DOI. 10.1109/SACI.2016.7507362

TEEARTS: Time and Energy Efficiency Analysis for Real Time Systems Framework

Cristina S. Stangaciu, Valentin Stangaciu, Mihai V. Micea, Andrei Stoica,Vladimir I. Cretu
Politehnica University, Timisoara, Romania
Computer and Software Engineering Department
2, Vasile Parvan Blvd., 300223, Timisoara, Romania
certejan@dsplabs.cs.upt.ro, valys@dsplabs.cs.upt.ro, mihai.micea@cs.upt.ro, andreiz14@gmail.com,
vladimir.cretu@cs.upt.ro

***Abstract*—In this paper we introduce TEEARTS (*Time and Energy Efficiency Analysis for Real Time Systems*) Framework. TEEARTS is being developed as an energy consumption measurement and estimation framework for embedded systems, with applicability especially in real-time systems. Beside the measurement modules, an energy estimation model is proposed, along with a methodology to determine its parameters. A set of experimental results regarding the use of this methodology and the energy consumption analysis framework on an embedded target are also presented and discussed.**

I. INTRODUCTION

The main factors which influence the energy consumption of an embedded system are, on one hand, the hardware, with its particularities and on the other hand, the energy requirements of the applications which run on the system and the energy requirements of the operating system[1].

This variation of the energy consumed by a system in different scenarios has been deeply studied in the last decade, mostly due to the evolution of the hardware which has the capability to adjust its power consumption depending on the application demands. These studies led, among others, to the definition of the so called "*power-aware*" systems, as systems that "*scale their power consumption with changes in their operating scenario with a view to maximizing their energy efficiency*" [2].

In a more restrictive sense, we refer to those systems capable of programmatically scaling their power consumption according to changes in their operating scenario (e.g. systems capable of dynamic power management or dynamic power/frequency scaling).

We can say that the energy consumed by a running system is given by two main types of factors: power consumption factors (which have a direct influence on the energy consumed by the system) and temporal factors (which have an indirect influence).

Each of the two parameter sets need to be determined, specified and modeled as a measurable and controllable parameter set, as part of a consumption model. This model can be further used in order to make an estimation about the energy consumed by the targeted system on different scenarios.

Starting from classic task models for real-time systems, regarding the temporal parameters such as period, deadline, worst case execution time, etc., new task models have been derived, which include also power consumption/energy parameters specific to each task execution scenario. A brief overview of these models can be found in [3].

In order to be able to predict the energy consumed by a running system for a given scenario, both the temporal and the power consumption parameters must be determined.

While there is significant work presented in the literature, regarding the measurement and evaluation of the temporal parameters (e.g. worst case execution time, best case execution time, etc.), there are much fewer examples of the measurement and evaluation of the energy consumption parameters.

In this paper we propose a framework for measuring and estimating the energy consumed by a running system for a given period, a consumption model for a class of real-time embedded systems, a methodology to determine the parameters included in the model and a set of measurement result and interpretations.

This work is structured as follows: in the second section other notable approaches will be presented, in the third section the proposed TEEARTS Framework is described in terms of architecture, model and methodology. Section IV presents the conducted experiments and a set of measurement results. The article ends with a section of conclusions.

II. RELATED WORK

The measurement and estimation for the consumption of a running embedded system has been studied at different levels: instruction level, application level and operating system level on one hand and at different hardware components on the other hand.

One of the first methods analyzed for power consumption and profiling was at instruction level [4]. Real world data showed that there is a measurable difference [5] in the power consumption of the different sets of instructions (arithmetic, multiplication, load & store), but this difference and overall consumption per instruction was overshadowed by the power consumption of other components like the cache, decoding logic, etc. in modern integrated circuits. This method is an inefficient profiling method for use with today's processors.

Another method of evaluating the power consumption at a higher level, is implemented for entire blocks of code or algorithms. Normally a shorter execution time means less energy draw, but there are exceptions [6]. It was proven [7] that for sorting algorithms there is no direct correlation between the execution time (and complexity) and the energy consumption being measured. Profiling the energy consumption at program or algorithm level is a

more efficient method to use when we make our design decisions for the application being built [8].

