of CMOS technologies in computing systems is dynamic voltage scaling (DVS). The relation between voltage and frequency offers the foundation for DVS. Energy of CMOS circuitry is expressed as [9-12]:

$$E = P \times t \tag{1}$$

where E denotes energy consumption, and P is the average power for running the processor for t units of time. Due to space limitation, we skip some relevant details but reader may see [8] for better understandings. The dominant capacitive switching applicable in this context can be written as follows:

$$P = \gamma \lambda V^2 F \tag{2}$$

where transition activity dependent parameter is represented by γ, switched capacitance is denoted by λ, V is the supply voltage, while F denotes the system speed. It is worth noting that V is directly proportional to F and (1) and (2) show that lowering system speed results in significant energy reduction. Equation (2) provides the foundation for DVS [9]. Assuming $g(F(t))$ is a function of F, the power consumption P of a processor running at speed F in interval $[t_0, t_1]$ can also be expressed as:

$$P = \int_{t_0}^{t_1} g(F(t))dt \tag{3}$$

An ideally processor should be able to run at continuous speed levels according to (3) but it is impractical due to fundamental physics constraints. Latest processors only operate at a few discrete speed levels. In this work, we assume our system supports nine discrete speed level where 1.0 is the maximum, and 0.2 is the lowest speed with a step size on 0.1.

We now introduce out task model executed by the embedded system. The system supporting handling data from n number of IoT devices and each IoT device is a task that generates the request for CPU slot after fixed intervals. We model this task as a hard periodic task τ_i. In worst case, all tasks arrive simultaneously, and each task presents e_i units of system processor time which is the worst case computation demand. All tasks are independent and pre-emptible while task deadline d_i is equal to task period p_i. This assumption ensures that each task receives its required time slots before another job of the same task reappears. The scheduling algorithm used is preemptive and assigns task priorities based on tasks periods where highest priority task has the smallest task period.

We use the model presented in Table 1 of Transemmta's Cursoe processor extracted from [5,29,30] which supports 10 discrete levels. We extrapolate the first row of Table 1 from the data given in [5,29,30] to keep the system up and the minimum speed to entertain critical systems tasks/interrupts is 200 MHz. For each voltage level, there is a corresponding CPU speed, i.e., system speed and supply voltage are almost directly proportional to each other and faster system speed is obtained at the price of higher operating voltage. In other words, a task that takes 10 ms while executing by the system at speed 500 MHz will just take 5 ms when run of 1 GHz. However, there is a price associated with higher speed in terms of power consumption, i.e., the system roughly consumption 8 times more are 1 GHz than running at 500 MHz.

TABLE 1. Characteristics of Transemmta's Cursoe processor [5,29,30]

Voltage	Frequency	Power
1.18	200	1.20
1.20	300	1.30
1.23	400	1.80
1.35	500	2.73
1.53	600	4.21
1.75	700	6.43
2.00	800	9.60
2.35	900	14.91
2.80	1000	23.52

4. **Power Efficient Schduling.** For scheduling the task set, we use RM algorithm and for schedulability analysis we follow exact condition. This approach results in higher system utilization as long as the task set is schedulable on the system. In addition, we incorporate the power efficiency component into feasibility analysis as discussed in [12] for reducing the overall energy consumption. We now use the model developed in Section 3 and integrate with real-time scheduling theory for enabling a system to run on the most appropriate frequency so that the deadlines of the tasks are never compromised. To maintain the critical components such as timer up and keep system active, the CPU runs at 200 MHz. To schedule n periodic task on a single processors system, rate monotonic scheduling algorithm is the optimal policy [3]. Each task is assigned a unique priority by rate monotonic scheduling algorithm which does not change at all on run time. Checking schedulability of a task is subject to feasibility analysis. A task τ_i is schedule on a single-CPU system when:

$$\min_{t \in S_i} \frac{w_i(t)}{t} \leq 1 \tag{4}$$

where $w_i(t)$ is collective system demand of a task τ_i at time t, and S_i is the set of scheduling points for τ_i that depends on the task period of all higher priority tasks and task of τ_i itself. A task is schedulable if and only if (4) holds.

Recently authors in [8] introduced a feasibility test that test checks task feasibility by starting feasibility analysis with largest point in set S_i for task τ_i.

