OCEMES: Measuring Overall and Component-based Energy Demands of Mobile and Embedded Systems

Christian Bunse

Software Systems
University of Applied Sciences of Stralsund
Zur Schwedenschanze 15
18435 Stralsund, Germany
Christian.Bunse@fh-stralsund.de

Hagen Höpfner

Mobile Media Group
Bauhaus-Universität Weimar
Bauhausstraße 11
99423 Weimar, Germany
hoepfner@acm.org

Abstract: Resource scarcity is a common challenge in mobile and embedded computing. Mobile and embedded devices are typically battery-driven. Thus, their energy demand is in the focus of attention. Various techniques for saving energy exist; Their evaluation warrants approaches for measuring the energy demand during the use of the devices. Modern smart-phones offer sensors and corresponding APIs for gathering energy data, but they are mostly limited to overall measurements regarding the characteristics of the entire system. Moreover, most embedded systems lack such sensors. We present an approach for measuring the energy demand of mobile and embedded systems through additional hardware. It allows for measuring the energy demand of individual (hardware) components, as well as the entire system and therefore enables researchers in the field of energy-aware computing to draw more precise conclusions.

1 Introduction and Motivation

Energy is one of the most limiting factors of information and communication technologies. Especially mobile and embedded devices do *not* have a permanent power supply, but depend on rechargeable batteries. Due to increases in hardware performance, display sizes and resolution, the incorporation of more hardware components (GPS receiver, accelerometer, etc.), the devices' energy requirements are still growing. Energy-aware software development, energy-aware algorithms and energy-aware sensor substitution are only three examples of recently initiated research areas that try to reduce energy requirements by optimising the software rather than the hardware [HB11]. A common problem of researchers is the lack of guidelines and approaches to evaluate results. However, precise measurements are an integral part of systematic research. In this paper, we address the question on how to systematically measure energy requirements on smartphones and embedded systems. It can be used to obtain data at low levels of abstraction. A typical example is the energy demand of specific hardware components in relation to a utilised software. Our approach is scalable, cost-efficient and easy to use. It provides a valuable tool for researchers and practitioners who are interested in optimising the energy characteristics of their embedded or mobile applications. In other papers we discussed alternative approaches that

focus on software-driven measurements for smartphones [SH12]. In [HSB12], we have shown that such approaches are appropriate for doing comparative measures regarding accumulated energy values. In this paper, we introduce an experiment setup for measuring energy requirements of dedicated hardware components. This is not possible by using the smartphone's energy APIs only.

The remainder of this paper is structured as follows. Section 2 briefly introduces basic principles of physics and electrical engineering regarding the measurement of energy demands. Section 3 gives an overview of the proposed measurement set-up and discusses its use in the context of embedded and smartphone-based systems. Section 4 presents results of the evaluation of our approach. Section 5 provides a summary and draws conclusions.

2 Principles of physics

All electricity effects depend on moving charges that –in turn– depend on electrons and protons. The basic terms of energy measurement are defined as follows:

- The electric current (I), which is measured in Ampere (A), represents the amount of electric charge that passes a point in an electric circuit per unit time.
- The electric voltage (U), which is measured in Volt (V), is the potential difference between two points.
- The electric power (P), which is measured in Watt (W), is the rate at which electric energy is transferred by an electric circuit.
- The energy (E), which is measured in Joule (J), is a function of voltage and current over time. In other words it is the consumed power for a given time.

