MyBatRecommender: Automated optimization of energy consumption for Android smartphones in software layer

Marcel Popolin de Araújo Cunha
UFSCar - *campus* Sorocaba
Sorocaba, São Paulo 18052-780
Email: mpopolin@gmail.com

Luciana Aparecida Martinez Zaina
UFSCar - *campus* Sorocaba
Sorocaba, São Paulo 18052-780
Email: zaina.luciana@gmail.com

***Abstract*—Nowadays smartphones are composed of a wide range of sensors, components and resources such as GPS (Global Positioning System), Bluetooth and Internet connection through Wi-Fi, 3G, among others. Along with the smartphone's increasing popularity around the world, there is an increasing development and popularity of power-hungry applications: applications that take advantage from these resources and may reduce the energy life time of smartphones to a few operation hours a day. The article goal is to present a mechanism that is able to dynamically manage the smartphone components states, for example turning off unnecessary interfaces, at run time. For this the mechanism collects and analyze data from the smartphone usage along the days in order to predict when the components should be managed. The experimental results show that up to 30% of energy savings is achieved when comparing to the energy dissipation of an smartphone without the proposed mechanism installed.**

***Index Terms*—Energy management, Energy optimization, Mobile applications, Context sensitive applications.**

I. Introduction

Nowadays smartphones have become extremely popular and technologically evolved, being equipped with a wide range of sensors, such as: GPS, light sensors, accelerometer, network interfaces, CPUs (Central Processing Unit) and many others. This scenario has allowed that a wide variety of applications for these smartphones have been developed. From e-mails managers to navigation systems, they are interested in providing a rich user experience, making intense use of the components available on the device and therefore consuming energy. For example, the context-sensitive applications, that are able to adapt their operations without explicit intervention of the users, providing information and services that are relevant for users to perform their tasks using information taken out of the interaction context [1]. For this, they collect user's contextual information, using sensors present on device. An example are the navigation systems, that uses the GPS to locate the user in a determinate map.

Basically, the amount of energy available on device is the result between the difference of the energy provided by the battery and the energy consumed by the sensors and components. It is considered that the energy consumption by the sensors and components is increasing and that the technological evolution of the smartphone's battery has not followed the evolution of the other components of the device, resulting in a frequent scenario where the smartphone is out battery before the end of the day, as stated by [2]. Although there is a crescent effort on studies that aim to evolute the battery on hardware level, new ways to manage and optimize the energy consumption are necessary. Due to this lack, many researchers are concentrating their efforts to provide solutions on software layer.

The researches on this area exist before the popularization of smartphones, with some studies dated of 2004, when a similar problem occurred on PDAs (Personal Digital Assistant) [3]. However, the research on this area has increased with the smartphone's and its applications popularization, in 2011, mainly to the Android based smartphone popularization, with around 380 thousands applications and 10 billion downloads [4]. Basically, the Android framework is organized in the architectural layer represented in the Figure 1. The first layer is the layer based on the kernel Linux, which handles the access to the hardware level components via drivers. Above this layer there are other two layers of libraries: one developed in C and C++ programming languages and the other developed in the Java programming language. This last one is the responsible for providing the interface with the last layer, that is the layer of Android applications, exposing the hardware controls (e.g., managing sensor states) through APIs (Application Programming Interface) and *wakelocks*[1] to developers [5], [6]. This approach could lead to the development of energy inefficient applications if the hardware components are not used correctly, leading to a unnecessary energy waste [7].

Based on demand of reducing the energy consumption, this paper has the main goal of presenting a solution in the software layer to save energy on smartphones. This is done by presenting a mechanism, MyBatRecommender, that manages dynamically the state of the components of the smartphone. The proposed mechanism efficiency is also measured using experimental tests.

