User information Analysis for Energy Consumption Optimization in Mobile Systems

Conference Paper · January 2015

CITATIONS
0

READS
115

4 authors, including:

Smail Niar
University of Valenciennes and Hainaut-Cambresis
143 PUBLICATIONS **379** CITATIONS

SEE PROFILE

User information Analysis for Energy Consumption Optimization in Mobile Systems

Ismat Chaib Draa
LAMIH, University of Valenciennes, France
ismat.chaibdraa@univ-valenciennes.fr

Jamel Tayeb
Intel Corp., Portland, USA
jamel.tayeb@intel.com

Smail Niar
LAMIH, University of Valenciennes, France
smail.niar@univ-valenciennes.fr

Mikael Desertot
LAMIH, University of Valenciennes, France
mikael.desertot@univ-valenciennes.fr

ABSTRACT

Personal and mobile computing systems appear as one of the most important segment of the electronic market and they are rapidly becoming the central device for processing and for communication. Managing energy efficiently is paramount in modern smartphones. With an intelligent energy management, important benefits such as system reliability and augmentation of battery life can be obtained. In these systems, power goes mainly to the processing elements and Input/Output (IO) subsystems like communication peripherals, LCD display and microphone. In this paper, we propose a new approach that exploits user's informations, habits and behaviours to adapt energy consumption. Our approach makes use of Inputs Libraries (ILs) and Actuators Libraries (ALs) in order to manage screen brightness and speaker audio level. Experimental results on Windows 8.1 has shown that with our approach, we save about 30% of the energy consumption of the whole system with a negligible overhead.

Keywords

Mobile Systems, Energy Consumption, Run-Time Users / Application Analysis, Windows 8.1

1. INTRODUCTION

Embedded Sensors, communicating protocols and communication Input/Output augment and are widely present in the new smartphones and tablets generations. These systems provide high computing and proceeding performances, imaging, graphics, etc. They are programmable and contain an important set of powerful embedded sensors such an accelerometer, gyroscope, GPS, microphone and embedded camera. In addition to these embedded sensors, they use and exploit various communication protocols (Bluetooth, WiFi, etc.). All these characteristics, features and components impact significantly the total energy consumption[4].

One of the most used method to save energy in desktop and High Performance Computer Server without impacting negatively functionalities and performance, is to use services which are provided by OS [4] in order to switch off unused devices. In these systems the running applications can control the power consumption because software has a unique understanding on when and which components can be switched on/off, like CPU-GPU Frequency or power capping. However in mobile and communicating systems, hardware features mentioned above are not envisaged. Furthermore, there is behavioural difference with the HPC[1] model where the software has an exclusive use of the platform and with the desktop where the number of active IO devices is fixed and doesn't vary. In addition, the increasing number of application and diversification of their nature.

Our approach consists in the development of a new methodology to reduce and to optimize energy consumption in mobile and embedded systems in order to facilitate energy management provided by OS. The proposed solution is based on the analysis of the interaction between the user and running applications. In this paper we propose the architecture presented in figure 1 which is composed of the **User/Application Interaction Analysis (UAI)** and the **Embedded Software (ES)**.

User/Application Interaction Analysis (UAI): This phase permits us to analyse the interaction between the user and the application. By this tool the main objective is understanding the different hardware and software components utilization at run-time, once for each application and system, statically. This phase is used to verify our hypothesis and develop the Embedded Software (ES).

Embedded Software (ES): The main purpose here is to control at run-time the system components activities which have higher impact on the energy consumption. This infor-

[1]High Performance Computer

mation is combined with user habit informations and its current position that are obtained from the embedded sensors. These informations can be used for an intelligent power/energy consumption optimization. The embedded software in Sec. 3 is represented in our approach by the implementation of Inputs Libraries (ILs) and Actuators Libraries (ALs).

The figure 1 presents the architecture of the proposed framework with **UAI** and the **ES**

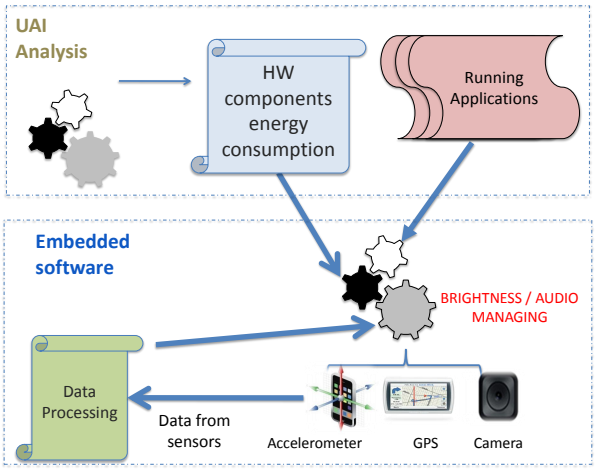

Figure 1: Architecture of the proposed framework

In this paper the idea is that in embedded and mobile systems, it is possible to reduce and save energy consumption by exploiting user behaviour that could be attained by :