There are also a series of papers which analyze the energy consumption of different operating systems (e.g. Linux, and a number of its services is analyzed in [9], a series of four microkernels: uC/OS-II, ECHIDNA, NOS and LIMIT are analyzed in [10] and Android is analyzed in [11]). Still, the number of the systems analyzed is relatively small, compared to the number of the operating systems analyzed in terms of performance in general, where the energy consumption factor is missing.

III. TEEARTS FRAMEWORK

TEEARTS is a measurement and analysis framework proposed for specification, analysis and estimation of the time and energy behavior of a real-time embedded system.

The proposed architecture is presented at the block level in Figure 1. The Framework is developed around a time-energy behavior model of the periodic tasks, which is described in the next section. The framework contains also measurement modules for both the energy and the time parameters.

The considered targeted systems are real-time embedded systems with power-aware capabilities.

A. Task Model

The proposed task model was developed starting from two different directions: the target system model and the application model, thus it contains both temporal and power/energy consumption parameters:

$$\theta_i \equiv \{T_i, C_i, D_i, \varphi_i, \Psi_i\} \quad (1)$$

where, θ_i is the task i of the set, defined by the parameters (T_i – task period, C_i – worst case execution time, D_i – the relative deadline of task i, φ_i – the start time or phase, Ψ_i– a vector containing the devices required by the execution of task i).

Regarding the temporal parameters, they must be specified using an absolute reference. Either they are specified in subunits of seconds or in clock cycles of a fixed (nominal) computing frequency. We must consider the fact that the task computation time is affected by the change of the actual computation frequency, but the relative deadline and period is unaffected by this change.

The power/energy consumption behavior is determined firstly by the set of the powered devices used by each task:

$$\Psi \equiv \{\psi_0, ..., \psi_j, \psi_{j+1}, ..., \psi_m\} \quad (2)$$

where, ψ_j represents a device used by the system.

Each device is characterized by a state (running scenario), for which the power must be determined.

A device state is specified as following:

$$S_j^k = \{P_j^k\} \quad (3)$$

where S_j^k represents the k state of the device j, P_j^k represents the power of the device j in state k.

According to the energy consumption formula, the energy consumed by a device j in the k state, during the execution time (C_i) of a task θ_i is equal to:

$$E_{i,j,k}(0, C_i) = \int_0^{C_i} P_j^k(t)dt \quad (4)$$

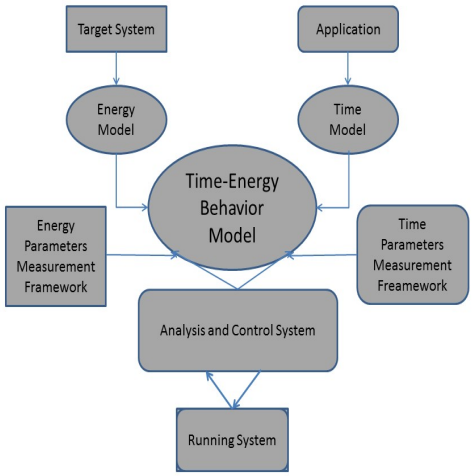

Figure 1. TEEARTS Framework

If the power function is constant, then the equation becomes:

$$E_{i,j,k} = P_i^k \cdot C_i \quad (5)$$

In this paper, we consider systems which run periodical tasks and we try to determine the energy consumed during an energy consumption period of the task set. For some particular cases the energy consumption period can be the hyper period of the task set. This is true if each job of a task has the same energy consumption. The hyper period is equal to the least common multiplier (LCM) of the task periods:

$$H_\theta = LCM(T_i, i = 1, n) \quad (6)$$

If a real-time operating system (RTOS) is considered, to the energy consumption due to the application, we add the RTOS energy consumption during a cycle (E_{RTOS}). The total energy calculated on an energy consumption period becomes:

$$E_{tot} = E_{APP} + E_{RTOS} \quad (7)$$

where, E_{APP} represents the energy consumed by the application during an energy consumption cycle.

