Metrol. Meas. Syst., Vol. XX (2013), No. 3, pp. 419–430.

METROLOGY AND MEASUREMENT SYSTEMS

Index 330930, ISSN 0860-8229

www.metrology.pg.gda.pl

METHODS FOR MEASUREMENT OF ENERGY CONSUMPTION IN MOBILE DEVICES

Robertas Damaševičius[1), Vytautas Štuikys[1), Jevgenijus Toldinas[2)

[1) Kaunas University of Technology, Software Engineering Department, Studentų st. 50-410, LT-51368, Kaunas, Lithuania (✉ robertas.damasevicius@ktu.lt, +37 370 300 399)

[2) Kaunas University of Technology, Computer Science Department, Studentų 50-209, LT-51368, Kaunas, Lithuania (✉ eugenijus.toldinas@ktu.lt, +37 370 300 389)

Abstract

Mobile devices have become an integral part of our life and provide dozens of useful services to their users. However, usability of mobile devices is hindered by battery lifetime. Energy conservation can extend battery lifetime, however, any energy management policy requires accurate prediction of energy consumption, which is impossible without reliable energy measurement and estimation methods and tools. We present an analysis of the energy measurement methodologies and describe the implementations of the internal (profiling) software (proprietary, custom) and external software-based (Java API, Sensor API, GSM AT) energy measurement methodologies. The methods are applied to measure energy consumption on a variety of mobile devices (laptop PC, PDA, smart phone). A case study of measuring energy consumption on a mobile computer using 3DMark06 benchmarking software is presented.

Keywords: battery management, energy consumption, energy measurement, mobile computing.

1. Introduction

In portable electronic devices such as laptops, tablet PCs or smart phones, energy is an important constraint, because 1) mobile computers are not always connected to a stationary power supply, but rather are supplied from batteries; 2) complex hardware components and mobile apps require ever more energy; 3) the portability requirement imposes severe constraints on the size and weight of a hand-held system, including its batteries.

While computing and communication capabilities of hand-held devices and complexity of hardware and mobile services have increased by several orders of magnitude in the past two decades, the battery energy density has only tripled in the same period of time [1]. Consequently, the energy budget is a single most important factor that limits usability and evolution of mobile devices [2]. As a response, 80% of mobile phone users take measures to increase their battery lifetime [3]. Therefore, energy consumption evaluation, awareness, and management methods for extending operational time of mobiles are a hot topic now.

Extensive research exists on improving battery lifetime of mobile computing systems. Energy conservation allows to extend battery life and to include more services that can be provided by a device for the same battery capacity. In particular, human factors such as charging behavior, understanding of battery indicators and customizing power-saving settings, have been considered [3]. To facilitate energy conservation through different modes of operation, the methods that allow predicting power consumption both at the device and device component level are needed [4]. However, any energy management policy requires accurate prediction of energy consumption, which is impossible without reliable energy measurement and estimation methods and tools.

Article history: received on Nov. 06, 2012; accepted on May. 18, 2013; available online on Sep. 15, 2013; DOI: 10.2478/mms-2013-0036.

Considering a vast variety of hardware architectures, operating systems, drivers and device connection types, development of usable and reliable energy measurement methods is a formidable scientific problem. The researchers use battery models (e.g., [5] to avoid problems related with direct measurement and profiling of energy consumption. The models are simple linear models or complex models that incorporate non-linear behavior of a battery such as voltage hysteresis or the relaxation phenomenon. Nevertheless, such models often lack validation. Another approach is to use energy consumption estimation models to estimate the energy cost of an application, which can be utilized to make subsequent energy management decisions based on user input and sustainable battery life [6]. This includes low-level energy modeling [7], energy macro-modeling [8], and system-level energy modeling [9].

Energy measurement can be performed at hardware, instruction-level [10], OS [11], algorithmic, data, application software and user [12] levels. On the application level the aims are: 1) to develop new applications consuming less energy; 2) to improve existing applications using more energy-efficient algorithms or computation methods; 3) to select optimal parameter value combinations for such applications. On the user level, the responsibility for energy management is left to the user. The energy-aware user should be able to consciously select applications which are needed for his/her work and entertainment and consume less energy while provide their services at the acceptable level of quality. In all these cases it is important to know energy consumption characteristics of the application software, analyze the reasons for energy waste, and to promote energy-lean applications, which a user can use to assemble his/her energy-efficient digital environment.

Here we present an analysis of the energy measurement methods and describe three energy measurement methods proposed and used by the authors of this paper on a variety of mobile devices. Note that we use energy measurement as a tool to analyze energy efficiency of an entire mobile computing system (hardware and software) rather than stand-alone applications or battery properties. The structure of the paper is as follows. Section 2 presents an overview of energy measurement methods. Section 3 describes the energy measurement methods used by the authors of this paper. Section 4 summarizes the results of our previous case studies and presents a new case study of energy measurements using the 3DMark06 benchmarks. Finally, Section 5 presents conclusions and discusses future work.

2. Overview of energy measurement methodologies

Previous studies on the measurement of energy consumption in mobile devices are based on 1) hardware-based measurements, 2) energy profiling software running on a mobile device, and 3) external measurement software running on a PC connected to a mobile device.