1. A set of sensors accessible in mobile device. Sensors are exploited to retrieve data and associate it to user needs, habits and behaviours. Then obtained data are correlated to application possible actions. In this paper, we use principally gyroscope, inclinometer, accelerometer, ambient luminosity sensor and ambient noise sensor.

2. Generally, applications sequences also called scenarios or application patterns are recurrent and correspond to distinct user situations. In the scope of our work, these informations are exploited and manipulated for energy saving. Mobile applications are classified depending on the user behaviours and needs as described in [10] in order to provide a per-user phase to the system.

To resume, the main objective of our work and the proposed framework is to develop a mechanism which learns about user habits and application behaviours in order to collect, store and update the user needs. Then, with the retrieved data, we will impact the hardware and software components with actions like: Application Scheduling, Dynamic Reconfiguration of Hardware and Software and Device managing. (The figure 2 recapitulates the main purpose of our work).

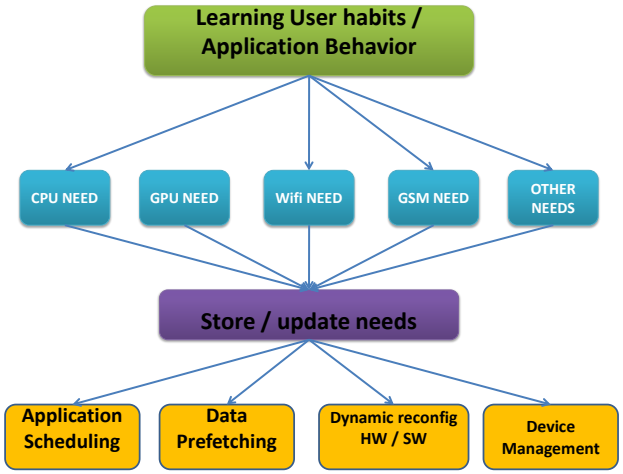

Figure 2: Overview of the proposed approach

The rest of this paper is organized as follows. After a state-of-the-art survey in Sec. 2, we present the used tools for the achievement of the solution in Sec. 3. In Sec. 4, we explain the key idea behind our contribution. The framework architecture and scenarios of utilization will be presented. We also illustrate how our approach based on Intel tools can improve the basic energy management provided by conventional OS. In Sec. 5, experimental results and comparison in terms of consumed energy and execution time between our approach and [4] will be presented. Finally Sec. 6 concludes the work and gives perspectives.

2. STATE OF THE ART

In this section we present the literature concerning energy consumption of mobile systems and we focus on those using user date. In [13], the authors proposed an approach which takes into account the quality of user experience to apply low power techniques. They developed a new *cpufreq* governor based on understanding and analyzing the user-percieved response time of smartphone application during runtime. The developed *cpufreq* gouvernor reduces the CPU energy consumption by up to 65.5% over the Android's default `ondemand` *cpufreq* governor.

In [14] the authors proposed an energy-efficient system for GPS sensor on smartphone using Bayesian networks to decrease the unnecessary usage of GPS receiver at indoors. They exploited built-in sensors in a smartphone to get location information efficiently. Experimental results has shown that it is possible to predict user's location at either indoor and outdoor. The accuracy of their results is higher in weekday than in weekend. This information implies that user's job and lifestyle must be taken into account to develop energy consumption optimization for the GPS.
Our approach differs from [14] by the fact that our analysis takes into account the different embedded sensors and the correlation with running applications.

In [12] the authors demonstrated the adequacy to study real user action and activity to characterize power consump-

tion and to control the development of power optimizations. Their experiments on an HTC ARM based mobile phone expose an important difference in users behaviours. CPU and screen were the most demanding components in terms of energy, for the screen the total utilization time is dominated by a small number of long intervals (100 seconds or more). Our proposed optimization is different to the one proposed by the fact that here, power consumption decreasing is realized by applications behaviour based on facts retrieved from the current user state, earlier runs and geographical position.