The remainder of this paper is organized as follow: the

[1] developer.android.com/reference/android/os/PowerManager.WakeLock.html

DOI reference number: 10.18293/SEKE2016-131

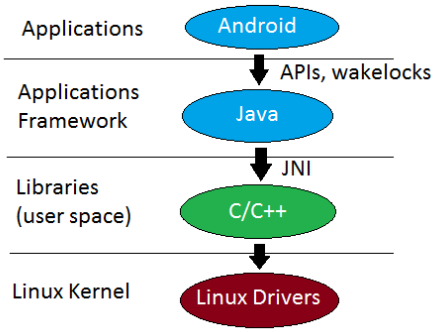

Fig. 1. Android architectural overview

Section II presents the main related works of the present study, the Section III presents the MyBatRecommender mechanism characteristics and implementation for the Android operational system. The Section IV presents the conduction and results of MyBatRecommender experimentation. Section V summarizes the results of the present study, and concludes indicating possible new directions for further works.

II. Backgrounds and Related Works

The researches with different approaches have increased in this area and could also be observed also in a qualitative way. Ones address the subject from analyzing the integration of smartphones with Cloud Computing as discussed by [8] and [9] that are focused in the Cloud offloading method, that tries to offload the execution of part of the application to the Cloud, to sensor's optimizations as discussed by [10], that tries to improve the energy efficiency of location components and others that identify how the energy is consumed on the device by its sensors and applications, as discussed by [11]. This last category of studies helps to identify new ways which could be explored to save energy and is the base for another category of studies that proposes solutions for the management and optimization of energy consumption. They are complete, complex and include most of sensors and components available on a device, as the work presented by [12] that shows how the components 3G, Wi-Fi and GPS consume energy on different states such as idle, concluding that, even in the idle state, when the components are turned on but are not actively executing their tasks, they have an energy consumption of around 25% of the overall system. Based on these measurements the authors propose a technique that try to avoid this unnecessary energy consumption.

Furthermore, recently, the company Google has shown its interest in this research area presenting in its Android new version, named Android Marshmallow, some features related to energy savings [13].

III. MyBatRecommender

The energy consumption in smartphones varies according to the user usage profile, that dictates how the components are used along the days. These components are turned on to provide the resources needed by the applications, but may still turned on when they are not necessary any more, what consequently increases the energy consumption. Considering this, one possible approach to reduce the energy consumption is to avoid that the components stay on unnecessary states, turning them off when they are not being used. Based on this, the current work proposes a mechanism, named MyBatRecommender, which aims managing and optimizing the energy usage in smartphones to reduce it. This mechanism acts managing the states of the smartphone's components, using for this the user usage profile, that is created by the analysis of the smartphone usage data, collected along the days. The "recommender" part of the mechanism name exists considering that the MyBatRecommender, as will be further detailed, adopts its operation depending on the user, managing different sensor and components according to the user usage profile.

Figure 2 shows an overview of MyBatRecommender, that is composed by four main elements, MyBatLogger (1), MyBatServer (2), MyBatProfile (3) and MyBatSaver (4), which will be detailed on the following sections. Also, the mechanism works in a continuous way, once that each element is always executing its role to keep improving the mechanism efficiency. The mechanism is also automatic in the way that once it is installed on the user's smartphone it will automatically run and execute its role without the need of user intervention, running always in background, not interfering in the user experience with the smartphone.