Theorem 4.1. [8]: *A real-time system consisting of n periodic tasks* $\{\tau_1, \ldots, \tau_n\}$, τ_i *can be feasibly scheduled on uni-processor system for all tasks phrasings under rate monotonic scheduling algorithm iff:*

$$\max_{1 \leq i \leq n} \left\{ \min_{t \in O_i} \frac{w_i(t)}{t} \right\} \leq 1 \tag{5}$$

Here O_i is the set of scheduling points ordered in descending order in contrast to S_i. Static speed is possible to be obtained in first place for the task set before deploying the task set using sufficient condition for RM schedulability [12] but such arrangements can possibly run the system at higher speed. The system speed can be further lowered at run time by utilizing the slack time due to early completion of a task [12] but considerable CPU time can be wasted on schedulability analysis instead of processing the actual computation demands of tasks. It is worth mentioning that many scheduling algorithms have been used in real-time system, but we use the ones proposed in [8] for being an efficient algorithm. Inequality (5) determines task schedulability but does not help lower the speed. For calculating desirable system speed, the approaches derived in [18] are applicable in this context.

Our algorithm considers the system speed as discussed in [18,23,30] but extends Theorem 4.1 for lowering the system speed. Based on our system model, speed F_i appropriate for a task τ_i is extracted from a range of nine speed level, i.e., $0.2 \leq F_i \leq 1$. We assume that the value of F_i is rounded to a single decimal number so that it matches the underlying hardware supported speed levels. In our system on n tasks, a periodic task τ_i can be feasibly scheduled for all tasks phrasings using rate monotonic scheduling algorithm *iff*:

$$\max_{1 \leq i \leq n} \left\{ \min_{t \in O_i} \frac{w_i(t)}{t} \right\} \leq \frac{F_i}{F_m} \tag{6}$$

where $F_m = 1$, i.e., maximum system speed is 1.0 and hence limits the utilization by 100%. To obtain the desired speed F_i, system schedulability of an individual task is tested at all scheduling points one by one in set O_i and the first point that answers the schedulability of the task concludes the test with suitable theoretical speed. Inequality (6) becomes exactly like Inequality (5) when running at maximum speed of 1.0. We use the floor function on F_i to make it appropriate for an actual speed level supported by the hardware. This mechanism has an associated disadvantage that the system energy consumption might increase but at the same time, ensures the timing constraints for the task deadline, which cannot be compromised in hard real-time system and hence the focus of this work.

5. **Experimental Results.** In this section, we discuss our experimental results for the task and system model discussed in Section 3. We generated task sets of different sizes and each value plotted is obtained after 400 iterations. The microprocessor model used for analysis purposed is sketched in Table 1. Run the system at various utilization in range of [69% to 99%]. We generated synthetic task set of size 2, 4, ..., 20 where each task is a sensor data. Figure 1 shows the energy consumption of the system for n tasks when no DVS scheme is used versus a system using DVS scheme. The utilization of task set varies from 69% to 99%. To derive values for task periods, we used uniform distribution