In [8], mobile system user utilizations were analysed. Authors deducted that applications are responsible of half of the traffic in theirs experiments. They propose optimization solution of the power consumption for the radio and they reduced it by 35% with minimal impact on the performance of packet exchanges. They also indicated that optimizations and mechanisms could not work for a large fraction of users, in preference learning and adapting to user behaviours is likely to be more adequate. The conclusion was that effective adaptation would require future platforms to support light-weight tools that guide and learn user behaviours as demonstrated in [9].

Other works, like [6], demonstrated the efficiency of using embedded sensors to optimize resource utilization. In this work the authors tried to optimize wireless procedures by using sensors to exploit users behaviours. Our paper is different from [6] as here we focus on the energy consumption of the whole system.

In [4], the authors used the Intel [1] Energy Server (ESRV). ESRV is a universal tool to measure energy consumption for mobile and embedded systems. They used Context Aware Mobile Services (CAMS) [7], a framework which is able to adapt services in order to optimize and reduce power consumption. The main purpose of the paper was to allow ESRV to use context data thanks to CAMS.
In [4] services were adapted to the user's environment and depend for example of the user location, the device position or the CPU need. By exploiting these informations and using adaptive rules, the purpose was to reduce energy consumption. Data were captured via the sensors and retrieved, registered and processed in CAMS. After the registration phase, services are monitored in order to detect possible optimizations scenarios. When such scenario is possible, CAMS takes the necessary actions to impact the hardware and the software to optimize power and energy consumption. The solution in [4] was updatable at run-time with a dynamic behaviour.

The main drawback in [4] is performance degradation due important overheads in execution time and energy. These overheads are due to the use of high level technologies like Java Native Interface (JNI) and different programming languages. In fact ESRV was developed in C/C++ and CAMS was developed in JAVA. The communication between both components was achieved by JNI. This structure make the framework utilisation costly.

The proposed approach in this paper, is based on the use of ESRV and the Intel Modeler [1] tools but without exploiting a high level framework like CAMS. Our approach simplifies the utilisation of services The energy overhead and the increasing of the latency cost resulting in [4] are also optimized.

3. ESRV AND THE MODELER TOOLS

This section presents the used tools components, their usage and the global architecture.

3.1 Key Component List

In this paper we use two principal components: ESRV and the Intel Modeler (called here the Modeler). ESRV is the main driver of the tool where all user interactions are channelled. The Intel Modeler is a device support module for ESRV used as the kernel's tool. It provides basic services and offers multiple expansion mechanisms exposed to developers. Expansion modules can be developed toward these interfaces and used with the Modeler. In our case, expansion modules are mainly Inputs Libraries (ILs) and Actuators Libraries (ALs).

3.1.1 Energy Server (ESRV)

The Energy Server (ESRV) is a major component of the Intel Energy Checker Software Development Kit (iECSDK) [1]. The iECSDK has been designed to help measuring application energy efficiency. Energy efficiency is defined as the ratio of the amount of useful work done by the software and the energy consumed by the system to run the application. ESRV provides counters for the cumulative energy consumed, as well as the immediate power draw from the power meter or power supply. Counters store the number of times a particular events or process has occurred.

Measuring energy in a seamless way to the users requires to communicate with a broad range of hardware devices. By handling for the user various communication interfaces and protocols and by exposing the power and energy metrics in a standard and unified way, ESRV simplifies the access to this important components of the energy efficiency.

3.1.2 Intel Modeler

The Intel Modeler is a device support module for ESRV. It was created to design and run platform specific power models. A power model is a set of equations that describes the power consumption of a platform. With such power model, it is possible to accurately estimate the energy consumption of a system without the need of a measurement device. The Modeler is mainly composed of three components: the Front-End, the Input Bus (IB), and the Back-End.
The **front-end** is responsible of collecting data or input's values. The expansion of the front-end is done by **Inputs Libraries**.
The **Input Bus** can be considered as a shared memory between the Front-end and the Back-end components.
The **Back-end** provides basic services such a logger or a power-to-inputs automatic correlation, it is also modular and the expansion is provided by **Actuators Libraries**.

a) Input Libraries (ILs): ILs were developed to expand the collection Front-end. In all cases, data flow within the Modeler starts with inputs. Inputs are provided by Input libraries. ILs are responsible of collecting a well-defined

set of data. There are 2 types of ILs, based on their inputs nature, static when the inputs count is fixed and dynamic when the inputs count can vary over time.

