Method for Optimizing Energy Consumption using Context Awareness per Application on Mobile Devices

José Carlos Valdivia Bedregal
APOYO TOTAL

Eveling Gloria Castro Gutierrez
CICA-UCSM

Robert Arisaca
MICRODATA S.R.L.

Abstract

Mobile devices are powered by batteries whose capacities are limited by size, so efficient management of energy becomes a very important issue in such devices. Many solutions have been proposed seeking to extend mobile device's battery life, but there are only a few who takes the user as a decisive and determining factor. Only the user determines how to consume battery's energy. Therefore the proposed application aims to solve this issue by learning about user's context. By analyzing this information, automated actions are performed in order to optimize energy consumption.

1. Introduction

Today's society has become an interconnected society; having timely information gives the user several advantages. Mobile devices provide opportunities when quick and updated information is needed, a trend that is becoming a necessity in our society.

Besides considering the many benefits they bring, let’s consider the limitations. Mobile devices, including cell phones, are powered by batteries that are limited by size. This situation limits the ability of fully enjoying the benefits of the device. Correct management of energy stored in the battery should be considered as an issue of utmost importance in such devices.

Efficient energy management requires an understanding of where and how energy is used [1]. To this end we present a detailed analysis of energy consumption in a Samsung Galaxy Nexus Smartphone.

Within the analysis end user must be consider an important factor in addition to energy consumption as end user is commonly overlooked. Ultimately, the power consumption of a mobile device is defined by the user activity.

Section II describes related work. Section III describes “Power Tutor” tool. Section IV provides an analysis of applications that consume more energy and the components responsible of this consumption. To perform this analysis, we used the PowerTutor tool [2], which provided further information on the components with higher energy consumption per-application. Section VI shows the results obtained from the analysis and also a classification of the applications is presented in Section V. Finally, in section VII, necessary measures are implemented to achieve energy savings.

2. Related Work

Lide Zhang [2], describes PowerBooter, as a modeling technique that uses automated voltage sensors to monitor the power consumption of different components in the device. It also describes PowerTutor as a tool that analyzes the energy management and uses the model generated by PowerBooter to estimate real-time energy consumption.

Aaron Carroll [3] describes energy consumption of the device according to major hardware components. Description for micro-benchmarks as well as a number of realistic usage scenarios. These results are validated by overall power measurements from other two devices: the HTC Dream and Google Nexus One.

Alex Shye [4], in his paper studies mobile architectures in their natural environment, developing a logger application for Android G1 and seeking to record the information in a real user activity environment. Also presents a regression model to estimate the power consumed based on collecting measurements easily accessible by the logger. This model accurately estimates power consumption and also provides detailed information on the distribution of power between hardware components.

Fangwei Ding, Feng Xia[5], developed an intelligent system for monitoring energy consumption called SEMO for Smartphones that use Android operating system.

3. About Power Tutor

PowerTutor is an application for Android devices that displays the amount of energy consumed by major system components such as CPU, network interface, display, GPS receiver and other applications. The application allows software developers to understand the impact of design changes in the efficiency of energy use. Users of the application can also get used to determine how their actions affect battery life.

PowerTutor builds a model of energy consumption by direct measurements. These measurements are caught during a time of detailed control of the devices that deliver energy. The model provides an estimation of energy consumption with an estimate error of 5% approximately from actual values. In this investigation the error may increase as unspecified constants have been used for the Samsung Galaxy Nexus. Estimates are given through a configurable display that provides history of energy consumption. It also shows users textual output that contains detailed results. Additionally, PowerTutor can be used to monitor power consumption of any application.

Table 1: Cell phone Tecnical Especifications

Device	Samsung Galaxy Nexus
	GT-i9250
Operating System	Android 4.2.1
Processor	1.20GHz TI OMAP 4460 (ARM Cortex A9 + PowerVR SGX540)
RAM	1 Gb
Display	Oled 4.65"
Resolution	1280x720
Battery Capacity	1750 mAh
Connections	Wi-Fi
	3G (HSPA+ 21)

PowerTutor was developed by Ph.D. candidates University of Michigan Mark Gordon, and Lide Zhang Birjodh Tiwana under the direction of Robert Dick and Zhuoqing Morley Mao at the University of Michigan and Lei Yang from Google under the direction of Professor Theodore Baker. He has received previous support from Google and the National Science Foundation under Grant CNS-0720691, and was done in collaboration with the University of Michigan and Northwestern University in the framework of Empathic Systems Project.

