Mobile Phones and Energy Consumption

Cristea Vlad Vasile, Colin Pattinson and Ah-Lian Kor

Abstract This paper is an extension of (Cristea et al. in International SEEDS conference, 2015) [1] and abstracted from (Cristea in Energy consumption of mobile phones, 2015) [2]. This research contributes to the potential of greening software application as discussed in (Kharchenko et al. in Green IT engineering: concepts, models, complex systems architectures, vol 74, 2017a) [3] and (Kharchenko et al. in Green IT engineering: components, networks and systems implementation, vol 105, 2017b) [4]. Additionally, green design principles abstracted from this research will be relevant for designing the systems in (Kondratenko et al. in Green IT Engineering: Components, Networks and Systems Implementation, 2017) [5] and (Kuchuk et al. in Green IT Engineering: Components, Networks and Systems Implementation, 2017) [6]. Battery consumption in mobile applications development is a very important aspect and has to be considered by all the developers in their applications. This study will present an analysis of different relevant concepts and parameters that may have impact on energy consumption of Windows Phone applications. This operating system was chosen because there is limited research even though there are related studies for Android and iOS operating systems. Furthermore, another reason is the increasing number of Windows Phone users. The objective of this research is to categorize the energy consumption parameters (e.g. use of one thread or several thread for the same output). The result for each group of experiments will be analyzed and a rule will be derived. The set of derived rules will serve as a guide for developers who intend to develop energy-efficient Windows Phone applications. For each experiment, one application is created for each concept and the results are presented in two ways: a table and a chart. The table presents the duration of the experiment, the battery consumed in the experiment, the expected

C. V. Vasile
Garmin, Cluj-Napoca, Romania

C. Pattinson · A.-L. Kor (✉)
School of Computing, Creative Technologies, and Engineering, Leeds Beckett University, Leeds, UK
e-mail: A.Kor@leedsbeckett.ac.uk

C. Pattinson
e-mail: C.Pattinson@leedsbeckett.ac.uk

© Springer Nature Switzerland AG 2019
V. Kharchenko et al. (eds.), *Green IT Engineering: Social, Business and Industrial Applications*, Studies in Systems, Decision and Control 171,
https://doi.org/10.1007/978-3-030-00253-4_11

battery lifetime and the energy consumption, while the charts display the energy distribution based on the main threads: UI thread, application thread and network thread.

Keywords Energy efficiency · Mobile phone · Smartphone energy consumption Rules

1 Introduction

In recent years, the smartphones market has a significant boost. According to eMarketer, the number of smartphone users has grown from 1.13 billion in 2012 to 2.03 billion in 2015 [7]. This ascending trend yields a prediction of around 2.5 billion smartphone users by 2017. This means that around 30% from the world's population will own such a device. There are two dominant operating systems that run on these smartphones: iOS and Android. According to the same source, in the last quarter of 2014 the percentage of smartphones which support Android is 76.6%, while the smartphones which support iOS is represented by only 19.7%. The rest of 3.7% is split between Windows Phone operating system with 2.8%, BlackBerry operating system with 0.4% and others operating systems.

Although the difference between the first two operating systems and the rest is large, in the future these statistics will change. Statistica portal predicts that operating system market in 2017 will look like this: the Android market will decrease to a value around 68.3%, the iOS market will decrease to a value around 17.9% and the Windows Phone market will increase up to 10.2%. These data suggest the fact that Windows Phone operating system is in continual development and in the future it could be a competitor for Android and iOS operating systems.

According to Statistica portal, in October 2014 [8] there is a number of 1.3 million applications in App Store, 1.3 million applications in Google Play and only around 300,000 applications in Windows Store. TheNextWeb.com presents an article [9] in which a spokesperson from Microsoft confirms that the number of application from Windows Store reached 300,000 in June 2014 and the fact that "in the past year alone the Windows and Windows Phone app catalog has grown 94%, while the number of active developers has grown by 50%.". According to the newest statistics from Microsoft [10], in March 2015, there is a number of 585,000 applications in Windows Store. It is noted that the increasing rate of application development is very high, thus promoting Windows Store to become a competitor for App Store and Google Play. This is the reason for conducting experiments for Windows Phone in this study and to conduct detailed analysis of the concepts and controls used by Windows Phone developers.

According to Smart2020 report [11] information technology and communication (ICT) consumes around 2% of the world's energy, however, the ICT sector's emissions 'footprint' is expected to decrease to 1.97% of the global emissions by 2030 [12]. This number can be compared to the total energy consumed by airline industry. In 2020 the mobile phones will represent 1% from the ICT carbon footprint and the mobile network will represent 13%. It is very difficult to calculate very precisely the energy consumed by a smartphone, because this is not only an object used for communication. When a user charges his phone every day or maybe two times per day the total amount of energy consumed by a smartphone will become considerable. Another important factor that should be considered when the energy consumption is calculated, is the whole internet infrastructure. Nowadays the data generated by smartphones and transferred across the internet is significant and it grows continually, because the number of users that access the internet through a smartphone is in an upward trend.

The aim of this study is to compare concepts and controls that are used for developing Windows Phone applications, and to establish a set of rules that can be used by any developer who intends to build an energy efficiency application. There will be a predefined number of rules that will be tested and which will encompass the following components: UI, processing, and network.

This study makes the following contributions: it investigates the energy consumption of Nokia smartphones running on Windows Phone 8.1 operating system; it investigates the energy consumption of specific Windows Phone controls; it investigates the energy consumption of specific programming concepts; it provides a set of rules, which will optimize a mobile application from energy point of view.

2 Related Work

Smartphones' energy efficiency research is gathering momentum and it is growing in parallel with the development of the smartphones. Nowadays there are many components like processor or screen that can be optimized, but the battery is not one of them yet. This is why it is very important to have control over the battery and to know exactly which part of the application consumes more energy and why.

Related studies to this paper address the following issues: tools that measure energy consumption [13–15] and provide breakdown of energy consumption the mobile device's main hardware components [16]; overall consumption [17–20]; cloud services [21]; and network measurement [21–23] and power consumed whilst using LTE as well as WiFi network connections [24]. An analysis of the characteristics of the power consumption for context-aware mobile applications has been conducted to form the basis for energy-efficient context-aware services in mobile environments [25] while [26] develop a battery behavior model to explore the effects (i.e. energy consumption and batter drain) of different usage patterns in sensory operations for context-aware smartphones. An investigation on the energy consumption of wireless communication interfaces (e.g. Bluetooth, WiFi, and 3G) during various scenarios (e.g. standby, scanning, and transferring) [27]. Moreover, there are some studies that attempt to improve the battery life. One of these studies is investigated by Parkkila and Porras [28]. The mobile phones field is not the only one where researchers are trying to find some "green" optimizations. Networking is another area of research where a lot of optimizations are made. An example in this category the research by Drouant et al. [29] and Pattinson and Robinson [30].