There are 4 essential phases for the creation of an Input Library: the **Initialisation**, the **opening**, the **reading** and the **closing** phase. At the initialization phase, the inputs number and the IL type are defined. In the opening phase, the initialization of inputs is realized with inputs names, and inputs types. We distinguish 4 input types which are ULL (Unsigned Long Long), DOUBLE, STRING, and finally BLOB (Binary Large Object which can be composed of any other types). The opening phase permits to set and read inputs values. Finally the closing phase is used to de-allocate allocated resources in the initialization or opening phases.

The inputs sampled by the ILs are feed into the Modeler's Input Bus (IB). Once on the IB, the data is available to any back-end component. On arrival onto the IB, inputs are automatically renamed by getting an alias, this alias makes it easier to reference inputs in back-end modules. The alias of an input is derived from the IL name (as defined by its designer) and the rank of the input (among the total number of provided inputs by the IL). Inputs are visible in the IB as a vector, for example for an IL named X, inputs are visible as: X(0), X(1), ..., X(N).

b) Actuators Libraries (ALs): ALs are the counterpart of ILs. They are implemented and behaves similarly to ILs. Actuators were designed to perform specific actions, including system-level configurations. They can be used to perform platform power optimizations by applying business logic, built upon inputs available on the Input Bus thanks to ILs. Actuators can also be used to alter dynamically several Intel Modeler behaviour. Figure 3 shows the architecture and interactions between ESRV and the Intel Modeler's tools.

These tools were developed by Intel to monitor the OS and to implement communication between different components in a mobile and embedded systems. In this paper, we introduce the utilization of ILs and ALs for a new service. It concerns energy optimization. ILs are used to retrieve sensors data and ALs to manage devices like screen, Wi-Fi microphone, etc. In our framework, ILs and ALs implement the **the Embedded Software (ES)** component mentioned in Sec. 1.

The main purpose of our work is to offer to the user in different scenarios a better energy management than the standard energy management proposed by the OS.

4. ENERGY CONSUMPTION OPTIMIZATION USING ILS AND ALS

In this section, we present our methodology with the architecture and the communication system between components. The approach is mainly based on the tools described in Sec. 3. In fact, to avoid the energy, time execution overheads and in order to have an unified solution with the same abstraction level, a high level framework like CAMS [7] is not used. We centred our approach on the expansion of the Intel Modeler with the ILs and ALs. The proposed concept manages the screen brightness and the speaker sound

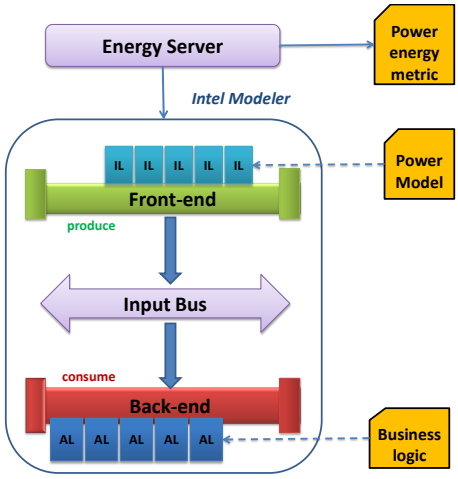

Figure 3: Interaction between the components

level. To do so, 2 ILs and 2 ALs were developed: The **Environment Sensors Library** , **Sound Input Library**, **Brightness Actuator** and **Sound Actuator**.

4.1 Environmental Sensors Library

This IL is responsible of collecting data from the embedded sensors in order to identify the most appropriate device position for the user. The Environmental Sensors Library interfaces with the embedded sensors which contain informations about the mobile device. These sensors are : **Gyroscope** and **Accelerometer** for the motion, **Inclinometer** for the orientation and **Ambient Luminosity sensor (ALS)**. Each sensor produces three dimensional values in X,Y,Z axis. Sensors names are registered in the IL. Then, standard sensors values when the device is in normal stand (i.e. it is in suitable position to the user) are obtained via Sensor Diagnostic Tool [3] and are stored at the initialisation of the the Environmental Sensors Library. After retrieving the sensors values, they are registered and indexed in the IL Import/Export macros. For each sensor dimensional value, an index is associated in order to store its value, for example for the inclinometer, we associate three indexes INC-X, INC-Y, INC-Z, one for each dimensional value. Later, whenever the device is in inappropriate position for the user (inclined, tilted, etc.) or the ambient luminosity is high, the new sensors values are collected via threads and updated in the **Environmental Sensors Library**. Finally all retrieved data are injected in the Input Bus(IB) in order to be consumed by our actuator (Brightness Actuator).