The energy consumed by the RTOS in a cycle is application dependent and can be divided into other components:

$$E_{RTOS} = N_{\Pr eempt} E_{\Pr eempt} + N_{Sched} E_{Sched} + E_{Other} \quad (8)$$

where, $N_{Preempt}$ is the (maximum) number of task preemptions which occur in an application cycle; $E_{Preempt}$ is the energy consumption of a task preemption; N_{Scbed} is the number of the scheduler instances in the cycle; E_{Sched} is the energy consumed by the scheduler; E_{Other} is the energy overhead due to the execution of the other operating system tasks/processes.

The energy consumed by the application (the task set) can expressed as a sum of the energy consumed by each task and an overhead introduced by the state transitions:

$$E_{tot} = \sum_{j=0}^{m} P_j * t_{cycle} + \eta_{Switch} \quad (9)$$

where P_j is the normalized power value of the device ψ_j, reported to the cycle length t_{cycle} and m is the number of devices considered.

P_j can be described as following:

$$P_j = \sum_{k=0}^{p} \alpha_{j,k} P_j^k \quad (10)$$

where, $\alpha_{j,k}$ is a coefficient which represents the fraction relative to the cycle length for which the device ψ_j is in the state S_j^k characterized by P_j^k.

To the energy consumption, due by the running devices, we can add the energy overhead η_{Switch} given by the transitions from one state to another and the energy overheads introduced by the execution of the operating system itself (see equations 8 and 9):

$$\eta_{Swith} = \sum_{k=0}^{m} \eta_k \quad (11)$$

where, η_k is the energy overhead due to the state transition k and m is the total number of transitions during a hyper period.

B. Power/Energy Parameters Measurement Framework Component

For the Power/Energy Parameters Measurement Module, part of the TEEARTS framework, a commercial system can be used. For example the Simplicity Studio tool kit can be used with the supported platforms. For the other platforms we propose the following architecture, which is presented in Figure 2. The system is capable to measure currents from 1pA to 10A and Voltages from 100nV to more than 100V.

The system is composed of the following main components:

- *High accuracy and high frequency Multimeter Keithley DMM7510* [12], capable of measuring currents from 1pA to 10A and storing the measured data with a high sampling rate (maximum of 1.000.000 measurements per second).

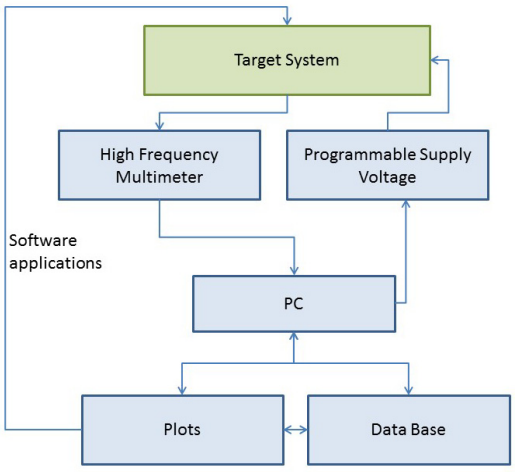

Figure 2. Power/Energy Parameters Measurement Module

- *Programmable Voltage Supply* HAMEG 7044 [13]. This is a four channel voltage Supply with a serial interface which can be used for controlling each channel.

- *The Target System* – e.g. Microcontrollers/ DSPs with different power supply values: ATMEGA16, LPC2294; complex power-aware DSPs (ex.ADSP-BF537); other power-aware targets.

Regarding the software, the system contains a PC application which controls the power supply and a software tool which extracts the measured values from the measurement device and graphically represents the power/energy consumption. The measured data can also be stored in a database containing the following fields: experiment id, voltage, current, time stamp.

C. Methodology

We propose the following measurement methodology:

- Determine all the states given by the application requirements for each device, as specified by the equation (3), starting with the processor. These can be obtained both from the datasheet and by measurement, if all the other components are kept in a neutral state (shut down if possible, or a known constant power state). For the running states with a relative constant power, the average value will be retained and in case of high power variations, the maximum value will be noted.
- For each state, the power will be noted.