4. First Approach

The first approach of this solution was meant to be a static approach to realize how it's possible to reduce energy consumption. After that, a complete solution was developed based on the knowledge taken from this first approach.

4.1. Energy Consumption Analysis

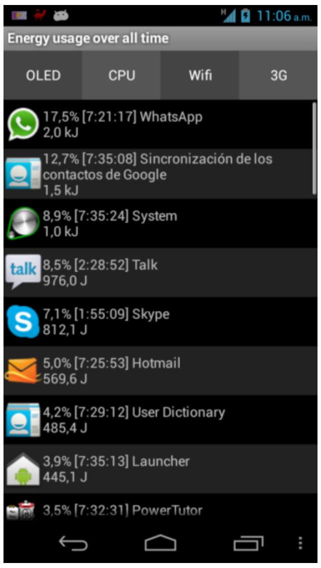
Using PowerTutor, consumption tests were taken on a Samsung Galaxy Nexus. The tests were conducted considering a common use by the test user for about7.5 hours.

Consumption measurements that have been taken are not physical as done in [3]. Aaron Carroll has more accurate measurements but they aren't very dynamic and condemn the cell to be connected to external devices.

Top consumer applications with higher energy consumption in sum of all of the cellphone's components and the overall results are presented in Fig. 1.

Fig. 1presents that the application with higher energy consumption is: WhatsApp, an internet based messaging application. Subsequently there are system services and processes: Google Automatic synchronization, through which the cell phone synchronizes contacts, calendar with Google's servers in the cloud and the Android OS.

Continuing, there are two communication applications: GoogleTalk and Skype. Then it shows an e-mail client: Hotmail. Finally other two processes are observed system, the dictionary and the home screen.