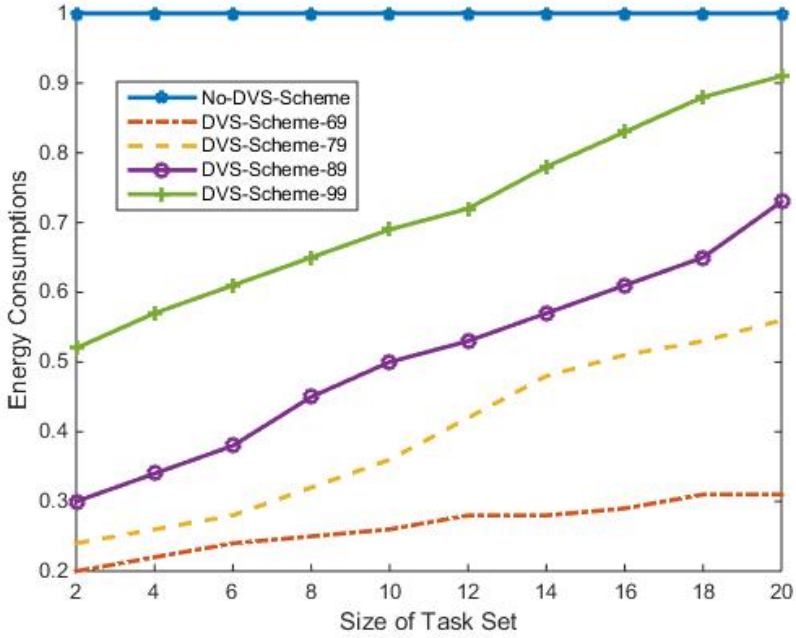

FIGURE 1. Normalized energy savings under varied system utilizations

and the same distribution was also used for extracting task computation times. For each task set, we run our simulation in Matlab while kept task priorities static. Computation cost of (4) against (5) is available in [8] and similar studies are made in [12,18,23]. In our experiment, the worst case execution times for tasks have variation of 5-10%, applied randomly. We represent (5) by No-DVS-Scheme and (6) with DVS-Scheme-U, where U is system utilization. In Figure 1, No-DVS-Scheme consumes 100% energy as the system is running at maximum speed in absence of DVS. It can be seen in Figure 1 that energy saving is higher for the smaller task set and increases as the task set grows. We show normalized values in Figure 1. First, we kept system utilization as low as 69% in Figure 1 that is shown by DVS-Scheme-69. With such utilization, this behavior in Figure 1 is due to the fact that the system is preemptive, and the workload is 69% which can be statistically scheduled for all tasks successfully. With increased task set size, more tasks were in competition for the CPU slots and to respect the deadlines. Due to this scenario, the system had to increase the operating speed. The increase in speed directly influenced higher power consumptions. Though at higher speed, a task now takes less time but since the role of voltage is quadratic, more energy was consumed.

The effect of utilization also affects the energy consumption as when system is heavily utilized, the opportunities for running the system at lower speed are rare. It can be seen from Figure 1 that under the same task set size, the corresponding energy reduction becomes more insignificant with higher utilization for DVS-Scheme. We changed the system utilization from 69% to 100% in Figure 1, with an increase of 10%, respectively. In Figure 1, the energy consumption with DVS-Scheme-69 is low as the system schedulability is determined faster by analyzing only a few points in the set of scheduling points and due to presence of slack, the CPU speed is minimum. It is worth noting that we plotted only the energy consumptions of those task sets that were declared schedulable according to rate monotonic scheduling while rest of the infeasible task sets were discarded. The energy saving decreased in DVS-Scheme-79, DVS-Scheme-89 and DVS-Scheme-99, where DVS-Scheme-99 was the worst case. DVS-Scheme-99 represents the values for the task set demanding 99% system utilization and DVS-Scheme has to check the schedulability of each task with (6) in descending order of points. This situation is understandable as

the system is fully utilized and the opportunities to lower the system speed levels are minimum and hence energy savings are insignificant.

6. **Conclusion.** Power consumptions of embedded systems capable of handling sensors data with a single processor system were discussed. The hardware studied in the work supported 9 discrete levels and a periodic task set was assumed that had fixed priority. Rate monotonic scheduling policy was used for scheduling tasks on the processor. Sensors were modeled as periodic tasks that generate computation workload after regular intervals. The computation demand was then transmitted to an embedded system that processes the data accordingly. It was noted that the energy gains were promising when system was underutilized and becomes insignificant when system utilization is very high (99%). The study was limited to middle layer only and as future work, we are intended to study energy aware scheduling in cloud environment when data is partially offloaded to high-end servers for further processing.

REFERENCES

[1] A. Hammadi and L. Mhamdi, Review: A survey on architectures and energy efficiency in data center networks, *Computer Communications*, vol.40, pp.1-21, 2014.

[2] S. R. Sarangi, S. Goel and B. Singh, Energy efficient scheduling in IoT networks, *SAC 2018*, 2018.

[3] Micrium, Why an RTOS for the IoT device?, https://www.micrium.com/iot/iot-rtos/, 2019.

[4] P. Sethi and S. R. Sarangi, Internet of Things, *Journal of Electrical and Computer Engineering*, p.6, 2017.

[5] D. Underwood, *The Relationship between Real-Time Computing and the Internet of Things (IoT)*, https://kingstar.com/relationship-real-time-computing-internet-things-iot/, 2017.

[6] C. L. Liu and J. W. Layland, Scheduling algorithms for multiprogramming in a hard real-time environment, *Journal of the ACM*, vol.20, no.1, pp.40-61, 1973.

[7] J. P. Lehoczky, L. Sha and Y. Ding, The rate monotonic scheduling algorithm: Exact characterization and average case behavior, *Proc. of the IEEE Real-Time System Symposium*, pp.166-171, 1989.

[8] N. Min-Allah, Effect of ordered set on feasibility analysis of static priority system, *The Journal of Supercomputing*, vol.75, no.1, pp.475-487, 2019.

[9] A. P. Chandrakasan, S. Sheng and R. W. Brodersen, Low power CMOS digital design, *IEEE Journal of Solid State Circuits*, pp.472-484, 1992.