When using the hardware-based methodology (see Fig. 1, a), a mobile device is treated as a black-box and measurement of energy is simplified to measurement of the supply voltage and current on the battery terminals. The measurement system consists of a device-under-test (DUT), a hardware data acquisition system (usually a measurement resistor, a digital multimeter and a data acquisition board), and a PC. Unfortunately, this method cannot be used for nickel-based batteries [13]. Moreover, inserting the measurement resistor increases circuit resistance, and therefore its power consumption. For example, Rice and Hay [14] insert a high-precision 0.02 Ω measurement resistor between a battery terminal and its connector on the phone, and use a PCI-MIO-16E-4 sampling board to measure battery voltage and voltage drop across the measurement resistor. Mayo and Ranganathan [15] measure total system power using a 0.10 Ω sense resistor between power source and the device. To reduce noise, the voltage across the resistor is amplified and measured by a data acquisition system. Thiagarajan *et al.* [16] use an Agilent 34410A multimeter to obtain a voltage drop on a 0.1 Ω resistor placed in series with the ground at a sampling rate of 1 kHz. The voltage drop is then used to calculate the phone's instantaneous power consumption.

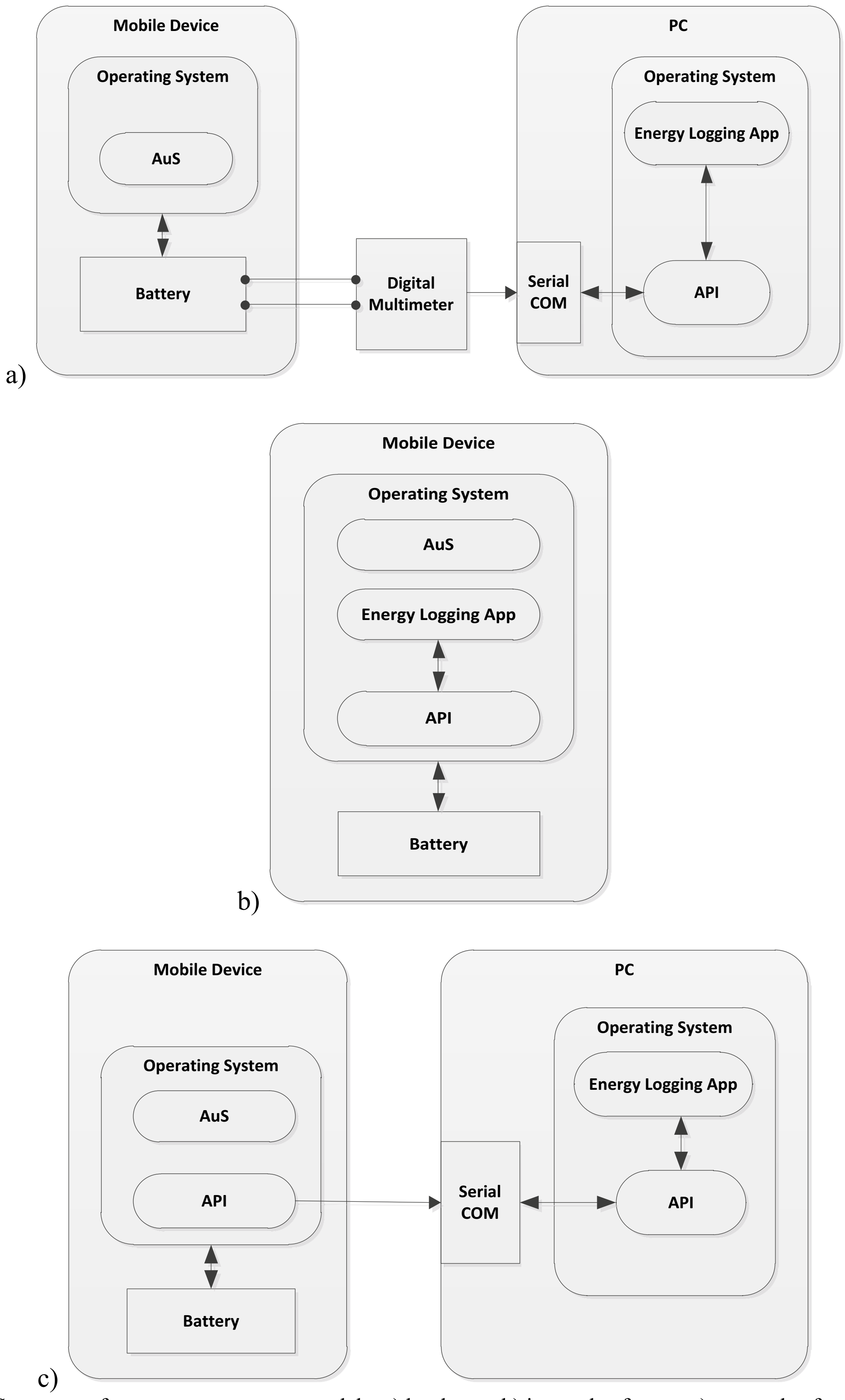

Fig. 1. Summary of energy measurement models: a) hardware, b) internal software, c) external software.