Energy usage over all time

17,5% [7:21:17] WhatsApp
2,0 kJ

12,7% [7:35:08] Sincronización de los contactos de Google
1,5 kJ

8,9% [7:35:24] System
1,0 kJ

8,5% [2:28:52] Talk
976,0 J

7,1% [1:55:09] Skype
812,1 J

5,0% [7:25:53] Hotmail
569,6 J

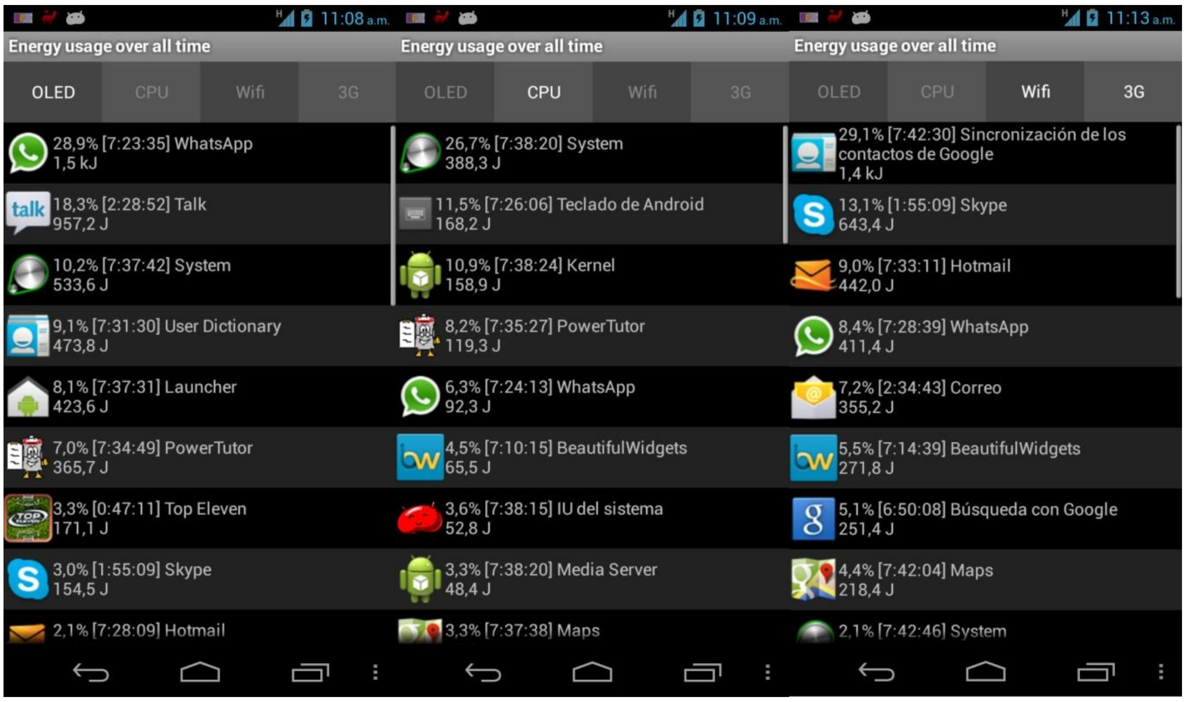
4,2% [7:29:12] User Dictionary
485,4 J

3,9% [7:35:13] Launcher
445,1 J

3,5% [7:32:31] PowerTutor

Figure 1: Overall Results

A deeper analysis by components for each application is shown in Fig. 2. Having WhatsApp application as a reference, it shows that there is an increased consumption in the OLED screen, which has a 75% of the total battery drained by the by the application. Next consumption is the internet communication, whether using Wi-Fi or 3G, it represents 20% of total consumption and finally the remaining 5% is CPU consumption.


11:08 a.m.
Energy usage over all time
OLED | CPU | Wifi | 3G
28,9% [7:23:35] WhatsApp 1,5 kJ
18,3% [2:28:52] Talk 957,2 J
10,2% [7:37:42] System 533,6 J
9,1% [7:31:30] User Dictionary 473,8 J
8,1% [7:37:31] Launcher 423,6 J
7,0% [7:34:49] PowerTutor 365,7 J
3,3% [0:47:11] Top Eleven 171,1 J
3,0% [1:55:09] Skype 154,5 J
2,1% [7:28:09] Hotmail

11:09 a.m.
Energy usage over all time
OLED | CPU | Wifi | 3G
26,7% [7:38:20] System 388,3 J
11,5% [7:26:06] Teclado de Android 168,2 J
10,9% [7:38:24] Kernel 158,9 J
8,2% [7:35:27] PowerTutor 119,3 J
6,3% [7:24:13] WhatsApp 92,3 J
4,5% [7:10:15] BeautifulWidgets 65,5 J
3,6% [7:38:15] IU del sistema 52,8 J
3,3% [7:38:20] Media Server 48,4 J
3,3% [7:37:38] Maps

11:13 a.m.
Energy usage over all time
OLED | CPU | Wifi | 3G
29,1% [7:42:30] Sincronización de los contactos de Google 1,4 kJ
13,1% [1:55:09] Skype 643,4 J
9,0% [7:33:11] Hotmail 442,0 J
8,4% [7:28:39] WhatsApp 411,4 J
7,2% [2:34:43] Correo 355,2 J
5,5% [7:14:39] BeautifulWidgets 271,8 J
5,1% [6:50:08] Búsqueda con Google 251,4 J
4,4% [7:42:04] Maps 218,4 J
2,1% [7:42:46] System

Figure 2: Per component results

In Google Talk, there are similar consumption patterns; screen consumption represents 90%, communication consumption represents 6% and only 2% is CPU consumption.

To confirm this behavior, there is Skype, which remained longer at background and doesn't have the same screen consumption; only 19%, but internet communications recorded an 80%.

4.2. Classifying applications

In the means of classifying applications it has been examined [3] and based on their classification it was decided to create four classifications that will be used to implement future actions:

Table 1: Application classifications Especifications

System Applications	User Applications
	CPU Applications
	Comunication Applications

4.2.1. System Applications

They are Android System's Applications, to which access or modification isn't convenient as this can result in a cell phone restart.