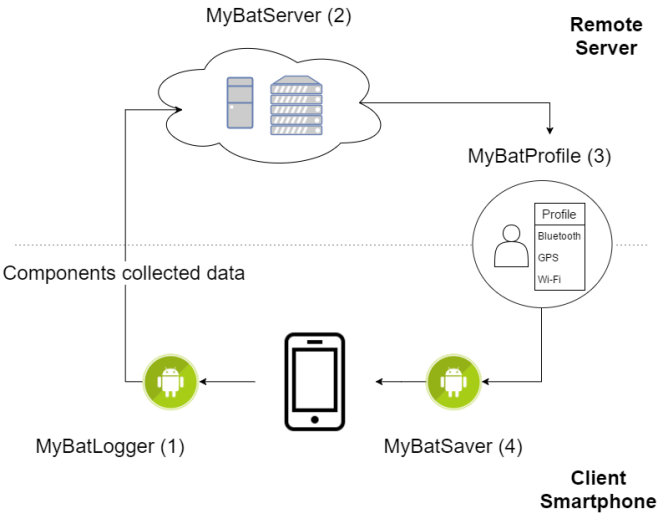

Fig. 2. MyBatRecommender overview

A. MyBatLogger

To understand how the components behave on users smartphones, data regarding the utilization of components should be gathered to be further analyzed in order to create the user usage profile. The element MyBatLogger is responsible for collecting data regarding the components *Bluetooth*, Display, Wi-Fi, Battery, Mobile Network, GPS, and sending to a remote server.

Based on this, the element MyBatLogger was implemented as an application for Android smartphones. This application is composed by a main Service[2] that is continuously running in background since the smartphone boot. This Service is the responsible for collecting the data of all above mentioned components every minute or of a specific component when there is some event regarding that component, for example, when Wi-Fi is turned on or off. This operation is denominated "generate *log*" where the *log* is the bundle of information representing the data collected of a component. Each component has specific data that is collected besides the *id*, unique for each smartphone, the *timestamp*, representation of when the *log* was generated and *type*, indicating the component that the *log* represents. The specific data and its collecting method depends on each component, but, in general, they represent the state of that component (on or off) and the connection state - if the component was connected, connecting, sending or receiving data, among other possible states.

B. MyBatServer

The MyBatServer element is the remote server side of the mechanism. It is the responsible for receiving requests from the client side, represented by MyBatLogger and MyBatSaver elements, and processing them according to three main actions. The first action is "Insert *logs*", responsible for receiving the collected *logs* from the MyBatLogger and inserting them on the corresponding database. The second action is "Create MyBatProfile", which analyses all the data collected and stored in order to generate the MyBatProfile, that represents the user usage profile. The MyBatProfile creation process will be explained on Section III-C. The last action, "Request MyBatProfile", is the responsible for returning the MyBatProfile created to the requesting part.

As the number of collected *logs* is huge, the MyBatServer element has fundamental role in storing and processing this data, once that if this task was done by the smartphone a non negligible amount of memory and energy would be consumed, decreasing the efficiency of the mechanism.

C. MyBatProfile

The element MyBatProfile represents the user usage profile, indicating how the smartphone's components are used along the days by the user. It is created by the MyBatServer, when the action "Create MyBatProfile" is invoked, and it is also the responsible for showing the MyBatSaver how it should behave, displaying the components states over the periods and days.

The MyBatRecommender's idea is to delegate the creation methods of MyBatProfile to the implementation itself, meaning that different methods can be used in order to analyze the data and generate MyBatProfile. Based on this, the current implementation is based on the existence of a user's daily routine. For example, for a certain university student, this routine could be the student going from his house to the university, watching the lessons and at the end of the lessons, going back to his house. Then, through this daily routine it is possible to obtain a smartphone usage daily routine, because the smartphone usage depends on certain environment variables such as availability of Wi-Fi networks.

Based on this, the present paper proposes the MyBatProfile generation so it will determine the configuration of the smartphone components for each period in a day and for every day in a week. To determine how many periods a day is composed by, a questionnaire was applied to six different persons, asking about how many different environments they visit along the day, once that, as discussed above, the environments may change directly the way the smartphone is used. The result was an average of 5.4 periods. As this number must be an integer, it was considered 6 as the number of periods. Each period has a start and end time that, considering a twenty four hours day, is four hours long.