When using the internal (profiling) software based methodology, the external equipment is not required. However, energy profiling software, such as Nokia Energy Profiler (NEP), is needed, which may be only available for certain mobile devices. Nurminen and Noyranen [17] use NEP to measure voltage and current for Nokia S60 mobile phones in runtime. Balasubramanian *et al.* [18] use both NEP as well as a Monsoon hardware power meter (http://www.msoon.com) to measure energy consumption on the HTC Fuze phone that runs Windows Mobile 6.5. Xiao *et al.* [19] use NEP on Nokia N95 running Symbian 60 OS for collecting battery current, voltage and battery temperature during runtime. Vallina-Rodriguez *et al.* [20] use Resources Tracker, a background process implemented using Android public APIs, which samples the status of battery level, temperature, current, voltage and charging status every 10 sec. The disadvantage of the method is additional power consumption and a shortened battery lifetime caused by the background process running on a mobile device.

When using the external software based methodology, a mobile device is connected to an PC, whose dedicated software queries a device for its energy-related characteristics. This allows to measure energy consumption without interacting directly with the phone, avoiding pressing any keys and having backlight of the display turned on, which could lead to misleading results. For example, Flinn and Satyanarayanan [21] use PowerScope for measuring energy consumption. The measurement system consists of two computers: the profiling computer and the data-collection computer. The tool itself has three components: System Monitor (SM), Energy Monitor (EM) and Energy Analyzer (EA). The SM samples system activity on the profiling computer by periodically recording information of the currently executing process. The EM runs on the data-collection computer and uses a digital multimeter to sample the current being drawn by the profiling computer from its power source. The EA runs on the profiling computer and uses the raw sample data collected by SM and EM to generate the energy profile. The EM runs on the data collection computer and communicates with the digital multimeter. The reported voltage variation is extremely small (less than 0.25%). Perrucci *et al.* [22] use Python scripts for energy measurement to control the Nokia N95 phone running Symbian OS 9.2. The authors observe that there is no significant penalty in terms of energy and performance by using Python as compared to Symbian/C++.

Table 1. Summary of energy measurement methods in mobiles

Methodology	**Mobile device**	**OS**	**External hardware**	**Software**	**Ref.**
Hardware	Openmoko Neo Freerunner	Android 1.5	Current-sense resistor, data acquisition board	National Instruments DAQmxBase 3.3 library	[3]
	Nokia S60	data n/a	Measurement resistor, data sampling board	data n/a	[11]
	data n/a	data n/a	Sense resistor, data acquisition board	data n/a	[12]
	Android Developer Phone 2	Android	Digital power multimeter	Custom profiler application	[16]
Internal profiling software	data n/a	Windows Mobile	Not required	Phone logging tool	[1]
	Nokia S60	data n/a		Nokia Energy Profiler	[17]
	HTC Fuze	Windows Mobile		Monsoon	[18]
	Nokia N95	Symbian 60		Nokia Energy Profiler	[19]
	data n/a	Android		Resource Tracker	[20]
External software	Laptop	NetBSD	Not required	Powerscope	[21]
	Nokia N95	Symbian OS 9.2		Python scripts	[22]

Energy measurement methodologies are summarized in Table 1, while energy measurement models are summarized in Fig. 1. In the hardware model, a digital multimeter is connected to the outputs of the battery and an Application under Study (AuS) is run. The current/voltage metering results are transmitted to a PC via a serial communication (COM) port and are logged by the Energy Logging Application (ELA) using the COM API functions. In the internal software model, both the AuS and the ELA are run on the same mobile device, and the ELA uses battery charge functions provided by the OS API. In the external software model, the AuS runs on a mobile device, while the ELA runs on a PC connected to the mobile device. The ELA uses the mobile device API to query battery charge values and the PC's OS API to get values transmitted from the mobile device to the PC.

The advantages and disadvantages of the energy models are summarized in Table 2.

Table 2. Evaluation of advantages and disadvantages of energy measurement models

Model	**Advantage**	**Disadvantage**
Hardware	Hardware-based measurements are considered to be most reliable	Noise, power loss during measurement; restriction to certain types of battery; additional hardware is required
Internal profiling software	No additional hardware device or a PC is required	Additional power consumption caused by energy logging application; restricted by functions provided by OS API and/or proprietary applications; installation problems
External software	Direct interaction with phone and additional applications are not required	Restricted by functions provided by OS API and/or proprietary applications; data communication problems

3. Energy measurement methodology

In our research, we analyze two methodologies for measuring energy consumption in mobile devices.

For the internal software based methodology, we have developed an algorithm (see Fig. 2) that enables to perform measurements of energy consumption and obtain the desired relationships between energy and performance characteristics of the mobile device. We use the Microsoft Visual Studio .NET Development Environment and C# programming language, so the need to access native platform API function is greatly diminished because .NET wraps up most of the functionality available on the platform.

We have developed the Pocket PC application that measures the battery status of the device. The application consists of two components: a business object and a smart client application object. To access a platform function such as battery, a life time measurement that is not exposed by the .NET Compact Framework, we make native API calls. The business object wraps native API calls for `GetSystemPowerStatusEX2`, to return the `SYSTEM_POWER_STATUS_EX2` structure with the `BatteryLifePercent` and `BatteryVoltage` values. The client application object uses the API wrapper business object and does some algorithmic calculations that provide observations of energy dissipation.

The measurement procedure using the external-software-based methodology is summarized in Fig. 3 and explained below. We use three different methods:

1) Java API based method: battery charge is measured by reading the Java system property 'batterylevel' (different phones may have differing names of this property) during execution of the Java application. The snippet of code in Fig. 4 returns the current battery level (0% – 100%). The method can only be used for a Java-enabled smart phone.