4.2 Brightness Actuator

The Brightness Actuator is developed to manage luminosity of the screen. The actuator handles the LCD driver of the device in order to apply business logic on the screen brightness values depending on the retrieved sensors values form the Input Bus (IB)(values which are available thanks to **Environmental Sensor Library**).
Whenever the values are updated in the IL, they are immediately injected in the IB and retrieved by the AL. The Brightness Actuator compares the new retrieved sensors values with the sensors values when the device is in normal stand. Then, according to the comparison result, the ac-

tuator manages the brightness. For example if one of the inclinometer dimensional values exceed the range of allowed values, for example if the INC-X is outside the range of authorized values (Sensors values in normal stand), the AL applies a specific optimization on the screen's brightness by decreasing it. The figure 4 shows the monitoring scenario.

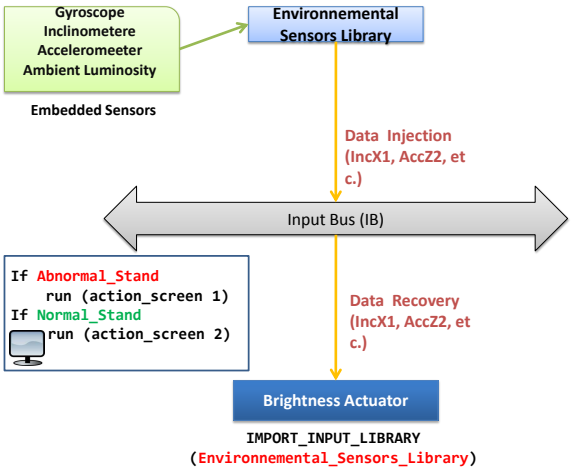

Figure 4: Screen Brightness Management

4.3 Audio Input Library and Sound Actuator

Audio Input Library and Sound Actuator were developed in order to manage audio speaker. The Audio Input Library retrieves and updates the ambient noise value and injects it on the Input Bus, then the ambient noise values are imported. The Sound Actuator handles the speaker driver and manages the sound depending on the ambient noise level. For example in this scenario when the ambient noise is at high level, the AL increases the speaker's sound level. On the other hand, when the ambient noise is at average or low level, the AL adapts the speaker's sound level. Figure 5 shows the monitoring scenario for audio managing.

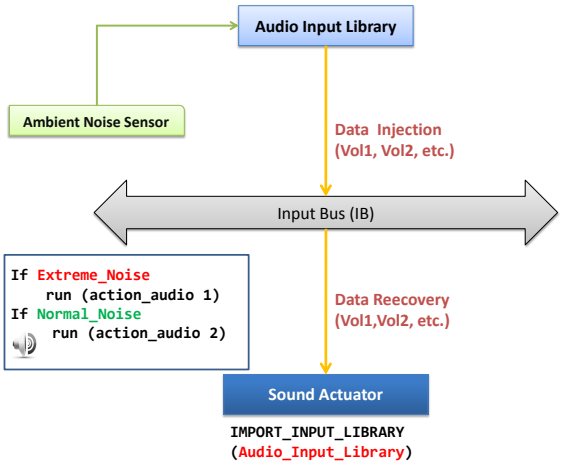

Figure 5: Speaker Audio Management

The Environmental Sensors Library, Brightness Actuator, Audio Input Library and the Sound Actuator permit us to propose an energy-efficient system for the whole mobile system. The developed ILs and ALs enable the user to have an important gain in terms of energy consumption in specific scenarios in comparison with the energy management proposed by the OS.

With the proposed solution we have 3 phases which are, the **Deployment and registration** phase, the **Collection Phase** and the **Action phase**. The deployment and registration phase represents the solution loading, and the sensors names registration. In the collection phase, ILs retrieve sensors values and transmit it in the IB. In the last phase, ALs import sensors values and manage devices. In Sec. 5, we will see different time intervals between the phases in order to demonstrate the response time of our solution.

With the developed ILs and ALs to manage brightness and audio, our approach reduces execution time and energy consumption without the energy overhead obtained in [4]. Our solution is repetitive and is transparent to the user. The user will have the choice to use each IL and AL depending on his needs.

The solution can be loaded and unloaded in the OS Kernel upon demand. The energy overhead resulting form our developed ILs and ALs is low and does not impact negatively the system with performance degradation like it will be shown in Sec. 5. The provided benefits are due to a smart brightness and audio managing in specific user sequences and the utilisation of the Intel tools without using high level technologies like JNI. The following section presents some experimental results, comparison between the solution developed in [4] and our solution with developed ILs and ALs and our solution cost in terms of CPU usage, time, physical memory(RAM) and power consumption .