To create the MyBatProfile, five main developed algorithms are executed, resulting in the profile element created and inserted in the MyBatServer local database. They are:

- ***separateByDays()***: this is the first step executed in order to create the MyBatProfile and is the responsible for grouping the collected *logs* according to the week day they were collected, using for this the attribute *timestamp*.
- ***separateByPeriods()***: once the *logs* are grouped in days, the second step runs and, for each day, groups the *logs* in the six defined periods.
- ***createUserProfile()***: having the *logs* grouped in days and periods, they are ready to be analyzed to create the MyBatProfile. So this step runs and, for each period, of each day, executes the last method, *analyzeType()*, which will be the responsible for determining the state of that component for that specifc period.
- ***analyzeType()***: this method analyzes the collected *logs* of each period to determine the state of the components for that period. The *logs* of each component are analyzed individually, using for this the attribute *type*. For this, the analysis considers, for each *log*, if it was **used or not**, condition that varies for each component, explained on Table I. Then, if the number of the *logs* considered **used** is greater than the number of *logs* considered **not used**, the state of that component, for that period, is set to be **on**, otherwise it set to be **off**.
- ***insertProfile()***: finally, with the MyBatProfile created, the last step is the responsible for inserting it on the MyBatServer database so it can be fetched later.

As mentioned in the beginning of this section, MyBatRecommender is continuously executing, meaning that it is always collecting new data from the user smartphone to be analyzed and create or update the MyBatProfile. This approach tries to improve the energy efficiency seen that it keeps track of the user usage, so if user change his operating way the MyBatProfile will be updated to reflect this new operating way.

[2]developer.android.com/guide/components/services.html

[3]developer.android.com/reference/android/os/BatteryManager.html

TABLE I
CONDITIONS TO CONSIDER COMPONENTS USED OR NOT

Component	Condition
Battery	The battery state must be different than BATTERY_STATUS_CHARGING defined by BatteryManager[3]
Bluetooth	The *Bluetooth* state must be equals to STATE_ON and the *Bluetooth* connection state must be equal to STATE_CONNECTED or STATE_CONNECTING, defined by BluetoothAdapter[4]
GPS	The GPS state must be equals to GPS_EVENT_STARTED or GPS_EVENT_SATELLITE_STATUS defined by GpsStatus[5]
Display	The *Display* state must be ON, represented by Intent.ACTION_SCREEN_ON, defined by Intent[6]
Mobile networks	The Mobile networks state must be equals to DATA_CONNECTING or DATA_CONNECTED and the connection state must be one of DATA_ACTIVITY_IN, DATA_ACTIVITY_OUT or DATA_ACTIVITY_INOUT, defined by PhoneStateListener[7]
Wi-Fi	The Wi-Fi state must be WIFI_STATE_ENABLED, defined by WifiManager[8] and the connection state must be CONNECTED or CONNECTING, defined by NetworkInfo.DetailedState[9]

D. MyBatSaver

The MyBatSaver is the element responsible to apply the configuration defined by MyBatProfile. For this MyBatSaver interacts with MyBatServer in order to retrieve the last MyBatProfile created, using the smartphone unique *id* as shown in Section III-A. Once in possession of the profile, the element schedules itself to run at the start of each period, using for this the Android AlarmManager[10]. Then, when executed, the element retrieves the configuration linked to that period and acts changing the state of the components accordingly, turning the components on or off. After this, MyBatSaver waits for the next period to start, so it can repeat the steps and change the smartphone components states again, reflecting the new period configuration.

This operating way creates a dependence on the number of periods, once that the MyBatSaver executes only at the start of each period. Thus it is important to base the number of periods chosen on the user behavior, as explained in the beginning of this section. Also, other ways could be explored to change this dependency, as stated in section V.

[4]developer.android.com/reference/android/bluetooth/BluetoothAdapter.html
[5]developer.android.com/reference/android/location/GpsStatus.html
[6]developer.android.com/reference/android/content/Intent.html
[7]developer.android.com/reference/android/telephony/PhoneStateListener.html
[8]developer.android.com/reference/android/net/wifi/WifiManager.html
[9]developer.android.com/reference/android/net/NetworkInfo.DetailedState.html
[10]developer.android.com/reference/android/app/AlarmManager.html

IV. EXPERIMENTATION

In this work, the experimentation phase was carried out by running two main experiments. The first one was the measurement of the mechanism overhead and the second was the measurement of the mechanism efficiency. To run both experiments, two smartphones Sony Xperia Z2[11], with the same configuration, were used and they will be referenced as DUT 1 and DUT 2 in the remainder of this section, where DUT are the initials of Device Under Test, indicating that these smartphones were used to run the necessary tests.