As energy is a function of time, it is not possible to measure it directly. However, as discussed in [HB10], it is possible to indirectly measure the energy "consumed[1]" by, e. g., a CPU. This is realised by tracking the electric power P_{CORE} consumed by the CPU core. As the power is the product of the core voltage U_{CORE} and the current flow I_{CORE}, we can calculate the energy afterwards as follows: $P_{CORE} = U_{CORE} \cdot I_{CORE}$ According to Kirchhoff's laws, the electrical current that is required by the core (I_{CORE}) equals the current through a sense resistor: $I_{CORE} = I_{SENSE}$. Following Ohm's law, I_{SENSE} equals the quotient of the voltage U_{SENSE} that is measured and the sense resistor value R_{SENSE}. Hence, I_{SENSE} can be calculated by $I_{SENSE} = \frac{U_{SENSE}}{R_{SENSE}}$, which leads to $P_{CORE} = \frac{(U \cdot U_{SENSE} - U_{SENSE}^2)}{R_{SENSE}}$. The energy value is equal to the integral of the power P_{CORE} over time t and can be calculated by $E = \int P_{CORE}(t)dt = \int \frac{U \cdot U_{SENSE} - U_{SENSE}^2}{R_{SENSE}} dt$ or approximated by $E = \frac{1}{R_{SENSE}} \Delta t \cdot \sum_{n=0}^{\frac{t}{\Delta t}} U \cdot U_{SENSE}(n \cdot \Delta t) - U_{SENSE}^2(n \cdot \Delta t)$, respectively.

[1] Please note, we wrote "consumed" in quotation marks, because energy is not consumed (in terms of using it up) but transformed into another form of energy.

3 Measurement setup

Measuring energy requires the collection of data over specific time periods. The relationship between the subject of research (e. g., an algorithm or a software component) and this time interval must be precise. To address this problem, we developed a simple measurement setup that can easily be adapted to a large number of domains, platforms, etc. Figure 1 depicts our general setup (cf., [BHRM11]). The main principle is a separation between the subject of research and the measuring device. The measurement system selects tasks from a predefined pool and sends them to the measured device that in turn, executes them. The nature of tasks is open and can range from simple sorting [BHRM11] tasks to complex sensor-based collision detections [BH11]. The actual measurement is

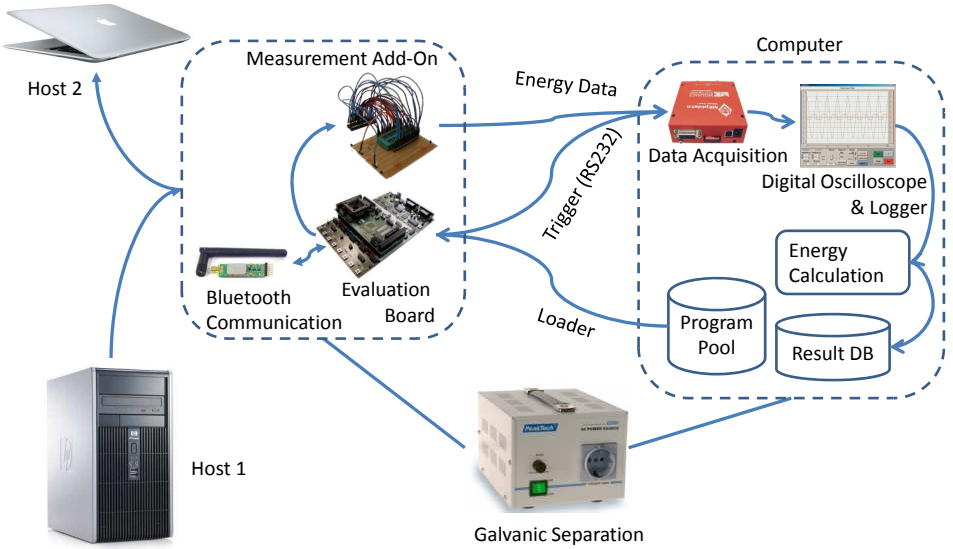

Figure 1: Experiment System Overview

performed by using a digital oscilloscope and a data logger. These devices are connected to the specific hardware components of interest (CPU, memory controller, display, etc.). We decided to use a digital device because of its processing speed and logging facilities. Measurements are started and stopped by trigger signals sent by the device under measurement, immediately before and after each task. Measurement results are collected and stored in form of CSV files and contain timestamp, trigger, current, and voltage for each measurement.

Our findings indicate that the quality of results directly depends on the sampling interval of the measurement. The shorter the sampling interval, the better the calculation result becomes. Our experiments have shown that reliable results require a sampling interval of at least 1 µs. As a consequence, our approach generates 10^6 data points every second. The resulting large amount of data implies that measurements should therefore only be

collected for a limited time period and *not* for the lifetime of a service.