Notably, most of the studies focus on the hardware components (e.g. optimization of power consumption in multi-core smart phones through power-aware scheduling algorithms, [31]) or on the network (e.g. power consumption analysis of transmission over 3G networks, [32]). Elliott et al. [33] has conducted an investigation of battery and energy consumption of media-related applications in Android smartphones. To date, there is limited detailed analysis of the energy consumption of software components. All of the studies are platform independent, so they can be made for Android, iOS or Windows Phone. For example, one study presents the energy consumption of a display in general but not the factors that influence this consumption. The research in this paper address this identified gap. It attempts to go a step deeper and to analyze different factors that can influence the energy consumption of a mobile application. From [17] work, it is known the fact that the display component is one of the components that consumes the most energy in an application. What is not known is the underlying cause of this phenomenon and how to improve the energy consumption. The purpose of this paper is to identify a part of the element that consumes most of the energy.

3 Methodology

As it was already mentioned in the Introduction Section, the purpose of this research is to provide a set of rules that can be used by developers in order to obtain mobile applications that are more energy-efficient. Nowadays, there are a lot of operating systems for smartphones, such as: Android, iOS, Windows Phone or Jolla. Each of these operating systems has many particularities, so it is very difficult to obtain a set of rules that can be applied to all operating systems. This study will focus only on one specific operating system, Windows Phone 8.1, a product of Microsoft Company released in April 2014 [34].

3.1 Tools

Applications for Windows Phone 8.1 can be developed using Microsoft Visual Studio 2013 [35]. This software is an IDE (integrated development environment) from Microsoft. It can be used for developing desktop applications, websites, web services, Windows applications and mobile applications. Microsoft Visual Studio 2013 includes the following programming languages: C, C++, VB.NET (Visual Basic), C# and F#. Besides Visual Studio, another tool is required in the development process: Windows Phone 8.1 SDK. This tool installs everything that is necessary for developing and testing Windows Phone applications. For the User Interface part, each application can be opened in Microsoft Blend, which is a specialized tool in UI design. Figure 1 presents a basic Windows Phone application in Visual Studio 2012.

The main components [36] that can be found in Visual Studio for Windows Phone are:

- ***Toolbox***—contains a list with all the controls that can be found in the basic installation. Extra components can be added to the project if they are referenced from the solution and from the current page;
- ***Design View***—shows the design of the application. The controls from Toolbox can be dragged directly to the design view and the XAML code will be automatically updated;
- ***XAML View***—shows the code that is generated for the interface. After each modification, the Design view part will be refreshed;
- ***Properties Windows***—offers the possibility to see and to modify the properties of different controls or files;
- ***Solution Explored***—shows all the projects and files that are included in the current solution, in a hierarchical structure;
- ***Target Device***—offers the possibility to choose the device on which the application will run. This device can be a virtual emulator or a real device. The

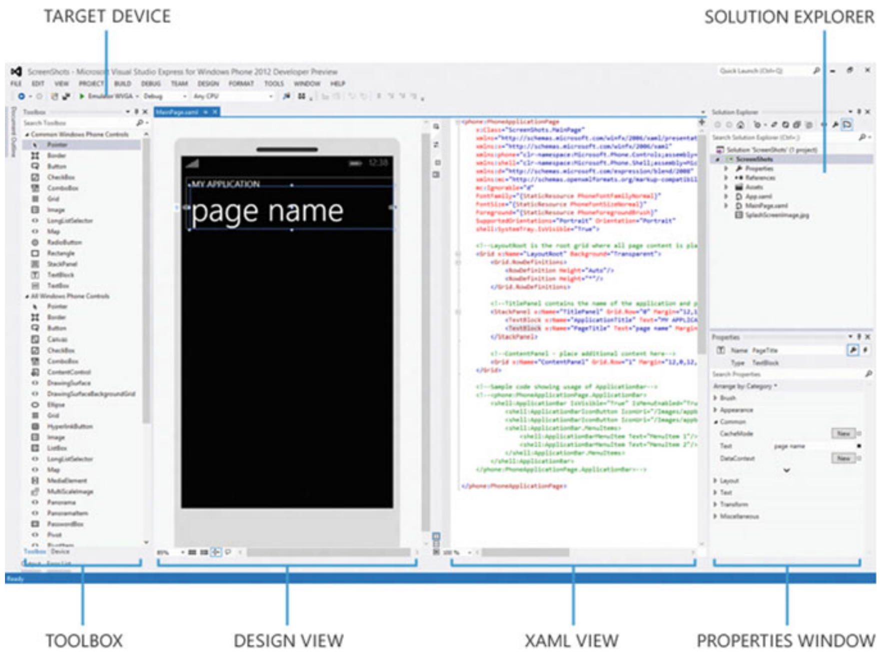


Fig. 1 Visual Studio 2012 for Windows Phone [34]

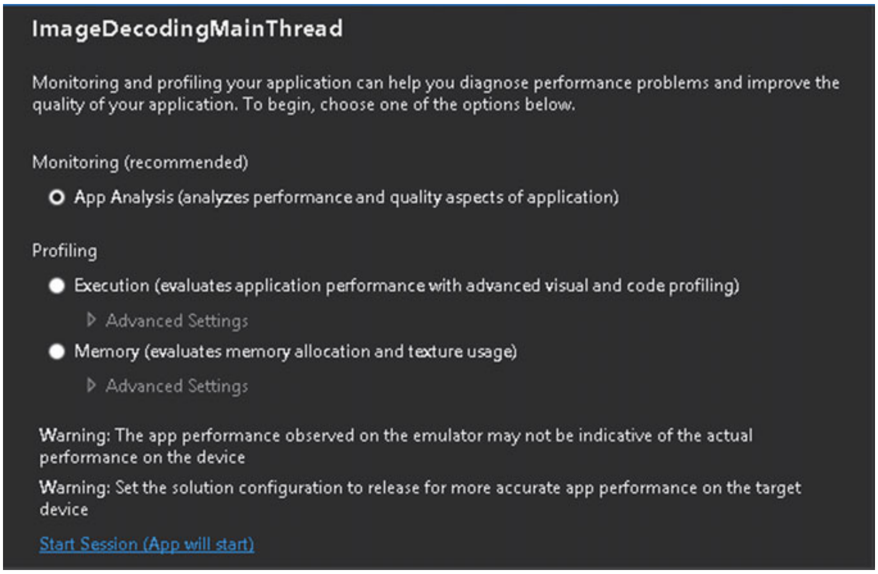
virtual emulator is a desktop application that offers the possibility to simulate a real environment for an application. The emulator is configurable and can simulate any real device, in terms of hardware and software components.