4.2.2. User Applications

User applications are those which are going to be manipulated seeking for a more efficient energy management. User applications can be:

- *CPU applications:* Applications that usually make greater use of the components; display and CPU. Especially when most processes are in foreground. For such applications energy saving will preferably be in display's energy consumption, reducing brightness and the time it takes to shut down. If they are applications running in background it could be convenient to close them.
- *Comunication Applications:* Applications that take greater use of communication activities and display. Display usage will occur when the application is in foreground, either to check or send information that the application needs.In this type of applications power consumption can be reduced in two ways: one is focused on the display, reducing brightness and time of suspension. And the other way is to focus on communication activities. Both approaches are effective, but the results depend on the type of users and the number of times and period of time spent using these applications during their daily routine.

4.3. First Results Analysis

Applications employed by the user during the time the test was performed got reviewed in order to

Table 3: Top 15 consumming Applicactions

Application	Total	Assigned Type	Oled	%Oled	CPU	%CPU	Wi-Fi+3G	%Wi-Fi+3G
WhatsApp	2000.0	Communication	1500	75.00%	92.3	4.62%	411.4	20.57%
Sync. Google	1500.0	Communication	0	0.00%	36.8	2.45%	1400.0	93.33%
Android OS	1000.0	CPU	533.6	53.36%	388.3	38.83%	104.4	10.44%
Google Talk	976.0	CPU	957.2	98.07%	18.8	1.93%		0.00%
Skype	812.1	Communication	154.5	19.02%	14.2	1.75%	643.4	79.23%
Hotmail	569.6	Communication	110.7	19.43%	17.0	2.98%	442.0	77.60%
Dictionary	485.4	CPU	473.8	97.61%	11.6	2.39%	0.0	0.00%
Home Screen	445.1	CPU	423.6	95.17%	21.5	4.83%	0.0	0.00%
Power Tutor	424.8	CPU	365.7	86.09%	119.3	28.08%	0.0	0.00%
Mail Client	379.1	Communication	16.5	4.35%	7.4	1.95%	355.2	93.70%
BeautifullWidgets	374.2	Communication	36.9	9.86%	65.5	17.50%	271.8	72.63%
Top Eleven	310.7	Communication	171.1	55.07%	35.6	11.46%	104.0	33.47%
Maps	279.3	Communication	15.9	5.69%	47.5	17.01%	218.4	78.20%
Google Now	272.7	Communication	0	0.00%	20.6	7.55%	251.4	92.19%

develop a table that classifies applications depending on their component's energy consumption.

It is possible to observe that most applications that our test user employed were "Communication" type. As it was pointed before, test user mostly uses the cell phone as communication tool.

However, it should be noted that the user didn't employ the cell phone as a music player or to play any of the video games previously installed. Had it done so, these new applications would have been classified as CPU and they should been placed at the top of the table.

It is also important to note that depending on how much each application uses the display can show if the application is kept at foreground or restricted to background. For example, WhatsApp application, according to the results, was the most employed by the test user and remained at foreground for the longest time.