In case of using a RTOS:

- The (worst case) energy overhead of each major component must be determined: the energy overhead due to the task preemption $E_{Preempt}$, the energy overhead due to the scheduler E_{Sched} and the energy consumed by other RTOS components.
- For a system which runs a set of periodical tasks, the execution will be cyclic, thus the energy consumption can have a cyclic behavior (see eq. 6). In this case the maximum number of the task preemptions and scheduler execution within a hyper period can be determined directly depending on the scheduling policy used.
- If the execution of the application requires the usage of other operating system components, the energy overhead of those components can be determined as well.

IV. Analysis and Evaluation

The target system considered for the evaluation of the TEEARTS framework is the EFM32-G890-STK, with an ARM Coretex-M3 Gecko processor.

For this study we have considered two simple main applications: an application which toggles four LEDs and one application which computed prime numbers.

In the first scenario there is a single task with 100ms period and a worst case execution time bellow 1ms, which counts from 0 to 15 and displays the count on four LEDs. In consequence the LEDs are toggled with different

frequencies: LED0 with 10Hz, LED1 with 5Hz, LED2 with 2.5 Hz and LED4 with 1.25 Hz.

According to the methodology proposed, in the first step we have determined the states of the processor ψ_0 and of the other devices used ψ_{1-4} (the LEDs in our case).

In order to simplify our approach, we have considered the average power values expressed in mW with two decimals. For the processor we have considered the following states: $S_0^0 = \{9.57mW\}$ - a running state (EM0), with core clock frequency of 14 MHz and peripheral clock of 14MHz and $S_0^1 = \{3.2mW\}$ - a CPU sleep state (EM1).

For the LEDs we consider only two states $S_{1-4}^0 = \{3.65$ mW$\}$ - on, $S_{1-4}^0 = \{2$ mW$\}$ - off.

In this case, even though the set consists of periodical tasks, the energy consumption period is larger than their hyperperiod and is equal to the task period multiplied by 16 (the number of the configurations of the four LEDs).

If we consider the following values for the parameters in equations (9) and (10): P_j^k is the power value corresponding to S_j^k, for the processor $\alpha_{0,0} = 0.01$ and $\alpha_{0,1} = 0.9$, (the difference of 0.09 is considered to be the processor usage of the RTOS and can be considered an unknown state), for all the LEDs we have the same value: $\alpha_{1-4,0-1} = 0.5$. Thus the application energy estimation according to equation (9) becomes:

$$E_{tot} = \sum_{j=0}^{4} P_j * 1.6s + \eta_{Switch} =$$
$$= 1.6s(P_0 + P_1 + P_2 + P_3 + P_4)mW + \eta_{Switch} =$$
$$= 1.6(0.01 * 9.57 + 0.9 * 3.2 +$$
$$+ 0.5 * 4 * 3.65 + 0.5 * 4 * 2) + \eta_{Switch} =$$
$$= 14.27mJ + \eta_{Switch}$$

The energy value measured for this application being 14.09 mJ without RTOS and 14.31 mJ with RTOS.

In Figure 5 and Figure 6, we can see the execution of the application with and without a RTOS and we can observe that there is a relatively small energy overhead introduced by the RTOS. This difference is marked in our model as E_{RTOS} and is better represented in Figure 3.

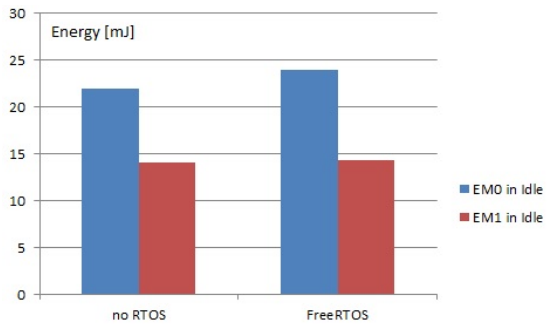

Figure 3. 4 LED Counter Application - No Sleep in Idle Mode

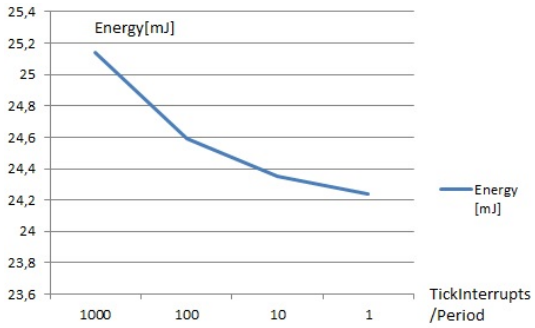

Figure 4. Number of Interrupts vs Energy Consumption

One component of the total energy consumed by the RTOS is the energy consumption due to the interrupts. A graphical analysis of the evolution for the energy consumed by the first application in different scenarios, depending on the system tick interrupts per task period can be seen in Figure 4.