A. MyBatRecommender overhead

The proposed mechanism, when implemented to the Android operational system, requires the usage of some smartphone's components and resources, what, consequently, incurs in energy consumption. This energy consumption, necessary to make the mechanism run, is considered the system overhead, for the current work.

To measure the MyBatRecommender overhead, the present work used the Keithley Source Measure Unit/Charge Simulator[12], an external equipment able to measure the electric current in device and, with an auxiliary software, save this data on computer for posterior analysis.

To run this experiment, the DUTs were configured in the following way: the DUT 1 had no mechanism installed and the DUT 2 had the MyBatRecommender installed. Also, both DUTs ran the experiment with the screen turned off, the airplane mode turned on and only with the component Wi-Fi turned on and connected to a network. This was necessary to avoid any energy consuming element besides the MyBatRecommender execution, that needs the Wi-Fi connection to send the collected data to the remote server. Considering this, both DUTs were connected to the Keithley Source Measure Unit during one hour to collect the data regarding the electric current. As Keithley Source Measure Unit has a sample rate of two second, the period of one hour resulted in a group of one thousand and eight hundred samples collected for each DUT, so only a sample is shown on Table II to demonstrate the format of the data.

TABLE II
KEITHLEY SOURCE MEASURE UNIT COLLECTED DATA

Time (ms)	Time	Offset (ms)	Current (mA)
1452099728914	15:02:08	531967	7
1452099730789	15:02:10	533842	7
1452099732664	15:02:12	535717	7
1452099734540	15:02:14	537593	7
1452099736415	15:02:16	539468	7

To analyze the data and compare the energy consumption of both DUTs, initially, the electric current average (AVG) was calculated for each DUT:

$$AVG = \frac{\sum(electric\ current)}{number\ of\ samples} \quad (1)$$

[11]http://www.sonymobile.com/global-en/products/phones/xperia-z2/
[12]http://www.tek.com/sites/tek.com/files/media/media/resources/2308.pdf

After, considering the DUTs battery capacity, 3200 mAh, it was calculated the representativity (REP), which indicates the DUT energy consumption per hour, in percentage:

$$REP = \frac{AVG \ * \ 100}{3200} \quad (2)$$

Finally, the difference of REP for both DUTs represents the overhead of the mechanism. The Table III show these values for each DUT.

TABLE III
OVERHEAD DATA ANALYSIS RESULTS

DUT	Start time	End time	MED (mA)	REP (%)
1	13:45:15	14:45:15	9.212389381	0.2878
2	15:02:08	16:02:09	10.12324493	0.3163
			Difference	0.0285

By the analysis of the data presented by the Table III, there is a difference of energy consumption between the DUTs of 0.0285% of energy per hour. Considering that the configuration difference of these DUTs is the presence of the MyBatRecommender on DUT 2, the conclusion is that the proposed mechanism has an overhead of 0.0285% of energy per hour. This value will be used on the posterior analysis to remove the mechanism overhead from the calculations when needed.

B. MyBatRecommender efficiency

The mechanism efficiency validation was driven by a proposed scenario, responsible for determining which component would be in which state for all periods in a day and all days in a week.

To run this experiment, the DUTs were configured in the following way: the DUT 1 had the MyBatRecommender installed and the DUT 2 had only the MyBatLogger element installed. This was necessary because the element MyBatLogger, present on both DUTs, was the responsible for collecting the information regarding the battery level and uploading it to the remote server, so it could be analyzed posteriorly. Considering this, both DUTs ran the experiment over two weeks, following the instructions presented by the proposed scenario. For DUT 1, both weeks were used to collect information about the battery level, but the first week was also the responsible for collecting the data necessary to create the MyBatProfile, and the second week was also the responsible for applying the energy saving actions, defined by the generated MyBatProfile. For DUT 2, as it had only the element MyBatLogger installed, both weeks were responsible only for collecting information about the battery level.