3.2 *Windows Phone Application Analysis Tool*

Another tool that is really useful is the Windows Phone Application Analysis tool and the interface is depicted in Fig. 2. This tool is used for monitoring and profiling an application:

- ***Profiling***—evaluate either execution-related or memory-usage aspects of a mobile application;
- ***Monitoring***—evaluate the behavior of the application.

The output generated by this tool can be general (see Fig. 3) or in detail (see Fig. 4). The general output is a summary of all parameters that are measured while the detailed output contains graphs that present the application during the execution time.



ImageDecodingMainThread

Monitoring and profiling your application can help you diagnose performance problems and improve the quality of your application. To begin, choose one of the options below.

Monitoring (recommended)

○ App Analysis (analyzes performance and quality aspects of application)

Profiling

● Execution (evaluates application performance with advanced visual and code profiling)

▷ Advanced Settings

● Memory (evaluates memory allocation and texture usage)

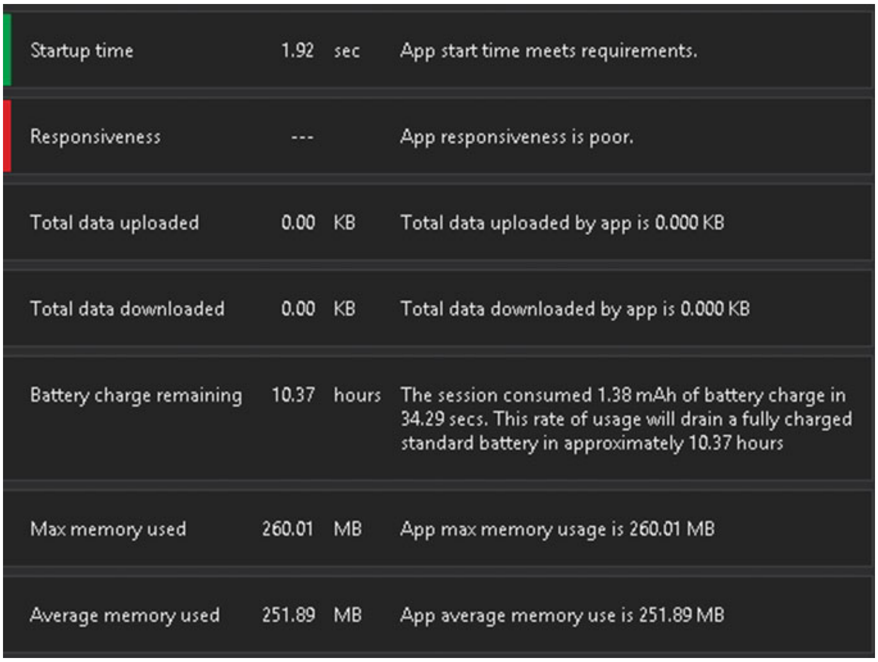
▷ Advanced Settings

Warning: The app performance observed on the emulator may not be indicative of the actual performance on the device

Warning: Set the solution configuration to release for more accurate app performance on the target device

Start Session (App will start)

Fig. 2 Windows Phone Application Analysis tool interface



🟩 Startup time	1.92	sec	App start time meets requirements.
🟥 Responsiveness	---		App responsiveness is poor.
Total data uploaded	0.00	KB	Total data uploaded by app is 0.000 KB
Total data downloaded	0.00	KB	Total data downloaded by app is 0.000 KB
Battery charge remaining	10.37	hours	The session consumed 1.38 mAh of battery charge in 34.29 secs. This rate of usage will drain a fully charged standard battery in approximately 10.37 hours
Max memory used	260.01	MB	App max memory usage is 260.01 MB
Average memory used	251.89	MB	App average memory use is 251.89 MB

Fig. 3 Windows Phone Application Analysis tool—general output

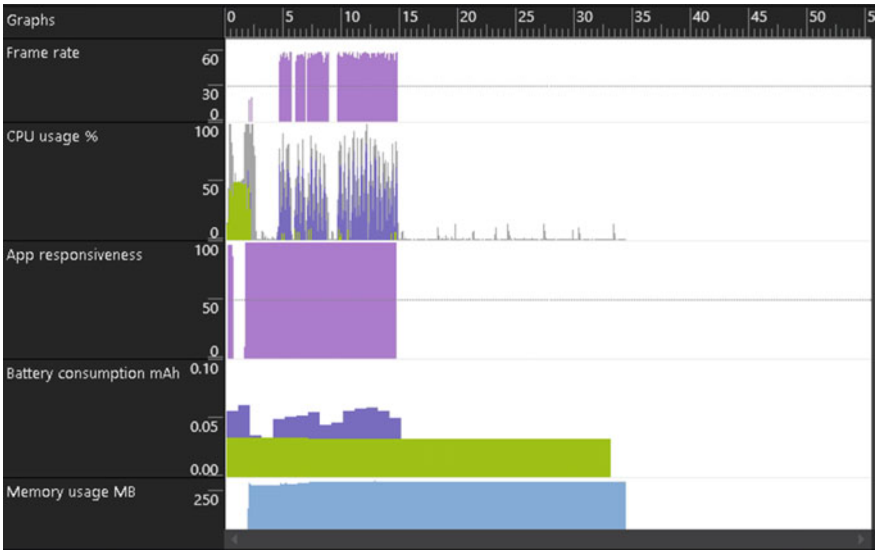


Fig. 4 Windows Phone Application Analysis tool—detailed output

3.3 Microsoft Expression Design 4

The last tool used for this thesis is Microsoft Expression Design 4, which is specialized in graphic design. It is used for complex objects that can be exported in different formats, like: XAML format or PNG format.

3.4 Experiments

The set of rules obtained is based on some common concepts that are used in programming or on the improvements that Microsoft brought into Windows Phone SDK. Oren Nachman, developer for Microsoft, said in one of his talks entitled "Windows Phone 8: Performance and Optimization for Developers" [37] that the performance of an application can be measured in "feelings". This means that a user who uses an application feels that the application is fast, that every action is processed immediately, that scrolling through pictures will not block the application and that navigating through pages is really smooth. This is the reason developers are focusing a lot on these aspects and try to optimize them. Also, the tools that are used by developers offer new controls that should be faster, more responsive and consume less memory. One aspect that is not always taken into consideration when a mobile application or a new control is developed is the battery consumption. The method chosen for this research is an experimental method. According to Oxford dictionary, an experiment is "a scientific procedure undertaken to make a discovery, test a hypothesis or demonstrates a known fact". This method is the most suitable for our research because currently only assumptions are made about whether the new controls are more efficient than the old ones, or whether one concept is more efficient than another one.