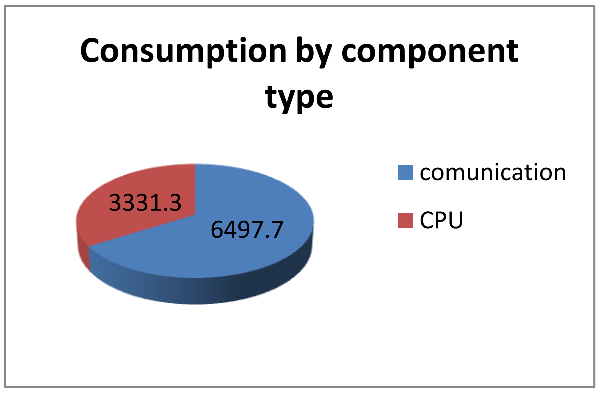


Figure 3: Consumption by component type

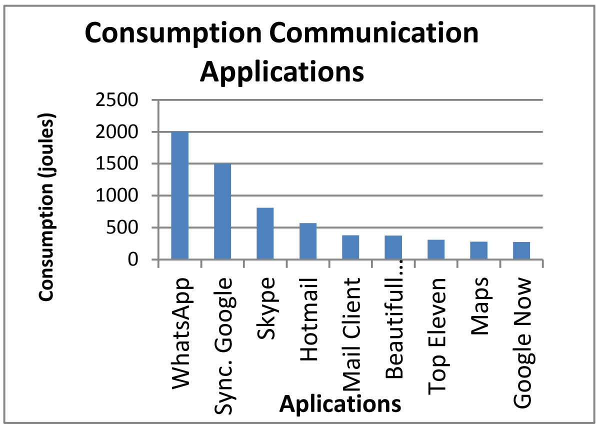


Figure 4: Consumption by Communication Applications

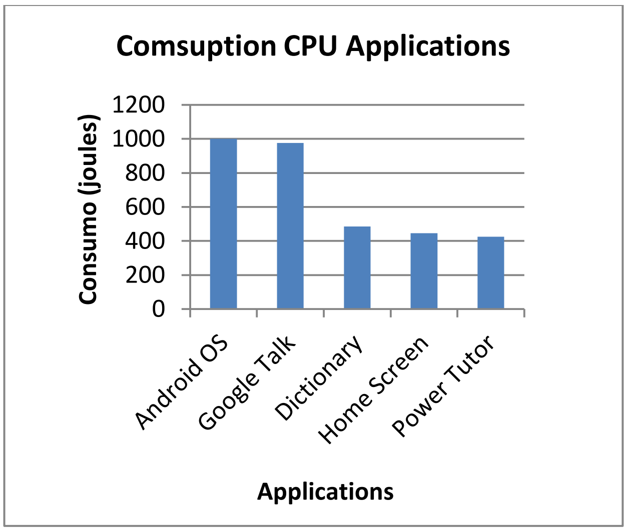


Figure 5: Consumption by CPU Applications

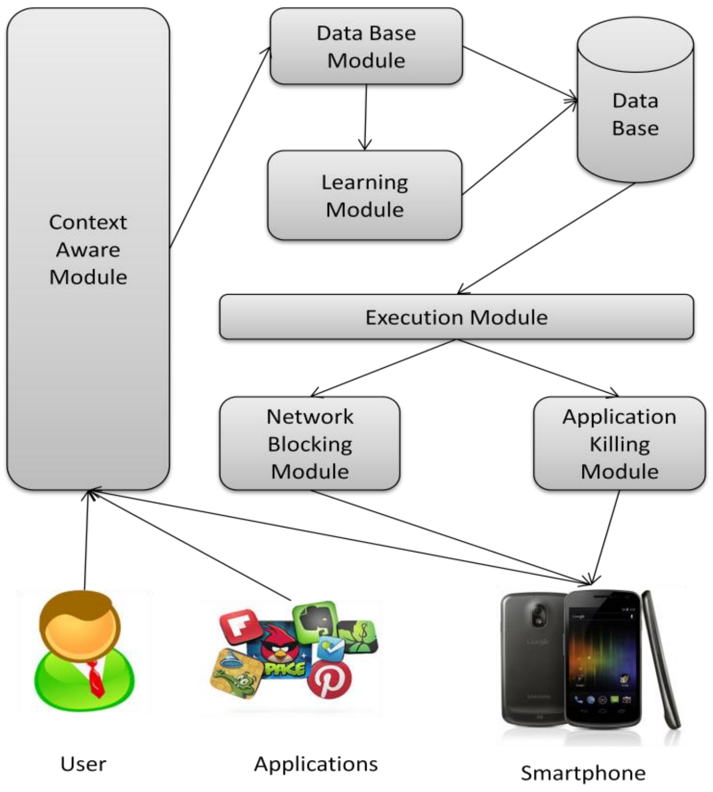


Figure 6: Module's Diagram

4.4. Actions Taken

To reduce energy consumption per application the following actions are proposed:

- **Turn off all unnecessary hardware:** [6] Turn off: GPS, Wi-Fi, Bluetooth, and use Edge connection instead of using 3G or LTE connections.
- **Optimize the power consumption of display:** [7] Turn it off if possible, otherwise use Automatic Brightness.
- **Close CPU consumption applications:**By eliminating background processes a huge energy saving can be achieved.
- **Increase the time interval between reloading information in Communication applications:**Set an interval of time where the communication applications will be frozen until information gets refreshed. In this work we define freeze as limiting all possible internet communication. Also information should be refreshed or updated within a determined interval of time.

4.5. Initial Tests

These actions were implemented statically within a project application.

Subsequently the Samsung Galaxy Nexus smart phone was left maintaining a communication with a medical device to receive medical information 1 time every second.

After 8 hours of this process, comparing results with the initial measurements, without any energy saving implementation, this static solution shows a 10% of energy saved.

5. Complete Solution

The final solution implements a learning module for context awareness purposes. Dynamic profiles are created per application for not compromising user experience. Fig. 6 shows the organization of the proposed modules.

5.1. Context Awareness Module

The proposed solution needs a large information base that will convert in information so it can work properly. The first step for saving energy is to take awareness of the context that surrounds the user and the Smartphone.

The Context Awareness Modules is primarily based on PowerTutor. PowerTutor uses variables to estimate energy consumption, but these variables are only available for PowerTutor's authors oriented Smartphones which are limited by number and most of them are old or discontinued. This problem makes PowerTutor's measurements not as exact as expected. Most of them are just used for referential which doesn't limit our Knowledge data base for one reason: Any kind of smartphone despite of their brand or manufacturer is going to have a similar way to consume energy, for example, on any smartphone the display is going to represent the most consuming component on an smartphone, followed by the Network Interfaces (3G or Wi-Fi).

Application's consumption is estimated based on PowerTutor reports and then classified by each one of the installed applications so then it can be subdivided by component consumption. Finally, all this data is processed and send to the Data Base module for storage.

5.2. Data Base Module

Data Base Module gets all the data from the Context Awareness Module so it can introduce it to the application's data base. Fig. 7 shows the structure of the data base.

After all data is inserted from the Context Awareness module to the data base the Data base Module gets one last call from the Context Awareness Module instructing it to start the Learning Module. The Learning Module process all the data and inserts the results in the data base for storage.

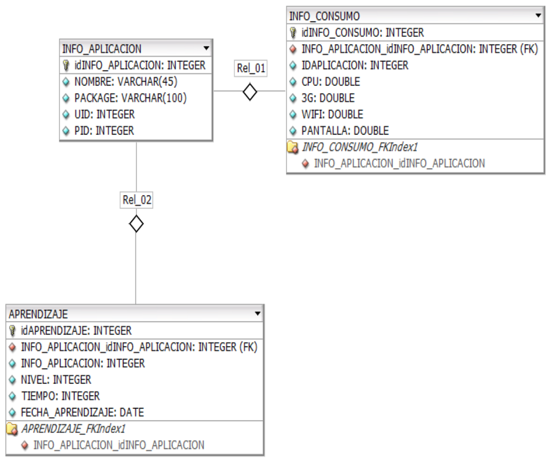


Figure 7: Data Base Diagram

5.2. Learning Module

After getting and storing all the data, there's a call to the Learning Module so it can obtain knowledge from this large amount of data.

The chosen method for getting knowledge using the Learning Module is classifying the applications in consumption divisions. Each application is assigned a level from 1 to 10 and a blocking time as shown in Table 4.

During this blocking time applications will get blocked from using networks interfaces for accessing to the web or applications will be closed.

This method was chosen because it guarantees the less energy consumption for the Execution Method.

Table 4: Blocking Time per Level

Nivel	Tiempo de Bloqueo (s)
1	30
2	60
3	90
4	120
5	150
6	180
7	210
8	240
9	270
10	300

To assign a level to each application, the solution focuses in display's energy consumption. First it sums the total display's consumption then compares it with the display's energy consumption for each application and then assigned a level to each application depending on the display's energy consumption.