The data analysis was driven by a research question an three main hypothesis, where one is accepted rejecting the other two. The research question is: *Is there difference of energy consumption between both DUTs?* and the hypothesis are: the null hypothesis (H_0), where there is no significant ($\geq 5\%$) difference of energy consumption between the DUTs, the first alternative hypothesis (H_1), where the DUT 1, with the MyBatRecommender installed, has a smaller significant difference of energy consumption than the DUT 2 and the second alternative hypothesis (H_2), where the DUT 2, without the mechanism installed, has a smaller significant difference of energy consumption than the DUT 1.

Based on this, to analyze and compare the energy consumption of both DUTs, the consumption rate (μ_T) was calculated for each DUT. The consumption rate is the percentage of energy consumed in a period (ϕ) divided by the time, in hours, of this period (τ). For this only the discharging periods of the experiment were considered. Discharging periods are periods were the energy is being consumed. The Tables IV and V show these values for the first and second weeks of the experiment, respectively.

TABLE IV
DUT 1 AND DUT 2 FIRST WEEK DATA

		τ	ϕ	μ_T
DUT 1	**Period 1**	66	94	1.4242
	Period 2	27	26	0.9629
	Period 3	54	45	0.8333
			Average	1.0734
DUT 2	**Period 1**	66	83	1.2575
	Period 2	27	29	1.0740
	Period 3	56	33	0.5892
			Average	0.9735

TABLE V
DUT 1 AND DUT 2 SECOND WEEK DATA

		τ	ϕ	μ_T
DUT 1	**Period 1**	60	35	0.5833
			Average	0.5833
DUT 2	**Period 1**	61	55	0.9016
			Average	0.9016

By the Table IV analysis, it is noticeable that the average of consumption rate for the first week is very similar for both DUTs, what is expected, once that in this week the MyBatRecommender was only collecting data in both devices. This is not the same result shown by the Table V where the DUT 1, with the MyBatRecommender installed, has a considerable (0.3183%) smaller average of consumption rate than the DUT 2. Discounting the overhead of the mechanism for the DUT 2, that used it only for collecting data regarding the battery level, the difference results in 0.2898%.

It is needed to validate, with some level of significance, if it is possible to reject the null hypothesis over the acceptance of one of the alternative hypothesis, considering the samples collected of both DUTs for the second week. For this, initially, the normal distribution was checked to define the statistical method to be applied and compare both samples. Using the Ryan-Joiner[13] method through the auxiliary tool Minitab[14], it was obtained that both samples have the p-value < 0.01, meaning that they do not have the normal distribution, as the Figures 3 and 4 show.

[13]http://www.statsref.com/HTML/index.html?ryan-joiner.html
[14]http://www.minitab.com/