The general strategy for hardware-based measurements of software-related energy demand has successfully been used in a number of contexts and for several devices [BHRM11, HB10]. The biggest advantage of this approach is its use of trigger signals, which are sent immediately before and after a specific software artifact is executed. This allows us to precisely relate software use and energy demand. Another advantage is that the approach can be used to likewise measure energy demand of processors, memory, etc. There is, however, a significant drawback. It requires measurement interfaces (pins) at each hardware component and significant preparation efforts.

In addition to embedded systems, the approach has also been successfully used in the context of mobile systems [BH11, BKS11]. At this, we integrated a modified smartphone in the general measurement environment (see Figure 2). The phone was altered so that the former USB-Port can be used for connecting measurement lines as well. Connectors were exchanged and extended. The disadvantage being that when connecting the phone to a PC or to its charger, the altered connector has to be used.

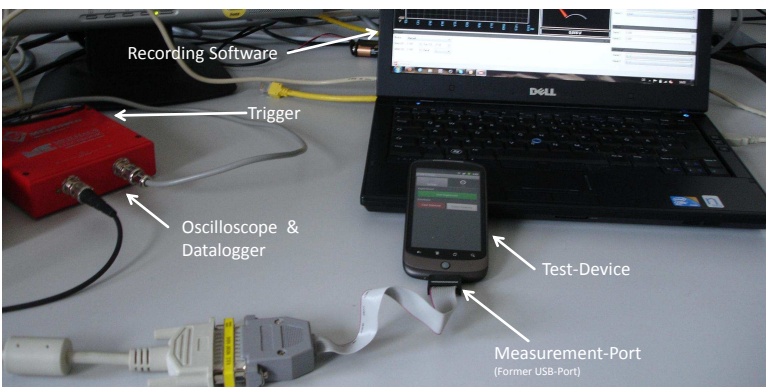

Figure 2: Typical set-up for measuring energy demands of (components of) a smartphone.

Modifications to the operating system were needed to avoid problems due to high system loads (i.e., on Android systems one can easily watch more than 70 parallel running threads), parallel execution, and to enable triggering for loss-free measurements. We stripped the operating system and switched off as many services and processes as possible. This custom operating system can be used across a large number of Android devices. However, measurements are only possible if the hardware allows. Smartphones that are highly integrated and that do not provide specific interfaces will be hard to evaluate. Out of these reasons we selected a Google Nexus One device. It is a developer phone that allows, due to an open boot loader, to install modified Android versions without the need for jailbreaking. It makes use of a 1 GHz, single-core Qualcomm processor (aka SnapDragon). So, problems that may arise when measuring energy requirements during parallel execution are avoided. It can easily be disassembled (cf. Figure 3(a)) in order to get access to relevant hardware components, and it can be altered in order to attach measurement lines. A closer look to the main board (see Figure 3(b)) shows that it is quite easy to identify its

power management facilities. The SnapDragon processor is located on the reverse side of the board, providing access to power lines and other elements. We attached sense resistors to the (direct) voltage supplies of core, power supplies, network units, etc. The digital oscilloscope continuously measures the voltage drops at theses resistors. Data is sent to and stored/processed by a central unit (a small industrial personal computer) that pre-processes the raw data. The host takes the measured voltages and calculates energy values (in Joule) for a predefined time period.

(a) Hardware components of the Nexus One

(b) Nexus One main board

Figure 3: Detailed view on the internals of the used Nexus One smartphone. (Both subfigures were published by Walter Galan at `http://www.ifixit.com/Teardown/Nexus-One-Teardown/1654/2` under the Creative Commons BY-NC-SA license. Thanks to him we did not have to reopen our device in order to prepare our own images. Thank you Walter!)

Live analysis of measurement data requires high-speed data transmission and processing. This would make the measurement approach inflexible and difficult to scale. Therefore, we decided to follow a post-treatment strategy: measurement data is collected and stored in a data file at first. Then, the collected data is analysed (i. e. to extrapolate energy values from measured current and voltage samples) after runs have been performed. However, logging voltages with a high frequency (sampling period below 1 µs results in a large number of data points. In the worst case, we collect more than 1.000.000.000 data points per second. Our experiments published in [BH11, BKS11] have shown that a sampling period of 45 µs (22.000 samples per second) results in meaningful data.