5. EXPERIMENTAL RESULTS

As mentioned above, the purpose of this work is to improve the energy management proposed by the OS, for this we used the tools mentioned in Sec. 3.

In order to validate our approach and illustrate its benefits, we take into account two studies. The first one has been elaborated in [11] about the consumed energy by some common sensors of the system. We also use thise set of sensors. The study demonstrated that energy consumption of embedded sensors in mobile systems represent less than 1% of the whole system energy consumption, this information permit us to exploit sensors mentioned in Sec. 4 without impacting the energy consumption of our whole mobile system. In [5], the authors proved that it is possible to save more than 30% of the energy consumption in mobile systems with an intelligent management of the brightness, WiFi and speaker. Our work uses the same assumption.

In the first part of this section, the evolution of energy consumption by using our ILs and ALs is shown. We demonstrate the gain in terms of energy consumption by using our solution in comparison with the energy consumed by the OS, Windows 8.1 in our case, without our optimization.

In the second part, we also compare between approach achieved in [4] and our approach with ILs and ALs in terms of energy overhead and latency. The solution cost is presented in the third part of this section.

The experimentations have been realized on an Ultrabook running Windows 8.1 with a 2.50Ghz Intel dual-core i7-3667u processor and 4GB of RAM. Energy measurement has been experimented using a power meter Yokogawa WT210 [2].

5.1 Energy Evolution With ILs and ALs

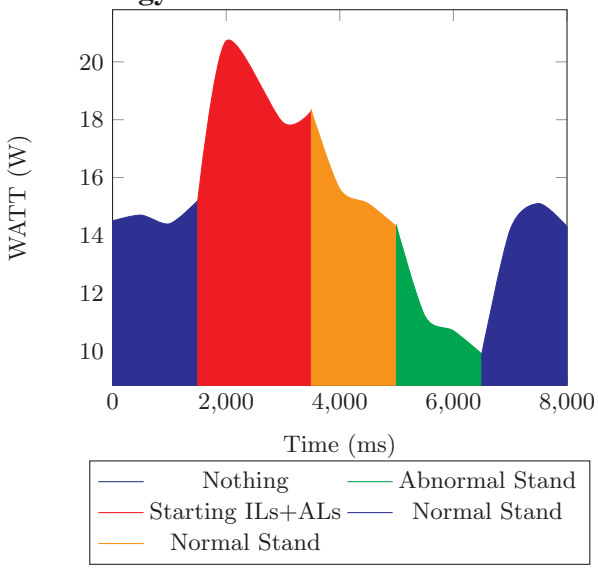

Figure 6: Energy evolution with ILs+ALs

The goal of this part of our experiments is to measure the consumed energy by the Ultrabook when Environmental Sensors Library and Brightness Actuator are running.

In order to have the most accurate measurement, we center on the brightness parameter, like shown in figure 6.

In this figure, we have 5 phases. The first phase, in blue, indicates the energy consumption before the deployment of the solution. In the second part, in red, we note an energy overhead which is due to the deployment of ILs and ALs and the sensors registration. The orange part represents the power consumption when the Ultrabook is in a suitable position for the user and power consumed is 14.7 Watt, i.e. the screen brightness increased. In the green portion, we notice a decreasing of the energy consumption which is due to the non utilisation of the Ultrabook, thus the Brightness Actuator decreases brightness level which implies a gain in terms of energy. The system consumes in this phase 9.87 Watt. Finally, in the last blue part, the brightness level is comes to its maximale value because the device is back in normal stand for the user.

In this graph, the principal information is the gain obtained in terms of energy consumption when the Ultrabook is in unsuitable position for the user. These gains are due to the deployment of the IL and AL for the screen brightness and speaker sound level managing, the gain is valued at 28.7% in comparison with the Windows 8.1 energy managing.