For the second application, which computes prime numbers and runs on FreeRTOS, we have performed a set of experiments for different number of tasks. The measurements show no noticeable differences in energy consumption values for different number of tasks for this application.

In this case the measured energy for a cycle is 6.3mJ, $\alpha_{0,0-1} = 0.47$ and the energy estimation for this case is:

$$E_{tot} = \sum_{j=0}^{4} P_j * 1s + \eta_{Switch} =$$
$$= (P_0)mW * 1s + \eta_{Switch} =$$
$$= (0.47 * 9.57 + 0.47 * 3.2) + \eta_{Switch} =$$
$$= 6mJ + \eta_{Switch}$$

V. Conclusions and Future Work

In this paper we propose a framework and a methodology for determining the energy consumed by a real-time embedded system running periodic tasks.

In this sense, we have described: a framework architecture, a methodology and a consumption model for the targeted system. With the help of this framework, we have run a set of measurement, regarding the energy consumption variations of a target system in different scenarios (ranging the number of tick interrupts, changing the RTOS). The main results where graphically represented and interpreted.

In the future we plan to use the proposed framework to measure and compare different operating systems based on highly used and on new developed energy efficiency benchmarks.

Acknowledgment

This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS – UEFISCDI, project number PN-II-RU-TE-2014-4-0731