Experiments: On Application-Related Components
The main criterion that is applied in the selection of the elements that constitute the experiments is the diversity in terms of application's components. It is very important to have at least one element from each component of a mobile application tested.

The basic structure of a mobile application contains three components:

- **Frontend component** or the User Interface—it refers to the controls that are displayed to the user;
- **Backend component**—it refers to all the processing made by an application: data processing, command handlers and services connections;
- **Web services component**—it refers to all the services that are stored on servers, and which expose the Create/Read/Update/Delete functionality.

Accordingly, we can group the elements listed above in the following three groups (Table 1).

Table 1 Experimental elements

Frontend components	VirtualizedStackPanel [38], StackPanel [39], ListBox [40], LongListSelector [41], ProgressBar [42], Opacity [43], Visibility [44], Storyboard [45], Image background creation, background property [46]
Backend components	Assembly, recursive function, iterative function, page constructor, onNavigatedTo event for [47], Thread, multithread [48], while [49], base64 string format [50], Image build action [51], synchronous loading, asynchronous loading, image decoding [52], image format: PNG [53], JPG [54], XAML [55]
Web services components	Clouds [56]

Hypotheses

After the decision has been made on experiments in this research, the next step is to identify the hypothesis. Due to the fact that the controls and concepts that are to be tested, are used in different contexts, it is impossible to have only one hypothesis. For this reason, the components are grouped based on their functionalities and followed by the formulation of a hypothesis for each group. Based on these groups a number of 25 hypothesis have been derived, tested and discussed in this paper. The hypotheses are presented in Table 2.

For each of these experiments one or two applications are created and executed. These applications are executed several times and an average value is shown as the final result. For collecting the results the Windows Phone Application Analysis software is used. The data collected are: battery charge remaining, the execution time and the battery consumption. After obtaining the battery consumption, the energy consumption is calculated using the following formula:

$$E = QV$$

where E is energy (Wh), Q is charge (Ah), and V is Voltage (V).

The value for voltage depends on the phone that we are using. Consequently, the voltage for a specific phone: Nokia Lumia 1320 is 3.7 V.

Experiment Configurations

The experiments for this study are device dependent. This means that the collected results are specific for a device. The configurations that are used for the experiments can be found in the following Table 3.

As it can be noted in the above table the only phone dependent values are: the battery and screen resolution. This means that we should obtain different numbers for different emulators, but the rules obtained are universal (i.e. can be applied to any device). Three threads are being measured: UI thread, application thread and network thread. The UI thread phone-dependent because it is dependent on the resolution screen. The battery properties are important for the transformation of battery consumption to energy consumption. Since the battery is the same type for a specific device it does not influence the final result. All the experiments are tested on an emulator. The interface of the emulator looks depicted in Fig. 5.

Table 2 Hypotheses for the experiments

No.	Hypotheses
1.	The darker colors used as background for a mobile application consume less energy than the brighter ones
2.	A JPG file format consumes less energy than a PNG file format in a mobile application
3.	Storing a visual object as image consumes less energy than storing the same object as XAML
4.	Using background threads consumes less energy than using the UI thread
5.	A static object consumes less energy than an animated object
6.	Using image decoder to size consumes less energy than using the default decoder
7.	Using asynchronous methods consumes less energy than using synchronous methods
8.	Using “Visibility” property consumes less energy than using “Opacity” property
9.	Using a determinate progress bar consumes less energy than using an indeterminate progress bar
10.	Using a “LongListSelector” control consumes less energy than using a “ListBox” control
11.	Setting “Build type” property to “Resource” for an image, consumes less energy than setting the same property to “Content”
12.	Storing a set of images in JPG format consumes less energy than storing the same images as base64 format
13.	A “for” loop consumes less energy than a “while” loop
14.	Using several threads to complete an operation consume less energy than using one thread to complete the same operation
15.	Executing a heavy processing operation in constructor consumes less energy than executing the same operation in “OnNavigateTo” event
16.	Using an iterative function consumes less energy than using a recursive function
17.	Using a “StackPanel” control consumes less energy than using a “VirtualizingStackPanel” control
18.	Using one assembly, for storing the resources, consumes less energy than using several assemblies
19.	An animated object that is created in the XAML file consumes less energy than an animated object that is created in procedural code
20.	An image stored locally consumes less energy than an image stored in the clouds
21.	A video file stored locally consumes less energy than an image stored in the clouds
22.	An audio file stored locally consumes less energy than an image stored in the clouds
23.	A JPG file format stored in clouds consumes less energy than a PNG file format stored in clouds
24.	Downloading an image and access it locally consumes less energy than accessing the picture multiple times in clouds
25.	Processing an operation locally consumes less energy than processing the same operation in clouds

Table 3 Device configuration

Property	Value
Battery voltage	3.8 V
Nominal voltage	3.7 V
Battery type	BV-4BW
Emulator type	720p
Emulator resolution	1280 × 720
Brightness	100%

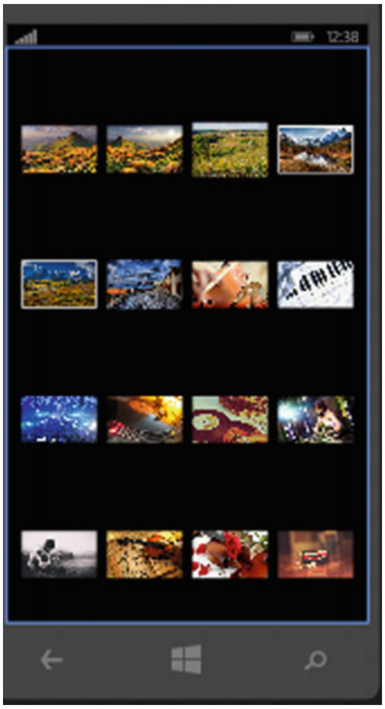


Fig. 5 Application snapshot

4 Results and Discussion

For each experiment there are two types of output: first output is a table which presents the duration of the experiment, the battery consumption, the energy consumption and an estimated value of the remaining battery life. The second output is a graph, which presents the distribution of battery consumption based on the main threads: UI thread, application thread and network thread. In order to obtain a result, several executions of the same experiment are made. This paper presents details of several experiments and a set of rules obtained.