2) Sensor API based method: battery charge is measured by reading sensor values that return the battery charge. This works only for devices that support JSR 256 Mobile Sensor API (http://jcp.org/en/jsr/detail?id=256). The API defines generic sensor functionality

optimized for the resource-constrained devices like mobile devices. The snippet of code in Fig. 5 calculates the current battery charge level (0% – 100%).

3) GSM modem-based method: battery charge is measured by connecting an external PC via USB to the GSM modem of a phone and issuing a specific command of the Hayes (AT) command language. The AT+CBC command returns the battery charge value (0% – 100%).

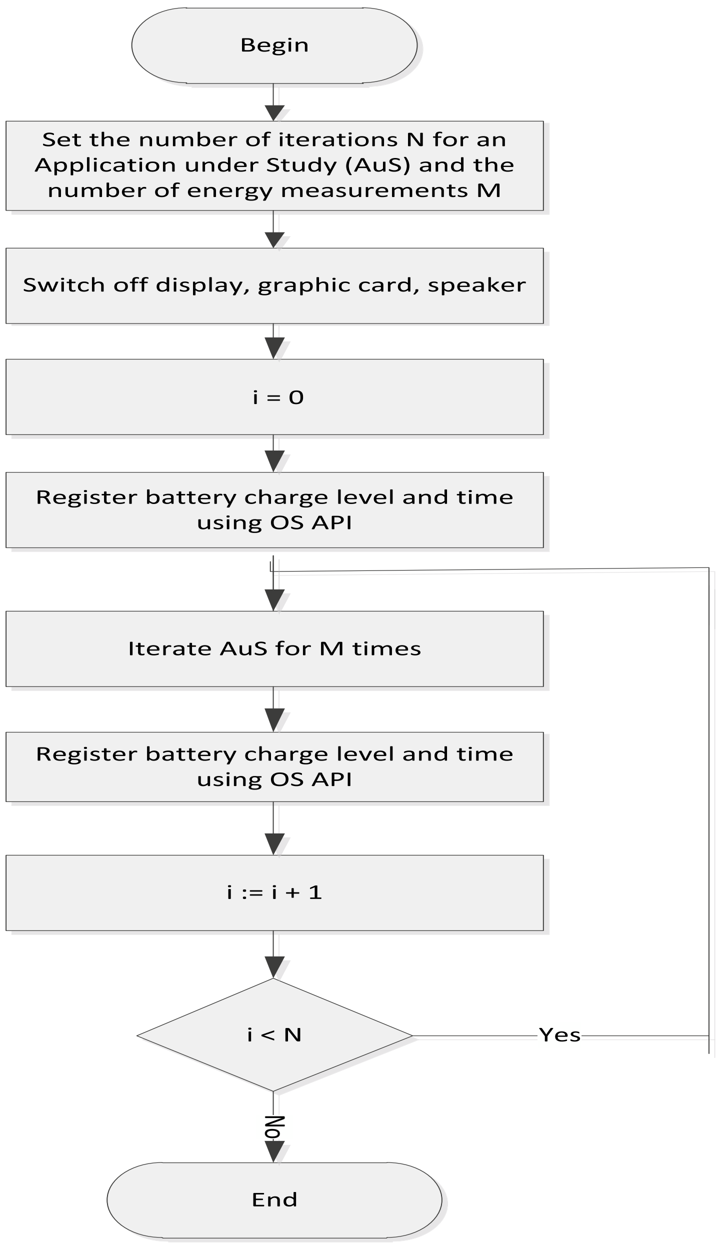
Begin
Set the number of iterations N for an Application under Study (AuS) and the number of energy measurements M
Switch off display, graphic card, speaker
i = 0
Register battery charge level and time using OS API
Iterate AuS for M times
Register battery charge level and time using OS API
i := i + 1
i < N
Yes
No
End

Fig. 2. Energy logging algorithm for energy measurement using the internal software model.