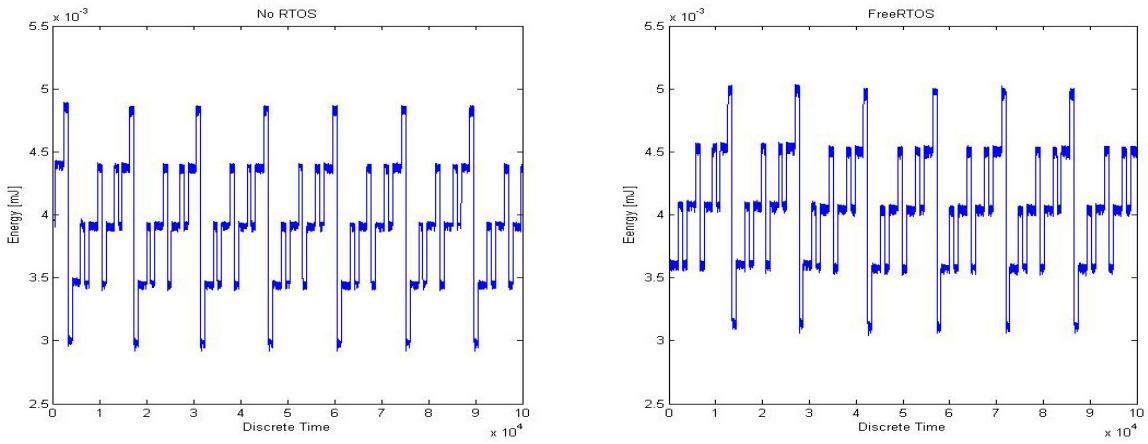

Figure 5. No RTOS vs RTOS with No Sleep in Idle

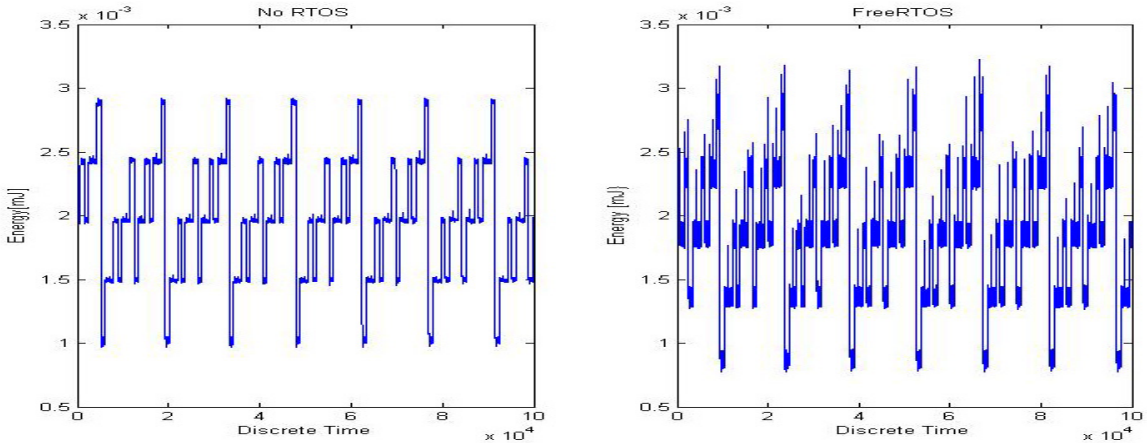

Figure 6. No RTOS vs RTOS with EM1 in Idle

REFERENCES

[1] C. S. Stangaciu, M. V. Micea, and V. I. Cretu, "Energy efficiency in real-time systems: A brief overview," in Applied Computational Intelligence and Informatics (SACI), 2013 IEEE 8th International Symposium on, 2013, pp. 275-280.

[2] M. Bhardwaj, R. Min, and A. Chandrakasan, "Power-aware systems," in Signals, Systems and Computers, 2000. Conference Record of the Thirty-Fourth Asilomar Conference on, 2000, pp. 1695-1701.

[3] C. S. Stângaciu, A. M. Horvath, M. V. Micea, V. I. Cretu, and V. Groza, "Analysis and improvements in energy consumption models for RTS," in Applied Computational Intelligence and Informatics (SACI), 2015 IEEE 10th Jubilee International Symposium on, 2015, pp. 277-282.

[4] H. Mehta, R. M. Owens, and M. J. Irwin, "Instruction level power profiling," in Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings, 1996 IEEE International Conference on, 1996, pp. 3326-3329.

[5] A. Sinha and A. P. Chandrakasan, "JouleTrack: a web based tool for software energy profiling," presented at the Proceedings of the 38th annual Design Automation Conference, Las Vegas, Nevada, United States, 2001.

[6] H. Bayer and M. Nebel, "Evaluating Algorithms according to their Energy Consumption," Mathematical Theory and Computational Practice, no. p. 48, 2009.

[7] C. Bunse, H. Hopfner, E. Mansour, and S. Roychoudhury, "Exploring the energy consumption of data sorting algorithms in embedded and mobile environments," in Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference on, 2009, pp. 600-607.

[8] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, and K. W. Cameron, "Powerpack: Energy profiling and analysis of high-performance systems and applications," Parallel and Distributed Systems, IEEE Transactions on, vol. 21, no. 5, pp. 658-671, 2010.

[9] B. Ouni, C. Belleudy, and E. Senn, "Accurate energy characterization of OS services in embedded systems," EURASIP Journal on Embedded Systems, vol. 2012, no. 1, pp. 1-16, 2012.

[10] K. Baynes, C. Collins, E. Fiterman, B. Ganesh, P. Kohout, C. Smit, T. Zhang, and B. Jacob, "The performance and energy consumption of embedded real-time operating systems," Computers, IEEE Transactions on, vol. 52, no. 11, pp. 1454-1469, 2003.

[11] D. Li, S. Hao, W. G. Halfond, and R. Govindan, "Calculating source line level energy information for android applications," in Proceedings of the 2013 International Symposium on Software Testing and Analysis, 2013, pp. 78-89.

[12] "DMM7510 7½-Digit Graphical Sampling Multimeter," Keithley, Ed., ed. http://www.farnell.com/datasheets/1869578.pdf.

[13] Hameg Instruments. Quadruple High-Performance. Power Supply HM7044 Available: http://www.teknetelectronics.com/DataSheet/HAMEG/WEBHAMEGHM7044.pdf