[10] T. D. Burd, T. A. Pering, A. J. Stratakos and R. W. Rodersen, A dynamic voltage scaled microprocessor system, *IEEE Journal of Solid-State Circuits*, vol.35, no.11, 2000.

[11] D. Shin and J. Kim, Dynamic voltage scaling of mixed task sets in priority-driven systems, *IEEE Trans. CAD of Integrated Circuits and Systems*, vol.25, no.3, pp.438-453, 2006.

[12] P. Pillai and K. G. Shin, Real-time dynamic voltage scaling for low-power embedded operating systems, *Proc. of ACM Symp. on Operating Systems Principles*, pp.89-102, 2001.

[13] H. Aydin and Q. Yang, Energy-responsiveness tradeoffs for real-time systems with mixed workload, *Proc. of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium*, pp.74-83, 2004.

[14] T. Gloker and H. Meyr, *Design of Energy-Efficient Application-Specific Instruction Set Processors*, Kluwer Academic Publisher, Dordrecht, 2004.

[15] S. Borkar, Design challenges of technology scaling, *IEEE Micro*, vol.19, no.4, pp.23-29, 1999.

[16] M. B. Qureshi, M. A. Alqahtani and N. Min-Allah, Grid resource allocation for real-time data-intensive tasks, *IEEE Access*, vol.5, pp.22724-22734, 2017.

[17] E. Bini, G. C. Buttazzo and G. Buttazzo, A hyperplanes bound for the rate monotonic algorithm, *Proc. of the 13th Euromicro Conference on Real-Time Systems*, pp.59-67, 2001.

[18] N. Min-Allah, H. Hussain, S. U. Khan and A. Y. Zomaya, Power efficient rate monotonic scheduling for multi-core systems, *Journal of Parallel and Distributed Computing*, vol.72, no.1, pp.48-57, 2012.

[19] A. Burns, R. I. Davis, P. Wang and F. Zhang, Partitioned EDF scheduling for multiprocessors using a C=D scheme, *Real-Time Systems*, vol.48, no.1, pp.3-33, 2012.

[20] N. C. Audsley, A. Burns, R. I. Davis, K. W. Tindell and A. J. Wellings, Real-time system scheduling, in *Predictably Dependable Computing Systems*, B. Randell, J.-C. Laprie, H. Kopetz and B. Littlewood (eds.), ESPRIT Basic Research Series, Springer, 1995.

[21] N. Guan, Q. Deng, Z. Gu, W. Xu and G. Yu, Schedulability analysis of preemptive and non-preemptive EDF on partial runtime-reconfigurable FPGAs, *ACM Trans. Design Automation of Electronic Systems*, vol.13, no.4, 2008.

[22] Y. Lyu, L. Chen, C. Zhang, D. Qu, N. Min-Allah and Y. Wang, An interleaved depth-first search method for the linear optimization problem with disjunctive constraints, *Journal of Global Optimization*, vol.70, no.4, pp.737-756, 2018.

[23] N. Min-Allah, A.-R. Kazmi, I. Ali, J. Xing and Y. Wang, Minimizing response time implication in DVS scheduling for low power embedded systems, *Innovations in Information Technologies*, 2007.

[24] R. Kravets and P. Krishnan, Power management techniques for mobile communication, *Proc. of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM-98)*, New York, pp.157-168, 1998.

[25] M. Weiser, B. Welch, A. Demers and S. Shenker, Scheduling for reduced CPU energy, *Proc. of the 1st Symposium on Operating Systems Design and Implementation (OSDI)*, Monterey, CA, pp.13-23, 1994.

[26] J. Jin, J. Gubbi, S. Marusic and M. Palaniswami, An information framework for creating a smart city through internet of things, *IEEE Internet of Things Journal*, vol.1, no.2, pp.112-121, 2014.

[27] D. Evans, *The Internet of Things How the Next Evolution of the Internet Is Changing Everything*, https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf, 2019.

[28] D. Ghrab, I. Jemili, A. Belghith et al., Study of context-awareness efficiency applied to duty cycled wireless sensor networks, *IEEE Wireless Communications and Networking Conference*, pp.1-6, 2016.

[29] A. Klaiber, *The Technology Behind Crusoe^{TM} Processors*, Transmeta Corporation, https://web.stanford.edu/class/cs343/resources/crusoe.pdf, 2019.

[30] S. Alrashed, Reducing power consumption of non-preemptive real-time systems, *The Journal of Supercomputing*, vol.73, no.12, pp.5402-5413, 2017.