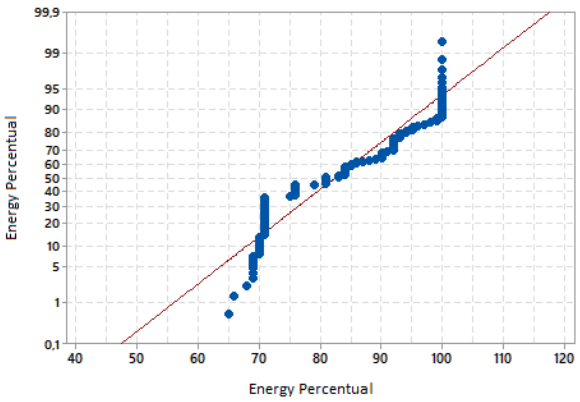

Fig. 3. Normally test for samples from the second week of DUT 1

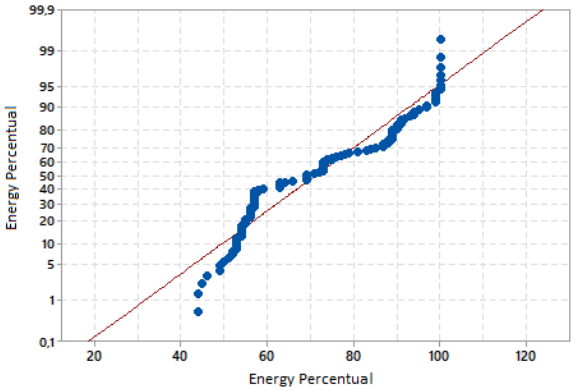

Fig. 4. Normally test for samples from the second week of DUT 2

Based on the normality test result, the Mann-Whitney U-Test[15] was chosen. Considering a level of significance α=0.05, it was obtained that the calculated p-value=0.0 is smaller than 0.05, rejecting the null hypothesis and confirming with a confidence level of 95% that the difference of energy consumption has statistical significance and it was probably caused by the presence of the mechanism MyBatRecommender.

Finally, calculating the relative difference of energy consumption between the DUTs, of 32%, the alternative hypothesis H1 is accepted, rejecting the other hypothesis. This result demonstrates that the proposed mechanism, MyBatRecommender, has an efficiency of 32% of energy savings when applied to the proposed scenario and compared to a smartphone without any mechanism installed.

V. Conclusion

The contributions of this work are: (i) the proposal of the MyBatRecommender, a mechanism for managing and optimizing the energy consumption in smartphones; (ii) its implementation for the Android operational system; (iii) its validation through experimentation, demonstrating the energy savings achieved when the mechanism is used in a controlled scenario. As future works this study can be extended to implement the proposed mechanism for other smartphone operational systems, such as iOS[16]. It can also be improved including new components to be analyzed such as NFC (Near Field Communication) and using other algorithms to generate the MyBatProfile than the proposed by the current work, as including a learning module that defines the number and duration of the periods dynamically instead of considering them static. Finally new experiments could be elaborated and applied, such as analyzing the energy saving by components, in order to validate the outcomes.

[15]https://statistics.laerd.com/minitab-tutorials/mann-whitney-u-test-using-minitab.php

[16]http://www.apple.com/ios/

References

[1] Anind K. D. and Gregory D. A., Towards a Better Understanding of Context and Context-Awareness, 1999.
[2] Robinson, Stuart, Cellphone energy gap: Desperately seeking solutions, 2009.
[3] Krintz, C. and Wen, Y. and Wolski, R.,Application-level prediction of battery dissipation, 2004.
[4] Google Play Wiki Page, http://en.wikipedia.org/wiki/Google_Play, 2013.
[5] Corral, L. and Georgiev, A.B. and Sillitti, A. and Succi, G., A method for characterizing energy consumption in Android smartphones, 2013.
[6] Pathak, Abhinav and Jindal, Abhilash and Hu, Y. Charlie and Midkiff, Samuel P., What is Keeping My Phone Awake?: Characterizing and Detecting No-sleep Energy Bugs in Smartphone Apps, 2012.
[7] A. Pathak, A. Jindal, Y. C. Hu, and S. P. Midkiff, What is keeping my phone awake? Characterizing and detecting no-sleep energy bugs in smartphone apps, 2012.
[8] Marin, Radu-Corneliu and Dobre, Ciprian, Reaching for the Clouds: Contextually Enhancing Smartphones for Energy Efficiency, 2013.
[9] Fekete, K. and Csorba, K. and Forstner, B. and Vajk, T. and Feher, M. and Albert, I., Analyzing computation offloading energy-efficiency measurements, 2013.
[10] Oshin, T.O. and Poslad, S. and Ma, A., Improving the Energy-Efficiency of GPS Based Location Sensing Smartphone Applications, 2012.
[11] Vallina-Rodriguez, N. and Crowcroft, J., Energy Management Techniques in Modern Mobile Handsets, 2013.
[12] Donohoo, B.K. and Ohlsen, C. and Pasricha, S. and Yi Xiang and Anderson, C., Context-Aware Energy Enhancements for Smart Mobile Devices, 2014.
[13] Google IO, https://www.android.com/intl/en/versions/marshmallow-6-0/, 2015.