4.1 Visual Object Storing

Aim To investigate the impact of storing a visual object as Extensible Application Markup Language (XAML) and as image on energy consumption (Table 4).
Based on the results of these experiments it can be concluded that it is more efficient to work with images than with XAML objects. The difference is not very big in terms of energy consumption, but if millions of applications that display images are considered, this can be a considerable improvement. Also from the user's experience point of view, it is a big improvement considering the fact that the battery will last longer. This difference occurs because when using XAML, the application will create an object for each tag and this can load the processor more, while in the case of image files the processor has to render an image that is stored locally and this will happen faster. For more complex objects the difference will grow. In Figs. 6 and 7 it is noted that the energy consumed by the UI thread (green color) is the same in both cases. The only noticeable difference is the energy consumed by the application thread. In this case, it can be concluded that more energy is required for creating the XAML object than to decode a picture.

4.2 Control Hiding

Aim To investigate the impact of "visibility" property and "opacity" property on energy consumption (Table 5).
Both the applications executed are the same, it is observed that the energy consumption is different. The difference is 0.07 mAh, which happens because the Opacity property will keep the rectangles in memory, in order to improve the speed of the application. Even though the application with enabled Opacity is faster, it costs more in terms of energy consumption. In the first graph (Fig. 8) it is noted that the energy consumption of the UI thread is lower because the objects are deleted. In the second case (Fig. 9), even if the objects cannot be seen on the screen, they are stored in memory so more energy is consumed. From Figs. 8 and 9 some interesting facts have been observed. The energy consumed by the application thread (purple color) is similar in both cases. There are small differences, but not significant ones. The energy difference that appears in this experiment is related to the UI thread

Table 4 Visual object storing—energy consumption

	Time (s)	Battery consumption (mAh)	Battery charge remaining (h)	Energy consumption (Wh)
XAML format	10.50	0.28	15.90	0.001036
PNG format	10.34	0.25	16.41	0.000925

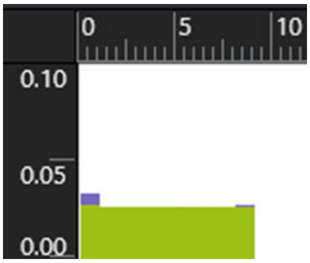


Fig. 6 XAML format

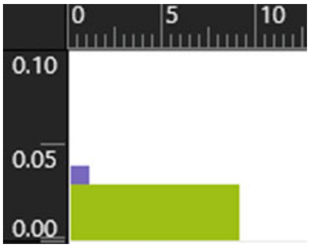


Fig. 7 PNG format

Table 5 Control hiding—energy consumption

	Time (s)	Battery consumption (mAh)	Battery charge remaining (h)	Energy consumption (Wh)
Visibility	20.71	1.26	6.83	0.004662
Opacity	20.63	1.33	6.44	0.004921

(green color). It can be seen in Fig. 8 that the UI thread consumes less energy while the objects are hidden. If the Opacity property is set, the energy consumed by the UI thread does not drop like in the previous case.

4.3 *Progress Bar Consumption*

Aim To investigate the energy efficiency of a determinate progress bar and an indeterminate progress bar (Table 6).

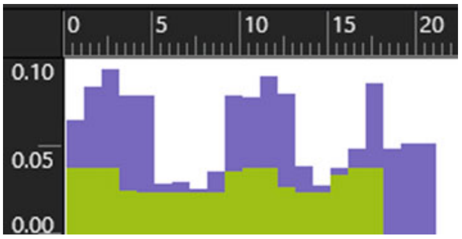


Fig. 8 Visibility property

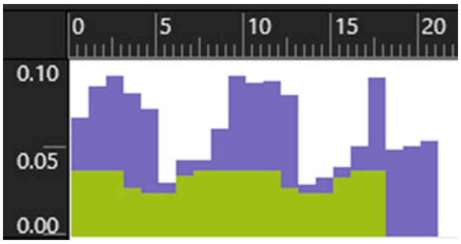


Fig. 9 Opacity property

Table 6 Progress bar—energy consumption

	Time (s)	Battery consumption (mAh)	Battery charge remaining (h)	Energy consumption (Wh)
Determinate	15.68	0.37	17.57	0.001369
Indeterminate	15.46	0.42	15.24	0.001554

As it can be seen from the charts above the determinate progress bar is more energy-efficient than the indeterminate one. This happens because the indeterminate bar is an animation which is shown all the time and which requires some processing. The determinate progress bar is based on a value so it does not require any repetitive pattern. This fact can be noticed in Figs. 10 and 11. The application thread (purple color) consumes more energy for an indeterminate progress bar because it supports the animation during the execution. In Fig. 10 it can be seen that it is required energy only when the application is launched. The UI thread (green color) consumes the same amount of energy in both cases. These controls can be used in different cases, but the determinate one ought to the preferred option.

4.4 Image Format

Aim To investigate the impact of displaying a set of images that is in a JPG (Joint Photographic Experts Group) format or in a base64 string format (Table 7).

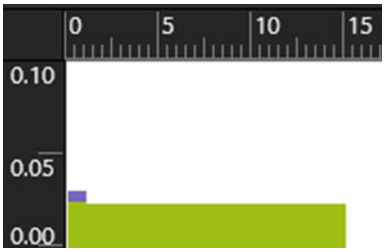


Fig. 10 Determinate progress bar

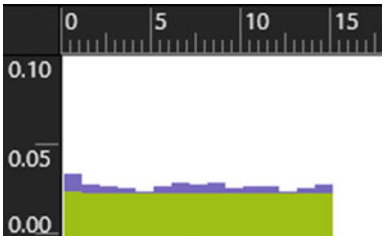


Fig. 11 Indeterminate progress bar

Table 7 Image format—energy consumption

	Time (s)	Battery consumption (mAh)	Battery charge remaining (h)	Energy consumption (Wh)
JPG	11.68	0.30	15.99	0.00111
Base64	11.30	0.30	15.90	0.00111

The battery consumption is equal in the both cases considered above, so it is not relevant if images are kept as JGP or as strings. Although the battery consumption is equal, it can noticed the fact that the distribution of application thread is different. In Fig. 12, it can be seen that it requires a lot of energy for computation (purple color) at the beginning, but after that it drops significantly. In the second case, it can be seen that the time for all the computation is longer. The energy consumed by UI thread (green color) is similar in both cases (Fig. 13).

4.5 Loop Instructions

Aim To investigate the energy efficiency of two loops instructions: for and while. Based on the results in Table 8, there is no difference between these two commands. This happens because, as previously mentioned, the only difference between the two instructions is the syntax. From Figs. 14 and 15 it can be seen that the energy consumption distribution of both UI thread (green color) and application thread (purple color) are the same in both cases.