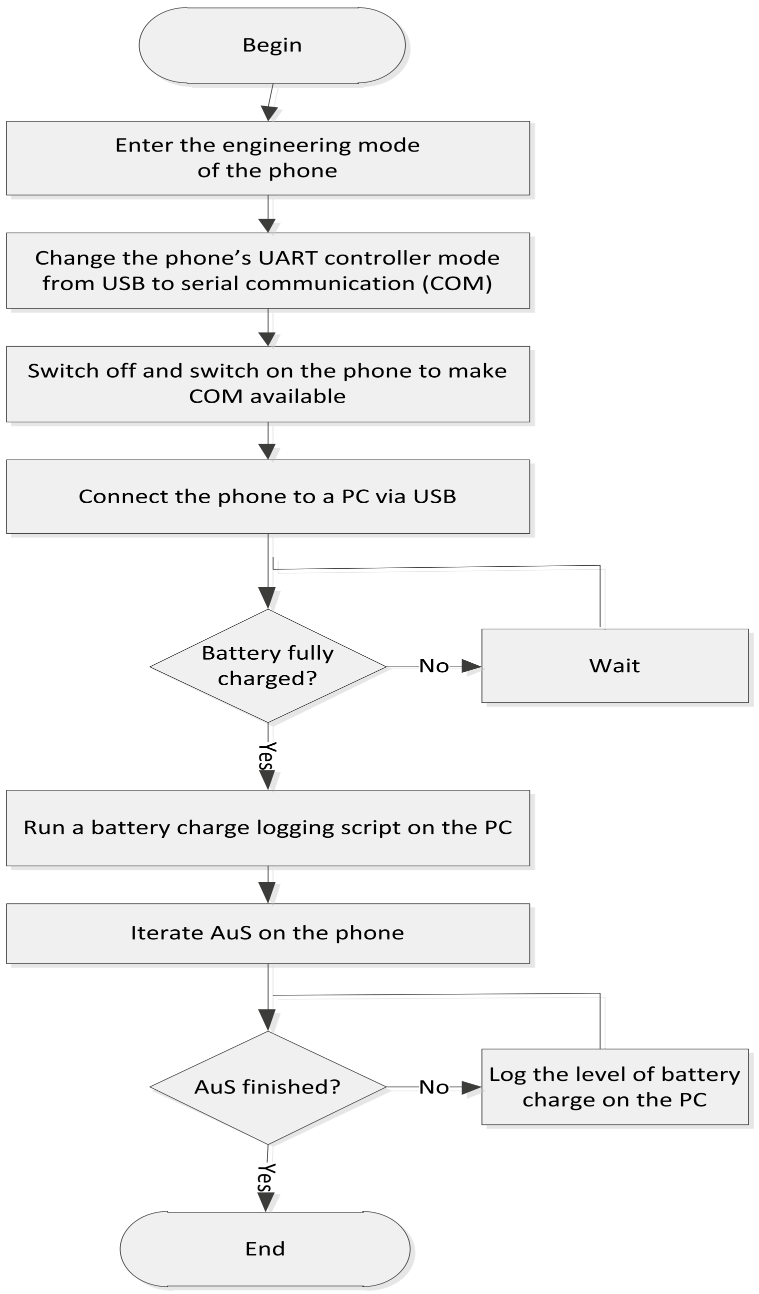
Begin
Enter the engineering mode of the phone
Change the phone's UART controller mode from USB to serial communication (COM)
Switch off and switch on the phone to make COM available
Connect the phone to a PC via USB
Battery fully charged?
No
Wait
Yes
Run a battery charge logging script on the PC
Iterate AuS on the phone
AuS finished?
No
Log the level of battery charge on the PC
Yes
End

Fig. 3. Energy logging algorithm for energy measurement using external software (modem-based) model.

```
String bLevel = System.getProperty("batterylevel");
if (bLevel != null)
   form.append (bLevel);
else
   form.append ("Not supported");
```

Fig. 4. Measurement of the battery charge level in Java application using the System property.

```
SensorManager sm;
SensorInfo[] bInfo = sm.findSensors("battery_charge", null);
SensorConnection sensor =
   (SensorConnection) Connector.open(batteryInfo[0].getUrl());
Data data[] = sensor.getData(1);
int batteryLevel = data[0].getIntValues()[0];
```

Fig. 5. Measurement of the battery charge level in Java application using the JSR 256 API.

The advantage of this methodology is that it is implementation language independent. A snippet of the measurement script implemented in the Perl script language using the Device::Modem module is presented in Fig. 6.

```
$modem = new Device::Modem( port =>'COM2' );
$modem->connect( baudrate => 115200 );
$modem->send_init_string();
$modem->atsend( "AT+CBC\r" );
($ok,@result) = $modem->parse_answer();
$charge = substr(@result[0], 8);
```

Fig. 6. Snippet of battery charge measurement script in Perl.

4. Case studies

First, we present a summary of the energy measurement experiments performed using the methods described in Section 3.

In [23], we have described energy consumption measurement of the computation-intensive DSP algorithms (approximate cosine calculation using look-up tables, specialized FFT, sparse matrix multiplication using the Storage-by-Columns algorithm) implemented in C#. The experiments were performed on the ASUS P750 PDA and on Compaq iPAQ H3900.

In [24], we have investigated the application-level models for the energy-efficient Wi-Fi communication on ASUS P750 PDA using the IEEE 802.11b standard.

In [25], we describe the results of experiments performed with 4 standard cryptography algorithms (AES, RC2, DES and 3DES) aiming to identify their greediness for energy, as well as to collect data for identification of relationships among various characteristics (folder size, information type, security level, algorithm type, encryption/decryption mode, performance/energy, etc.) of the algorithms. In a similar experiment [26], we analyze energy efficiency vs. cipher strength of the AES/Rijndael crypto algorithms in a mobile device with respect to block and key size. The measurements were executed on the ASUS P750 PDA using a custom energy profiling program that implements the algorithm (see Fig. 2) in C# for .NET Compact Framework.

In [27], we analyse energy efficiency vs. quality characteristics of hashing algorithms, and describe methodologies for energy measurement on a Java-enabled SciPhone i9+++ smart phone using the GSM modem-based battery charge measurement method.

In [28], we analyze the approximate calculation methods (Taylor series, Padé rational fractions and Chebyshev polynomials) for improving energy characteristics of the image processing algorithms.

The measurements were performed on a laptop PC with Windows XP Professional OS using the Intel® Application Energy Toolkit (http://software.intel.com/en-us/articles/application-energy-toolkit), which logs total and net power consumption of an application running on a battery-operated system. Table 3 summarizes the results of our previous case studies.