5.2 ILs and ALs overhead measurements

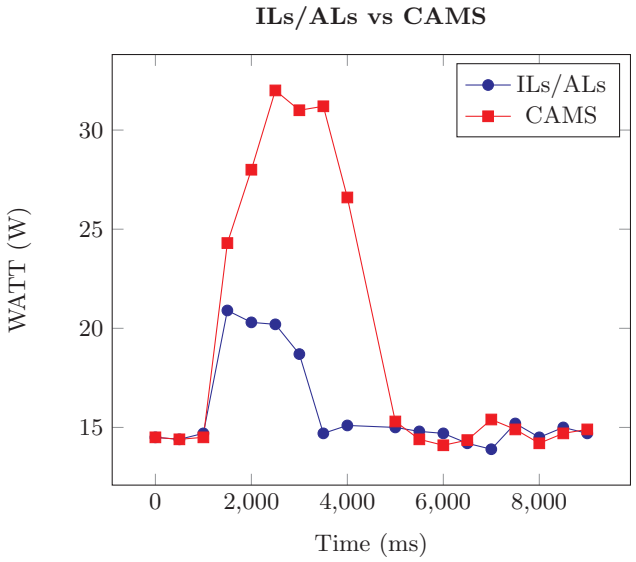

Figure 7: Comparison between the both approach

In figure 7, we compare the results obtained in [4] and results with our approach in terms of energy consumption and time overhead. The goal of these experiments is to measure deployment energy overheads of ESRV with ILs and ALs in one side and ESRV combined with CAMS in the other side. We measure the energy consumed at the starting point in order to compute the additional cost when starting the both approaches.

During the test, no application is running, only solutions are deployed. At the starting point before the deployment of our applications, the energy consumed is stable.

At 1700ms, we deploy the both approach. With with CAMS the energy overhead is estimated at 32W, while with our approach the energy overhead is estimated at 21W.
To resume the energy overhead is more significant in the solution developed in [4] in comparison with our approach. By deploying Environmental Sensors Library and Brightness Actuator in the OS, the gain of the consumed energy is evaluated at 29.2 % in comparison with the solution based on CAMS.

We also note that with our solution, the energy consumption is stabilized at 3100 ms, and the solution with CAMS is regulated at 5000 ms. This difference provided us a benefit in terms of latency which is valued at 22%.

5.3 Overhead Evaluation

In this subsection, the solution total cost is given in terms of CPU usage, physical memory , power consumption of the whole system and time duration of the different phases. To do so, we developed functions based on Windows API which are used as probe for the system resources. The probes was

implemented in the ILs and ALs in the different phases. In order to ensure the accuracy of the results provided by the probes, we also exploit the informations provided by the Windows Task Manager. These results are presented in order to demonstrate that our proposed solution based on ILs and ALs does not affect system and does not cause performance degradation for the user, rather it allows the user to save his mobile system energy consumption in comparison with what it is proposed by the OS.

Table 1: **Ils and ALs Cost**

Phases	Power (W)	CPU (%)	RAM (MB)	TIME (ms)
1	20.7	0.3	9.74	2000
2	16.8	0.3	10.45	1500
3	14.7 $\rightarrow$ 9.8	0.4 $\rightarrow$ 0.7	10.04	300
4	9.8 $\rightarrow$ 14.7	0.6 $\rightarrow$ 1.1	10.05	300
5	14.7	[0.2 - 0.6]	7.2	- - - -

In the table 1, we have 5 phases. For each one, we measure the power consumption of the whole system, the CPU usage, the physical memory consumption and the duration time. The first phase represents the deployment and the registration of the solution. In this phase we have a peak of 20.7 W in terms of power consumption, knowing that the average power consumption of the system is 14.7 W, the first phase provides to the user a power overhead of 6 W. This phase lasts for 2 seconds.

The phase 2 represents the collection of sensors vales, the power consumption in this phase is valued at 16.8 W which implies a power overhead of 2.1 W which lasts 1.5 seconds. The CPU usage in the two first phases is 0.3 % and the physical memory consumption varies between 9.74 MB and 10.45 MB.

Phases 3 and 4 represent the device management phases, the phase 3 is when the Ultrabook passes from usable state to unusable state, which implies a specific energy management. In this phase, we have a power decreasing from 14.7 W to 9.8 W which gives to the user a gain of 4.8 W whenever the Ultrabook can not be used. The CPU usage in this phase changes from 0.4% to 0.7%, this variation is due to brightness and sound decreasing performed by ALs.
The phase 4 is the counterpart of the phase 3 and represents the device management action when the Ultrabook switches from the unusable state to the usable state. In this phase we have a power increasing from 9.8 W to 14.7 W (Brightness and Sound Restoring)and augmentation in CPU usage from 0.6% to 1.1%. The action phases (3 and 4) last 300 ms and consume about 10.05 MB in terms of physical memory.

The last phase represents the system in normal stand with our running solution. The power consumption is valued at 14.7 WATT and CPU usage varies between 0.2% and 0.6%. The physical memory consumption is 7.2 MB.

These measures confirm that our solution is not greedy in terms of CPU, RAM and the whole system energy consumption is slightly affected by our solution at the deployment phase.