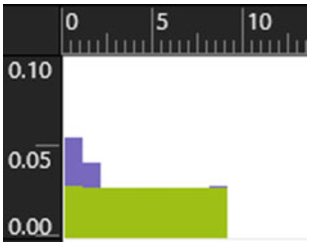


Fig. 12 Base64 format

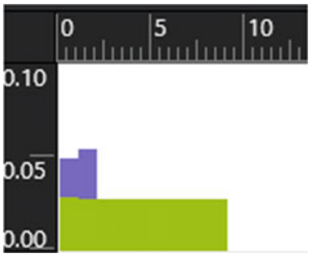


Fig. 13 JPG format

Table 8 Loop instruction—energy consumption

	Time (s)	Battery consumption (mAh)	Battery charge remaining (h)	Energy consumption (Wh)
For	21.67	0.56	16.10	0.002072
While	21.73	0.56	16.12	0.002072

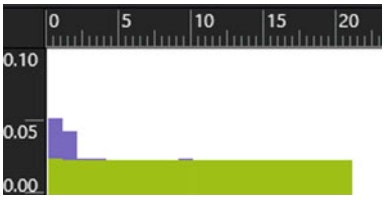


Fig. 14 For loop

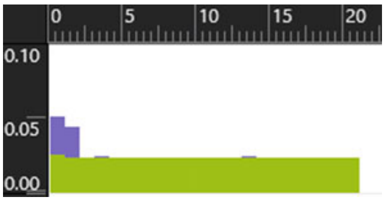


Fig. 15 While loop

4.6 Threads

Aim To investigate the energy efficiency of an application that uses one thread and of an application that uses more threads (Table 9).

Table 9 Threads—energy consumption

	Time (s)	Battery consumption (mAh)	Battery charge remaining (h)	Energy consumption (Wh)
Single thread	53.33	1.98	11.23	0.007326
Multithread	52.32	1.26	16.58	0.004662

As it can be seen from the charts above, the difference between the two approaches is significant. From Figs. 16 and 17 it can be noticed that the energy used by the UI thread (green color) is the same in both cases. There is a big difference in the application thread (purple color). For the single thread, it can be observed that it has required a lot of time to calculate all the numbers, which means a lot of energy wasted because the CPU is working. In the second case, the energy consumed by the application is very small because all the computations are done during the same time, in different threads. In the first case, 25 s are needed for processing while in the second case the results are shown immediately (Figs. 18 and 19).

4.7 Function Type

Aim To investigate the energy efficiency of an application that uses an iterative function compared to an application that uses a recursive function (Table 10).

The application that uses an iterative function is more efficient according to the graphs above. It is noticed that the recursive function requires more time to compute and it also consumes more energy (purple color). Moreover the user has to wait until all the results are loaded before the application can be used. In the case of the iterative function the amount of energy that is required is very low. Furthermore, it is noticed that in this case, the application is faster due to the fact that the thread is busy for a shorter span. The energy consumed by the UI thread (green color) is similar in both cases.

4.8 Storing Images

Aim To investigate the impact of displaying a set of images that is stored locally in comparison with a set of images that are stored in a web page (Table 11).

Loading images from different sources has a big impact on the total energy consumed by a mobile application. The application that stores the images locally consumes less energy than an application that requests the images from a web page. From Figs. 20 and 21 it can be noticed the fact that the UI thread (green) and the CPU thread (purple) consume the same amount of energy in both applications. The difference between the applications is made by the network (gray): the experiment

Fig. 16 Single thread

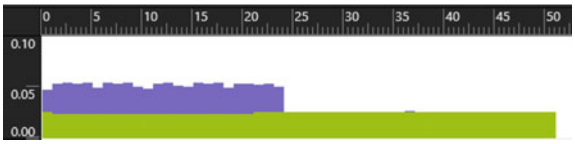


Fig. 17 Multithreading

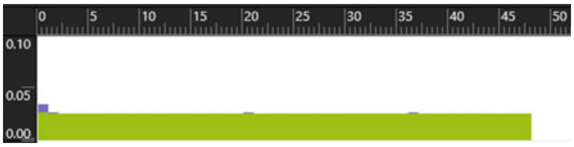


Fig. 18 Iterative function

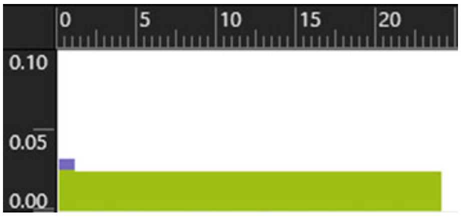


Fig. 19 Recursive function

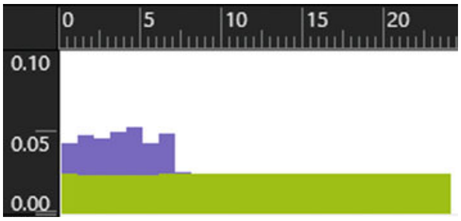


Table 10 Function type—energy consumption

	Time (s)	Battery consumption (mAh)	Battery charge remaining (h)	Energy consumption (Wh)
Iterative	25.28	0.61	17.29	0.002257
Recursive	26.73	0.77	14.55	0.002849

Table 11 Storing images—energy consumption

	Time (s)	Battery consumption (mAh)	Battery charge remaining (h)	Energy consumption (Wh)
From internet	21.96	0.92	10.00	0.003404
Stored locally	21.43	0.69	13.00	0.002553

presented in Fig. 20 shows there is no energy consumed by the network while the one in Fig. 21 shows a significant amount of energy that is consumed by the network.

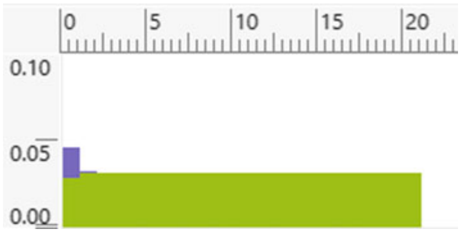


Fig. 20 Image stored locally

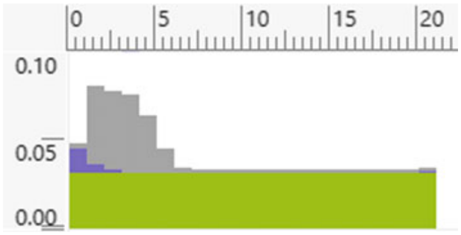


Fig. 21 Image downloaded

4.9 Image Format (JPG and PNG) in Clouds

Aim To investigate the impact of displaying a PNG (Portable Network Graphics) file format and a Joint Photographic Experts Group (JPG) file format, that is stored on a web page, on energy consumption.