4 Evaluation showcase

Regarding the overall energy demand of embedded systems, the presented framework was already evaluated in various application scenarios (cf., [HB10, BHRM11, BH11]). We adapted it to mobile systems in order to judge software-based measurement approaches provided by Android platforms. Due to the given space limitations, the following discussion focusses on overall, CPU, and memory energy demands of the respective Nexus

One measurement scenario presented in [HSB12]. We defined simple algorithms for each combination of the four major complexity classes (i. e., linear, polynomial, logarithmic, exponential) and executed them several times with an increasing input data size; display and all wireless communication interfaces were switched off; there was no user interaction during the measurement phase of the experiments; we only used internal memory.

Table 1 provides a compressed overview of the obtained results regarding the different combinations of runtime and space complexity. Per combination, it shows the following values: time required for the execution (e. g., 5.5 s), input data size n (e. g., $n = 2.5$M, i. e. 2.500.000), overall energy required E_o, energy demand by the CPU E_{CPU}, energy demand by the memory E_{RAM}, and the difference E_{rest} between E_o and the sum $E_{CPU} + E_{RAM}$ as well as its percentage. The latter shows that longer measures reduce experimental noise. However, all results illustrate that our setup works fine in terms of precision because in all cases less than 0.35 % of the overall energy demand resulted from background activities such as the operating system.

	runtime:	**log**	**lin**	**poly**	**exp**
space	**log**	5.5 s / $n = 2.5$M	5.0 s / $n = 2.5$M	2.4 s / $n = 750$	1.9 s / $n = 15$
	E_o	0.267 393 233 J	0.277 779 386 J	0.187 344 887 J	0.352 561 951 J
	E_{CPU}	0.060 774 376 J	0.057 467 053 J	0.037 662 773 J	0.136 502 168 J
	E_{RAM}	0.206 218 857 J	0.220 112 333 J	0.149 382 114 J	0.215 159 783 J
	E_{rest}	0.0004 J / 0.1496 %	0.0002 J / 0.0720 %	0.0003 J / 0.1601 %	0.0009 J / 0.2553 %
	lin	149.2 s / $n = 2.5$M	148.9 s / $n = 2.5$M	2.7 s / $n = 750$	2.0 s / $n = 15$
	E_o	57.218 169 45 J	55.277 561 38 J	0.213 874 96 J	0.352 561 951 J
	E_{CPU}	13.726 360 67 J	13.257 214 73 J	0.050 029 99 J	0.101 699 828 J
	E_{RAM}	43.490 808 79 J	42.019 946 65 J	0.163 544 969 J	0.155 028 174 J
	E_{rest}	0.001 J / 0.0017 %	0.0004 J / 0.0007 %	0.0003 J / 0.1403 %	0.0008 J / 0.3106 %
	poly	31.4 s / $n = 750$	30.5 s / $n = 750$	30.6 s / $n = 750$	2.1 s / $n = 15$
	E_o	11.629 093 14 J	11.643 259 19 J	12.557 835 64 J	0.259 478 817 J
	E_{CPU}	2.787 582 353 J	2.786 682 205 J	3.004 080 553 J	0.103 183 852 J
	E_{RAM}	8.841 110 784 J	8.855 876 984 J	9.552 955 085 J	0.155 394 965 J
	E_{rest}	0.0004 J / 0.0034 %	0.0007 J / 0.0060 %	0.0008 J / 0.0064 %	0.0009 J / 0.3468 %
	exp	163.7 s / $n = 15$	164.9 s / $n = 15$	165.1 s / $n = 15$	165.5 s / $n = 15$
	E_o	69.972 278 6 J	72.603 794 29 J	72.813 618 01 J	71.815 492 54 J
	E_{CPU}	16.786 946 86 J	17.423 510 63 J	30.741 372 25 J	30.326 504 05 J
	E_{RAM}	53.184 931 74 J	55.179 883 66 J	42.071 745 76 J	41.488 188 49 J
	E_{rest}	0.0004 J / 0.0006 %	0.0004 J / 0.0006 %	0.0005 J / 0.0007 %	0.0008 J / 0.0011 %

Table 1: Compressed comparison of results.