The provided benefits by our approach are due to the efficient utilization of ILs and ALs. Environmental Input Library, Audio Input Library, Brightness Actuator and Sound Actuator were developed in language C. The Intel Modeler and the Energy server are also developed in C/C++. The communication between all these components has been achieved without high level technologies which impact the performance and the energy like JNI and Java Virtual Machine. Therefore, the communication and synchronisation between the ESRV, ILs and ALs are less demanding in terms of energy, CPU and RAM than the communication between CAMS and ESRV.

6. CONCLUSION AND PERSPECTIVES

In this paper, we have proposed an energy-efficient system using the Energy Server, Inputs Libraries and Actuators Libraries, tools which have been developed by Intel in order to monitor mobile and embedded systems. This system can offer user to control screen brightness and speaker sound dynamically. In existing methods of energy consumption optimization, the users needs and behaviours are rarely taken into account. In this paper we have presented a new methodology to optimize energy consumption of the mobile systems by the analyse of the interaction between the user and applications. With our solution, we propose to the user a gain in terms of energy consumption in comparison with the energy management provided by windows 8.1 in some scenarios, the gain is evaluated at 30%.

Our approach is based on Intel tools with the introduction of Input Libraries which was used to collecting data and Actuators Libraries to control directly different hardware components. In comparison with the approach developed in [4], we have a gain of 27 % in terms of energy consumption at the deployment phase. The latency is also reduced by 22%.

In the next step of the project we will consider more possible user patterns. In addition, more sensors, such as compass and GPS will be used. New heuristic will be developed in ALs to control different sensors and Input/Output peripherals. Another important task is the deployment of power consumption models for estimating running and future predicted applications in ILs and ALs. Finally we will expand the use of IL and AL for other mobile platforms such as smartphones and tablets under Android and iOS.

7. ACKNOWLEDGEMENT

The Authors would like to thank Intel Corporation and especially the Intel Research Council for the support given to the project and the tools.

8. REFERENCES

[1] Intel energy checker software development kit userguide https://software.intel.com/en-us/articles/intel-energy-checker-sdk.

[2] Power meter yokogawa wt210. http://www.electro-meters.com/yokogawa/yokogawa-power-meters/wt210/.

[3] Sensor diagnostic tool. http://msdn.microsoft.com/en-us/library/windows/hardware/hh780319

[4] O. Carlier, J. Tayeb, M. Desertot, S. Lecomte, and S. Niar. Run-time users/applications interaction analysis for power consumption optimization. In *Energy Aware Computing Systems and Applications (ICEAC), Annual International Conference on*, pages 181–186, Dec 2013.

[5] A. Carroll and G. Heiser. An analysis of power consumption in a smartphone. In *USENIX annual technical conference*, pages 271–285, 2012.

[6] D. Corujo, M. Lebre, D. Gomes, and R. L. Aguiar. Sensor context information for energy-efficient optimization of wireless procedures. In *International Symposium onPersonal Indoor and Mobile Radio Communications (PIMRC)*, pages 1010–1014. IEEE, 2011.

[7] M. Desertot, S. Lecomte, D. Popovici, M. Thilliez, and T. Delot. A context aware framework for services management in the transportation domain. In *Annual International Conference on New Technologies of Distributed Systems (NOTERE)*, pages 157–164. IEEE, 2010.

[8] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin. A first look at traffic on smartphones. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, pages 281–287, 2010.

[9] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin. Diversity in smartphone usage. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pages 179–194, 2010.

[10] N. Goswami, R. Shankar, M. Joshi, and T. Li. Exploring gpgpu workloads: Characterization methodology, analysis and microarchitecture evaluation implications. In *IEEE International Symposium on Workload Characterization (IISWC)*, Dec 2010.

[11] I. König, A. Q. Memon, and K. David. Energy consumption of the sensors of smartphones. In *Wireless Communication Systems (ISWCS 2013), Proceedings of the Tenth International Symposium on*, pages 1–5. VDE, 2013.

[12] A. Shye, B. Scholbrock, G. Memik, and P. A. Dinda. Characterizing and modeling user activity on smartphones: Summary. In *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '10, pages 375–376, New York, NY, USA, 2010. ACM.

[13] W. Song, N. Sung, B.-G. Chun, and J. Kim. Reducing energy consumption of smartphones using user-perceived response time analysis. In *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, HotMobile '14, pages 20:1–20:6, New York, NY, USA, 2014. ACM.

[14] S.-H. Yi and S.-B. Cho. A battery-aware energy-efficient android phone with bayesian networks. In *Ubiquitous Intelligence Computing and 9th International Conference on Autonomic Trusted Computing (UIC/ATC), 2012 9th International Conference on*, pages 204–209, Sept 2012.