This experiment reveals the fact that working with JPG format is "greener" than working with PNG format, if the images are stored on a website. From Table 12, it can noticed that the difference between these two formats is significant. In Figs. 22 and 23, it can be observed that the difference in the consumed energy is made by the network thread (gray). The UI thread (green) and the application thread (purple) have similar values. The distribution of the energy consumed by these two threads is also similar. The energy consumed by the network thread differs because of the images' file sizes. After the transformation from JPG in PNG (using http://image.online-convert.com/convert-to-png website), the files stored as PNG have a bigger size than the JPG files, and that is why the application that displays the PNG files consumes more energy.

Table 12 Image format in clouds—energy consumption

	Time (s)	Battery consumption (mAh)	Battery charge remaining (h)	Energy consumption (Wh)
JPG	21.01	0.74	11.90	0.002738
PNG	25.36	1.09	9.67	0.004033

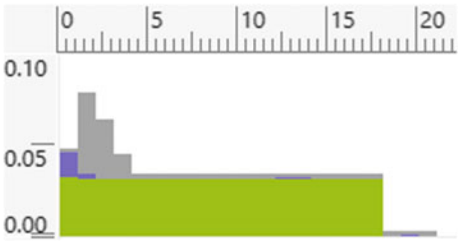


Fig. 22 JPG file format

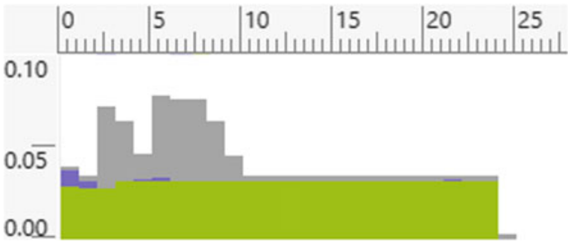


Fig. 23 PNG file format

4.10 Images—Multiple Access

Aim To investigate the impact on energy consumption of displaying multiple times the same picture from a web sites and the impact on energy consumption of downloading a picture and displaying it from a local source (Table 13).

From this experiment it can be noticed that the application which displays the images without saving them consumes less energy than the application which first downloads the picture. Figures 24 and 25 show that the energy consumed by the UI

Table 13 Multiple access—energy consumption

	Time (s)	Battery consumption (mAh)	Battery charge remaining (h)	Energy consumption (Wh)
From the same URL	31.28	0.96	13.54	0.003552
Download and display locally	31.19	1.02	12.74	0.003774

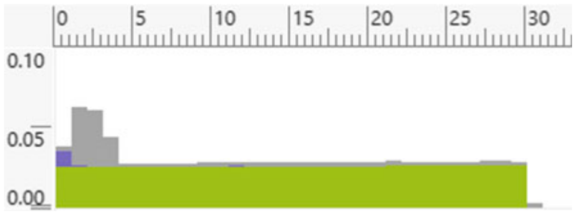


Fig. 24 Application that display the image from the same URL

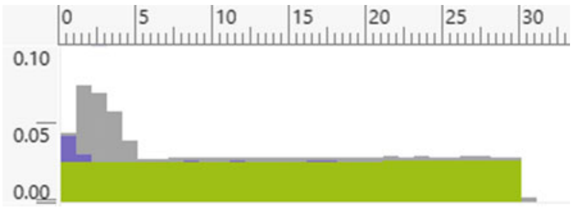


Fig. 25 Application that save and display the images

thread (green) is similar in both cases. The energy consumed by the application thread (purple) differs in these cases because it requires extra processing for saving the picture. The network thread (gray) consumes, also, less energy in the first case. Another fact that can be noticed is that each application makes a single request for the picture. In the first application this happens because of the cache mechanism that is implemented by default in Windows Phone 8. In the second case there is one request because we are downloading the image and using it after that from a local source.

4.11 Heavy Processing Operation

Aim To compare the impact on energy consumption of an operation that is run locally to an operation that is run in clouds.

The execution of some operations can significantly influence the energy consumption of an application. It can be seen in this experiment that executing some operations locally can save a lot of energy. From Table 14 it is noted that the difference between these two applications is significant. Figures 26 and 27 show that the UI thread (green) consumes the same amount of energy in both cases. In Fig. 27, the application thread (purple) request some energy only at the beginning while processing the data. For the other application the application thread consumes

Table 14 Heavy processing operation—energy consumption

	Time (s)	Battery consumption (mAh)	Battery charge remaining (h)	Energy consumption (Wh)
Cloud	42.34	1.73	10.23	0.006401
Locally	40.08	1.02	16.41	0.003774

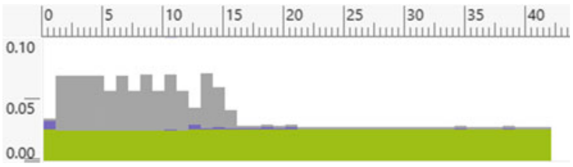


Fig. 26 Cloud processing

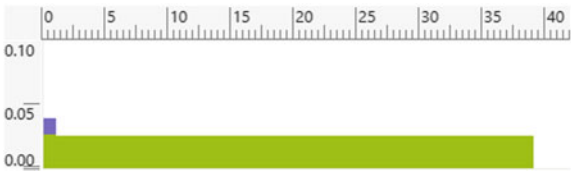


Fig. 27 Local processing

energy during the execution of the application because the data received from server has to be processed. The network thread (gray) shows a difference between these two applications, because in the first case there is a significant amount of energy consumed by this thread, while in the second case, the energy consumed by the network thread is 0.

4.12 Decoding Threads

Aim To investigate the impact of displaying images using backgrounds threads and using the UI thread on energy consumption (Figs. 28 and 29).

This experiment shows that the energy consumed by these two applications is different. From Table 15, it is noted that decoding an image in a separate thread is more efficient than using only one thread. Regarding the energy distribution, it can be seen that UI thread (green color) generates the same amount of energy in both cases while the application thread (purple color) generates less energy when using background threads. Another fact that can be noticed in the charts is the processing time. In the first case the application thread is working for 15 s while in the second case the application thread is working for 7 s. This happens because when using more than one thread, the tasks are executed in a parallel way. When all the processing is made by one thread it takes more time to decode all the pictures.