Larger energy consumption represents an application that spends more time on foreground so it's considered as a more important application for the user. Then, a level is assigned to each application where 1 represents a more important application for the user and 10 represents a less important application for the user. The lower the level of the application the more time it gets network blocked or closed by the execution module.

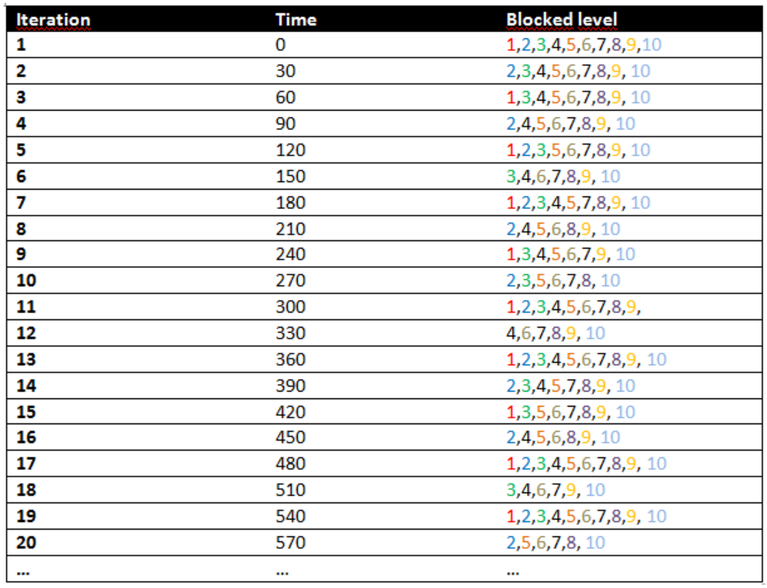
5.3. Execution Module

Execution module it's the most important module for the proposed solution. It can be considered as an orchestra conductor. After obtaining all data and knowledge from the data base the execution module starts to manage both subordinate modules: Network Blocking Module and Application killing module.

These two modules are just simple modules that follow instructions from the Execution Module.

The execution model lifecycle starts with the user's petition and starts iterating every 30 seconds. On each one of the iterations, the execution module decides whether to block the network or close an application as seen in Table 5.

Table 5: Levels blocked per iteration



Iteration	Time	Blocked level
1	0	1,2,3,4,5,6,7,8,9,10
2	30	2,3,4,5,6,7,8,9, 10
3	60	1,3,4,5,6,7,8,9, 10
4	90	2,4,5,6,7,8,9, 10
5	120	1,2,3,5,6,7,8,9, 10
6	150	3,4,6,7,8,9, 10
7	180	1,2,3,4,5,7,8,9, 10
8	210	2,4,5,6,8,9, 10
9	240	1,3,4,5,6,7,9, 10
10	270	2,3,5,6,7,8, 10
11	300	1,2,3,4,5,6,7,8,9,
12	330	4,6,7,8,9, 10
13	360	1,2,3,4,5,6,7,8,9, 10
14	390	2,3,4,5,7,8,9, 10
15	420	1,3,5,6,7,8,9, 10
16	450	2,4,5,6,8,9, 10
17	480	1,2,3,4,5,6,7,8,9, 10
18	510	3,4,6,7,9, 10
19	540	1,2,3,4,5,6,7,8,9, 10
20	570	2,5,6,7,8, 10
...	...	...

When the user decides to stop the execution module, its final action consists on reverting to the default configuration all modifications so the user can continue using his smartphone normally.

Another important task that this module has is to distinguished user applications from Android System Applications.

5.4. Application Killer module

The application Killer module is one of the simple modules that work under the directions of the execution module.

The Application Killer module's main purpose is to close and start applications on each iteration from the execution module.

When an application is installed on a smartphone; it's also assigned with an UID. The UID is a unique identifier that works as parameter for Application Killer module that executes these actions using Linux Commands like *kill*.