In this paper, we also present the results of energy consumption measurements of 3DMark 06 graphic card benchmarking software. 3DMark 06 is the worldwide standard in advanced 3D game performance benchmarking for Windows that is ideal for measuring the gaming capabilities of laptops, notebooks and tablets, as well as older PCs running Windows XP. The aim of the benchmarking tests is to determine the performance of a computer's 3D graphic rendering and CPU workload processing capabilities by performing standard graphics rendering algorithms such as HDR rendering, shadow mapping and surface rendering.

Table 3. Summary of energy measurement experiments in mobile devices.

Metho-dology	**Method**	**Mobile device**	**OS**	**Software**	**Object of study**
Internal software	Proprietary	Laptop PC	Windows XP Professional	Intel® Application Energy Toolkit	Image transformations [28]
	Custom	ASUS P750 PDA	Windows Mobile 6	Custom C # application	DSP algorithms [23], Wi-Fi communication [24], crypto algorithms [25,26]
		Compaq iPAQ H3900	Windows CE 3.0	Custom C # application	DSP algorithms [23]
	Proprietary	Laptop PC	Windows 7 Ultimate	Intel® Application Energy Toolkit	Graphic card benchmarking tests (this paper)
External Hardware	Java API	SciPhone i9+++	MTK	Custom Java application	Hashing algorithms [28]
	Sensor API			Custom Java application	
	GSM modem			Custom Perl application	

CPU load and battery charge level during the experiment (contains 9 full runs of benchmark tests) are presented in Fig. 7 (standard test) and Fig. 8 (Triangles test) as well as summarized in Fig. 9 (time required to discharge the battery from fully charged to computer stand by). The standard test is a DirectX 9 video card benchmark test. The Triangles test demonstrates graphics card memory throughput by testing how many triangles per second can be drawn. The CPU tests include demanding artificial intelligence, physics and game logic to generate multi-threaded workloads. Fig. 9 shows the energy efficiency of a mobile computer under test to fulfil different computational and graphics rendering tasks.

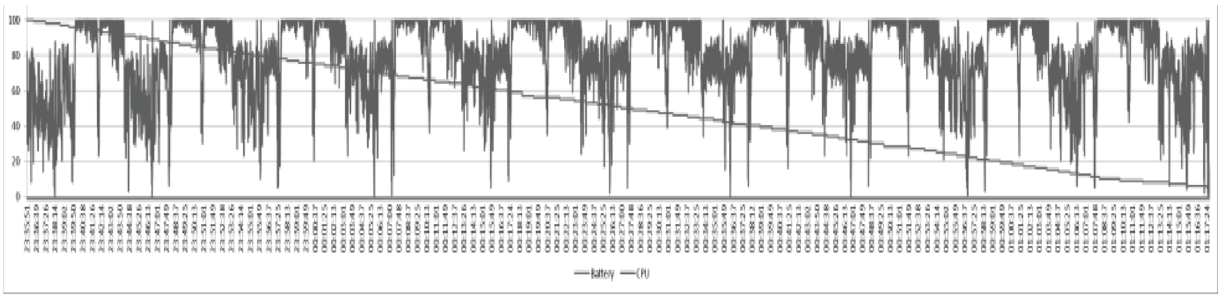

Fig. 7. Results of 3DMark'06 standard test.

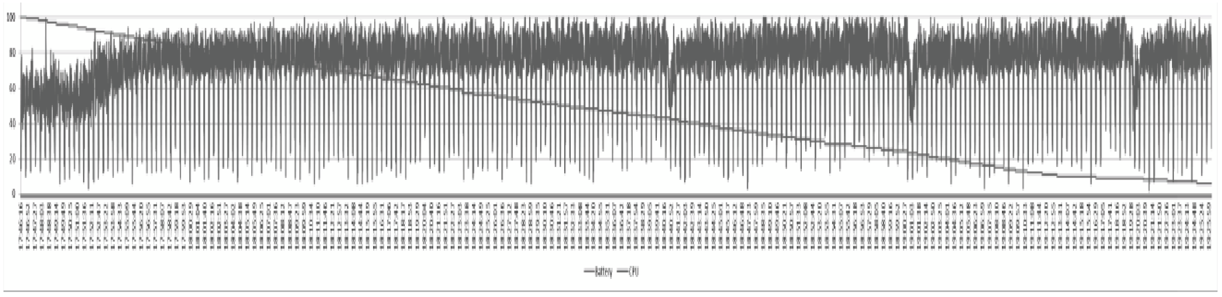

Fig. 8. Results of 3DMark'06 Triangles test.