The energy values listed in Table 1 represent the energy demand for executing the algorithm with the given n elements. As such, the overall measurement results are not surprising and follow the findings as reported by us in [BHRM11, BH11]. However, when examining the energy values regarding CPU and RAM, it shows that memory operations are far more expensive regarding energy than CPU-based operations. Optimisations should therefore reduce the memory usage of applications in order to save energy. Optimising performance will also contribute to the improvement of energy demand, but far less than expected. Thus, the widely accepted belief that energy demand strongly correlates to performance has to be further examined. This is inline with the findings presented in [BHRM11]. However, in contrast to environments that do not make use of an operating system (i. e., microcontroller), findings are less expressive, but clearly without ambiguity.

5 Summary, Conclusions, and Outlook

In this paper, we presented the OCEMES framework for measuring overall and software-driven component-based energy demand in the context of mobile and embedded systems. We introduced the measurement setup and discussed its limitations. Furthermore, we highlighted its applicability by summarising various measurement results. We demonstrated that the framework provides reliable and precise energy demand data. However, these results come with a price. Unfortunately, the approach cannot simply be transferred to other smartphones as hardware modification is required. In addition, the operating system needs to be modified as well (triggering, disabling of cores, etc.). We therefore would require an unlocked boot loader and the possibility to install a custom operating system. In summary, the approach needs upfront investment prior to data collection. Other approaches (cf., [HSB12]) are aiming at more cost-effective strategies. However, if precise data at a low level of abstraction is needed, the OCEMES approach is a good choice.

References

[BH11] Christian Bunse and Hagen Höpfner. Analyse des Zusammenhangs zwischen Energiebedarf, Dienstgüte und Performanz bei der Ressourcensubstitution in Softwaresystemen. In Wolfgang A. Halang, editor, *Herausforderungen durch Echtzeitbetrieb*, pages 101–110, Berlin / Heidelberg, November 2011. Springer. in German.

[BHRM11] Christian Bunse, Hagen Höpfner, Suman Roychoudhury, and Essam Mansour. Energy efficient data sorting using standard sorting algorithms. In José Cordeiro, Alpesh Ranchordas, and Boris Shishkov, editors, *Software and Data Technologies*, volume 50 of *Communications in Computer and Information Science*, pages 247–260, Berlin / Heidelberg, 2011. Springer.

[BKS11] Christian Bunse, Sonja Klingert, and Thomas Schulze. GreenSLAs for the Energy-efficient Management of Data Centres. In *Proceedings of the 2nd international conference on energy-efficient computing and networking (E-Energy2011)*, 2011. accepted for publication, forthcoming.

[HB10] Hagen Höpfner and Christian Bunse. Energy Aware Data Management on AVR Micro Controller Based Systems. *ACM SIGSOFT Software Engineering Notes*, 35(3), May 2010. `http://portal.acm.org/citation.cfm?id=1764810.1764820`.

[HB11] Hagen Höpfner and Christian Bunse. Energy Awareness Needs a Rethinking in Software Development. In Maria Jose Escalona, Boris Shishkov, and José Cordeiro, editors, *Proceedings of the 6th International Conference on Software and Data Technologies*, volume 2, pages 294–297, Setúbal, Portugal, July 2011. INSTICC, SciTePress.

[HSB12] Hagen Höpfner, Maximilian Schirmer, and Christian Bunse. On measuring smartphones' software energy requirements. In *Proceedings of the 7th International Conference on Software Paradigm Trends, ICSOFT 2012, Rome, Italy, July 24-27, 2012*, Setúbal, Portugal, 2012. INSTICC, SciTePress. accepted for publication, forthcoming.

[SH12] Maximilian Schirmer and Hagen Höpfner. Software-based Energy Requirement Measurement for Smartphones. In *Proceedings of the 42. annual conference of the German computer society (Gesellschaft für Informatik e.V. (GI))*, Lecture Notes in Informatics (LNI), Bonn, Germany, 2012. accepted for publication, forthcoming.