5.5. Network blocking module

The second simple module is the Network blocking module for blocking the applications to access to internet through any of the Network interfaces like 3G or Wi-Fi. It's based on IPTables, a linux tool that works as a firewall implemented in all Linux Kernels.

IPTables are a set of rules saved on a file that is reviewed by the system every time one application intents to access the web. That way, IPTable filters any Web Access that doesn't satisfy the IPTable's rules. Any blocked application thinks that there is no internet connection active so they don't consume energy during that time

Network Blocking Modules is based on DroidWall, an android Tool that provides an API for the creation and management of set of rules for filtering Network Packages.

6. Final Results

The final solution was installed in 3 test Android devices. First, some initial tests were taken for comparing purposes before and after the installation of the solution.

Then they were leaved for about one day learning about the user's context.

Then, there was a final energy consumption test which results are shown in Table 6.

The average percentage of energy saved between the 3 devices is 7% over the user's smartphone battery; but the most highlighting is that user's experience on the use of the smartphone is barely noticeable.

7. Conclusions

- It's possible to identify, between the installed applications, which represents the most energy consumption. After identifying the can be evaluated and energy consumption optimized.
- It's possible to store, in an unnoticeable way, user's context. Knowing that the user is one critic and important factor in energy consumption.

Tabla 6: Final Results

	Cellular	Before installing the Solution			After installing the Solution		
		Hours	Joules	Battery %	Hours	Joules	Battery %
1	Samsung Galaxy Nexus	8	4784	25%	8	4401	17%
2	Asus Nexus 7	8	4912	10%	8	4519	6%
3	LG L3 E400	8	2482	29%	8	2283	20%

- It's possible to analyze and understand user's context while he is using this smartphone, which is shown in the correct learning methods.
- Two actions were defined and implemented for saving precious energy consumption on an smartphone.
- Each application can be defined by a dynamic profile depending on its own energy consumption. This profile depends in which ACTION to use and how much time the application is going to be blocked or closed.

8. Future Works

- Optimizing energy consumption during learning time based on:
  - ✓ In learning time the solution increments consumption for registering all data;
  - ✓ In execution time there are no high energy consumption problems;
  - ✓ This high energy consumption problem is because PowerTutor is designed for getting exact data, even though we just need referential data.

- Optimizing CPU applications discrimination, based on:
  - ✓ It was noticed that some applications were closed even though the applications was very important for the user;
  - ✓ For example, a Music player was closed by the solution even though it shouldn't be closed;
  - ✓ Finally the solution can be optimized by filtering CPU applications in a more efficient way.

9. References

[1] Yi-Fan Chung, Chun-Yu Lin, Chung-Ta King "ANEPROF: Energy profiling for Android Java virtual machine and applications" in 2011 IEEE 17th International Conference on Parallel and Distributed systems, p. 372-379.

[2] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P. Dick, Z. Morley Mao, Lei Yang "Accurate online power estimation and automatic battery behavior based power model generation for smartphones" Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis. Pages 105-114

[3] Aaron Carroll, Gernot Heiser "An analysis of power consumption in a smartphone" Proceeding USENIXATC'10 Proceedings of the 2010 USENIX conference on USENIX annual technical conference Pages 21-24

[4] Alex Shye, Into the Wild: Studying Real User Activity Patterns to Guide Power Optimizations for Mobile Architectures . Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture Pages 168-178 ACM New York, NY, USA ©2009

[5] Fangwei Ding, Feng Xia, Wei Zhang, Xuhai Zhao, Chengchuan Ma "Monitoring energy consumption of smartphones" in 2011 IEEE International conferences on internet of things, and cyber, physical and social computing, p. 610-613.

[6] Soumya Kanti Datta, Christian Bonnet, Navid Nikaein "Android Power Management: Current and Future Trends"in the first IEEE workshop on enabling technologies for smartphone and internet of things.

[7] Stephen P. Tarzia, Peter A. Dinda, Robert P. Dick, Gokhan Memik"Display Power Management Policies in Practice"

[8] Josh Feiser, Vijay V. Raghavan, Teuta Cata "A risk-based classification of mobile applications in healthcare" in International journal of healthcare delivery reform initiatives, p. 28-31.