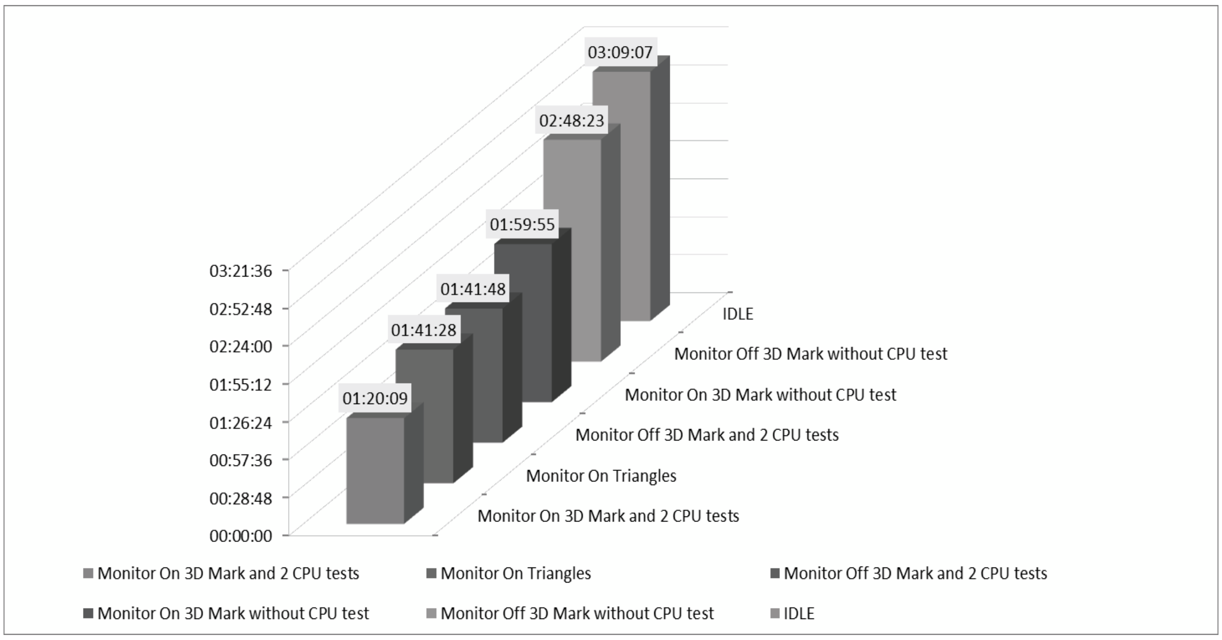

Fig. 9. Summary of energy measurement results using 3DMark'06 benchmarks.

All measurements were performed on a laptop PC (Intel Core Duo 1.73 GHz CPU, 4GB DDR2 RAM) running Windows 7 Ultimate OS (32 bit) using Intel® Application Energy Toolkit. The battery was fully charged and the monitor was switched off.

The accuracy of the external software profiling measurement approach using in this case study can be evaluated on the tool level and measurement level. On the tool level, Intel claims that the accuracy of Application Energy Toolkit (see its documentation) is 5% or better. To establish the accuracy of the approach on the measurement level, we repeated the measurements 8 times using two different laptop computers (Hewlett-Packard F.23, Windows 7 Ultimate 32 bit OS, Intel® Core Duo T2250 1.73 GHz CPU, 4GB DDR2 RAM 265.4 MHz, Mobile Intel® 945 Express Chipset, i945GM 400 MHz GPU with 8MB internal DDR2 RAM and Acer Aspire 1800, Windows 7 Professional 32 bit OS on Intel® Pentium 4 2.93 GHz CPU, 1GB DDR RAM 166MHz, Intel® i915P/i915g chipset, and PA3206U (59Ah, 17V) battery) and obtained relative std. deviation of 12% (for p = 0.95) and standard uncertainty of 4%, assuming normal distribution of measured values.

6. Conclusions

The wide variability of mobile device platforms, operating systems, models, ways to connect to external devices/PC, and APIs is the reason that there is not a single standard method to measure energy consumption of a mobile device. In this paper, we have discussed three different methods to measure energy consumption and overviewed the experiments in energy measurement using the internal (profiling) and external software methods. We have also presented the results of experiments using energy consumption measurement of 3DMark'06 graphical benchmarking tests on a battery-powered laptop.

One of the problems that haunt energy measurement is low accuracy of the existing energy consumption measurement methods (usually of the order of 1% of battery capacity). Another problem is the dynamic nature of the battery's behavior that depends upon many internal and external factors and tends to change over time as the battery undergoes discharge-recharge cycles. While currently none of the analyzed energy measurement methodologies prevails, the future belongs to the standardized APIs, such as the Java JSR 256 Mobile Sensor API, providing necessary data for energy-aware user applications.

Future work will include analysis and comparison of the energy measurement methods and models using benchmark algorithms and applications, and evaluation of the accuracy of the measurement methods with respect to hardware and software properties of mobile computing systems and properties of batteries.

Acknowledgement

The work described in this article has been carried out within the framework of the Operational Programme for the Development of Human Resources 2007-2013 of Lithuania „Strengthening of capacities of researchers and scientists" project VP1-3.1-ŠMM-08-K-01-018 „Research and development of Internet technologies and their infrastructure for smart environments of things and services" (2012-2015), funded by the European Social Fund (ESF).

References

[1] Silven O., Jyrkkä K. (2007). Observations on power-efficiency trends in mobile communication devices. *EURASIP Journal on Embedded Systems,* 2007:056976.

[2] Banerjee N., Rahmati A., Corner M.D., Rollins S., Zhong L. (2007). Users and batteries: interactions and adaptive energy management in mobile systems. In *Proceedings of the 9th International Conference on Ubiquitous Computing (UbiComp '07),* Zurich, Switzerland, 217-234.

[3] Rahmati A., Qian A, Zhong L. (2007). Understanding human-battery interaction on mobile phones. In *Proceedings of the 9th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '07),* Singapore, 265-272.

[4] Krintz C., Wen Y., Wolski R. (2004). Application-level prediction of battery dissipation. In *Symposium on Low Power Electronics and Design (ISLPED'04)*, Newport Beach, CA, USA, 224-229.

[5] Park S. Savvides A., Srivastava M. (2001). Battery capacity measurement and analysis using lithium coin cell battery. In *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED '01*), Huntington Beach, CA, USA, 382-387.

[6] Naik K. (2010). *A survey of software based energy saving methodologies for handheld wireless communication devices.* Technical Report 2010-13, Department of Computer and Electrical Engineering, University of Waterloo, Canada.

[7] Sinha A., Chandrakasan A.P. (2010). Energy aware software. In *Proceedings of 13th International Conference on VLSI Design (VLSI Design 2000),* Calcutta, India, 50-55.

[8] Tan K., Raghunathan A., Lakshminarayana G., Jha N.K. (2001). High-level software energy macro-modeling. In *Proceedings of the 38th Annual Design Automation Conference (DAC '01),* Las Vegas, NV, USA, 605-610.

[9] Palit R., Singh A., Naik K. (2008) Modeling the energy costs of applications on portable devices. In *Proceedings of the 11th International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '08).* ACM, New York, NY, USA, 346-353.

[10] Simunic T., de Micheli G., Benini L., Hans M. (2000). Source code optimization and profiling of energy consumption in embedded systems. In *Proceedings of 13th International Symposium on System Synthesis (ISSS 2000),* Washington, USA, 193-198.

[11] Sagahyroon A. (2006). Power consumption in handheld computers. In *Proceedings of IEEE Asia Pacific Conference on Circuits and Systems (APCCAS 2006),* Singapore, 1721-1724.

[12] Ravi N., Scott J., Iftode L. (2008). Context-aware battery management for mobile phones. In *Proceedings of the 6th Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM '08),* Washington, USA, 224-233.

[13] Krejcar O. (2011). Testing the battery life of mobile phones and PDAs. In *Proceedings of International Conference on Software and Computer Applications (ICSCA 2011).* IPCSIT vol. 9, 132-136. IACSIT Press, Singapore.

[14] Rice A., Hay S. (2010). Decomposing power measurements for mobile devices. In *Proceedings of IEEE International Conference on Pervasive Computing and Communications (PerCom),* Mannheim, Germany, 70-78. IEEE Press.

[15] Mayo R.N., Ranganathan P. (2004). Energy consumption in mobile devices: why future systems need requirements-aware energy scale-down. In *Proceedings of 3rd International Workshop on Power-Aware Computer Systems (PACS 2003),* San Diego, CA, USA. Lecture Notes in Computer Science vol. 3164, 26-40. Springer.

[16] Thiagarajan N., Aggarwal G., Nicoara A., Boneh D., Singh J.P. (2012). Who killed my battery?: analyzing mobile browser energy consumption. In *Proceedings of the 21st International Conference on World Wide Web (WWW '12),* Lyon, France, 41-50.

[17] Nurminen J., Noyranen J. (2008). Energy-consumption in mobile peer-to-peer - quantitative results from file sharing. In *Proceedings of Consumer Communications and Networking Conference (CCNC 2008),* Las Vegas, NV, USA, 729-733.

[18] Balasubramanian N., Balasubramanian A., Venkataramani A. (2009). Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement (IMC '09*), Chicago, Illinois, USA, 280-293.

[19] Xiao Y., Kalyanaraman R.S. Yla-Jaaski A. (2008). Energy consumption of mobile Youtube: quantitative measurement and analysis. In *Proceedings of the 2nd International Conference on Next Generation Mobile Applications, Services, and Technologies (NGMAST '08).* IEEE Computer Society, Washington, DC, USA, 61-69.

[20] Vallina-Rodriguez N., Hui P., Crowcroft J., Rice A. (2010). Exhausting battery statistics: understanding the energy demands on mobile handsets. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Networking, Systems, and Applications on Mobile Handhelds (MobiHeld '10),* New Delhi, India, 9-14.

[21] Flinn J., Satyanarayanan M. (1999). PowerScope: a tool for profiling the energy usage of mobile applications. In *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications (WMCSA '99).* IEEE Computer Society, Washington, DC, USA, 2.

[22] Perrucci G.P., Fitzek F.H.P., Widmer J. (2011). Survey on energy consumption entities on the smartphone platform. In *Proceedings of IEEE 73rd Vehicular Technology Conference (VTC Spring),* Budapest, Hungary, 1-6.

[23] Damaševičius R., Štuikys V., Toldinas J. (2008). Embedded program specialization for multiple criteria trade-offs. *Electronics and Electrical Engineering,* 8(88), 9-14.

[24] Toldinas J., Štuikys V., Damaševičius R., Ziberkas G. (2009). Application-level energy consumption in communication models for handhelds. *Electronics and Electrical Engineering,* 6(94), 73-76.

[25] Toldinas J., Štuikys V., Ziberkas G., Naunikas D. (2010). Power awareness experiment for crypto service-based algorithms. *Electronics and Electrical Engineering,* 5(101), 57-62.

[26] Toldinas J., Štuikys V., Damaševičius R., Ziberkas G., Banionis M. (2011). Energy efficiency comparison with cipher strength of AES and Rijndael cryptographic algorithms in mobile devices. *Electronics and Electrical Engineering,* 2(108), 11-14.

[27] Damaševičius R., Štuikys V., Ziberkas G., Toldinas J. (2012). Energy consumption of hash functions. *Electronics and Electrical Engineering,* 10(18), 81-84.

[28] Damaševičius R., Ziberkas G. (2012). Energy consumption and quality of approximate image transformation. *Electronics and Electrical Engineering,* 4(120), 99-102.