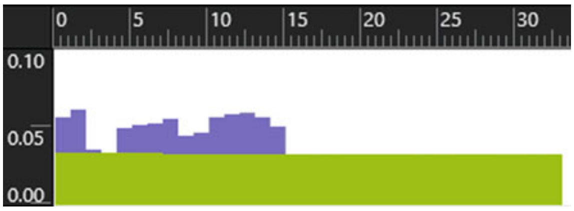


Fig. 28 CreateOption attribute set to BackgroundCreation

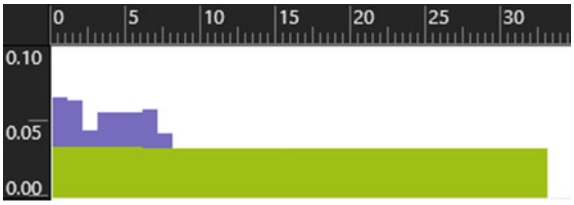


Fig. 29 Without CreateOption attribute set

Table 15 Decoding threads—energy consumption

	Time (s)	Battery consumption (mAh)	Battery charge remaining (h)	Energy consumption (Wh)
With CreateOption attribute	33.49	1.27	10.96	0.004699
Without CreateOption attribute	34.19	1.38	10.37	0.005106

4.13 Animations

Aim To investigate the energy efficiency of an application that displays an animation created in XAML file compared to an application that displays an animation created in procedural code (Table 16).

The charts above show that the energy consumption of the two applications is the same. This happens because the animation is the same in the both cases. Consequently the energy consumed is equal. It is noted that running the animation in the composition thread or in the UI thread gives the same effect. It might be possible to find some differences if the UI thread is overloaded. From Figs. 30 and 31 it is observed that both the UI thread (green color) and application thread (purple color) have a similar distribution of the consumed energy and of the amount of energy consumed.

From the total number of 25 experiments [1, 2], the assumed hypothesis was true in 14 cases. The hypothesis is not relevant in 5 experiments and it is false in 4 cases.

Table 16 Animations—energy consumption

	Time (s)	Battery consumption (mAh)	Battery charge remaining (h)	Energy consumption (Wh)
XAML	10.80	0.29	15.51	0.001073
Procedural code	10.51	0.29	15.30	0.001073

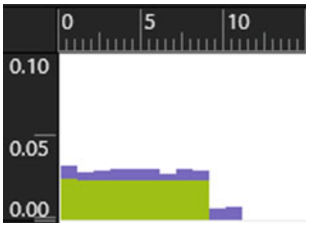


Fig. 30 Code behind

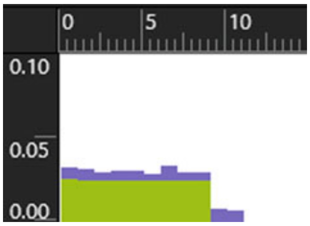


Fig. 31 XAML format

Table 17 Rules obtained after running the experiments

Hypotheses	Status	Rule
Hypothesis no. 1	Confirmed	Use darker colors in Windows Phone applications
Hypothesis no. 2	Not relevant	The PNG or JPG file format does not influence the energy consumption of a mobile application
Hypothesis no. 3	Confirmed	Use PNG format instead of XAML format for displaying images
Hypothesis no. 4	Confirmed	Use “CreateOption” attribute for all the pictures
Hypothesis no. 5	Confirmed	Use static objects instead of animated ones as much as possible
Hypothesis no. 6	Confirmed	Use decoder to size when the dimension of the image control is known
Hypothesis no. 7	Confirmed	Use asynchronous loading for pictures
Hypothesis no. 8	Confirmed	Use Visibility property for hiding an object instead of Opacity property
Hypothesis no. 9	Confirmed	Choose a determinate progress bar if the context allows this
Hypothesis no. 10	Rejected	For the basic use of a list use a “ListBox” control
Hypothesis no. 11	Confirmed	Use “Resource” value when developing mobile applications
Hypothesis no. 12	Not relevant	Either JPG format or Base64 format can be used for displaying pictures
Hypothesis no. 13	Not relevant	Either “for” or “while” loop can be used in developing a “green” application
Hypothesis no. 14	Confirmed	Use multi-threads in a mobile application
Hypothesis no. 15	Rejected	Use “OnNavigateTo” method for data initialization
Hypothesis no. 16	Confirmed	Use iterative functions instead of recursive ones
Hypothesis no. 17	Rejected	Use “VirtualizingStackPanel” inside “ItemsControls” elements
Hypothesis no. 18	Not relevant	Either storing the resources in a different assembly or in the same assembly, the energy consumption is the same

(continued)

Table 17 (continued)

Hypotheses	Status	Rule
Hypothesis no. 19	Not relevant	An animated object can be created either in XAML file or in procedural code
Hypothesis no. 20	Confirmed	User images stored locally
Hypothesis no. 21	Inconclusive	–
Hypothesis no. 22	Inconclusive	–
Hypothesis no. 23	Confirmed	Use JPG format if the picture are stored in clouds
Hypothesis no. 24	Rejected	Access the images directly from web service rather than downloading them
Hypothesis no. 25	Confirmed	Process data locally

Two hypotheses are inconclusive. Table 17 presents a summary of the results obtained from these experiments (note the third column is the energy efficiency rule).

5 Conclusions

Developing a mobile application has to be based on the user experience. Nowadays a user expects an application that is fast and responds to any input. The battery consumption is another aspect which is really important for a user, but which is associated most of the times with the phone and not with an application. It is true that the energy consumption of an application is not the same for two different mobile phones, but most of the energy consumption is application dependent. This study reveals the fact that there are some concepts, such as single threading, which consumes more energy than similar concepts (multithreading), which give the same output. For a developer it is very important to choose the right approach in order to offer the user the best experience when using an application and a longer battery life.

The second reason for this study is sustainability. Each experiment shows the energy consumed by each tested concept or control. The value obtained can be used for calculating the total impact that an application can have on the environment. This is an important aspect because nowadays ICT produces 2% from the total energy consumed in the world. This percentage will grow, because the ICT domain is in a continuous development, so it is very important to reduce the energy in all the aspects. There are studies conducted in this domain, but most of them focus on Android phones or on iOS phones. Windows Phone is not very popular at the moment, but, according to the sources presented in the Introduction section, there will be an increase in the next years. One aspect that could be very interesting to study is the energy consumption of each operating system and to compare the differences between them. Another future work will be exploring the relationship between energy consumption and different hardware components on the same platform. For example, it would be interesting to know the relationship between the energy consumption and the size of the screen, or the screen type. This study could help the producers to choose the right components for the future models of phones.

Acknowledgements This work was supported by the Erasmus Mundus PERCCOM programme.

References

1. Cristea, V.-V., Pattinson, C., Kor, A.L.: Energy Consumption of Mobile Phones. In: International SEEDS Conference, 17–18 Sept 2015, Leeds
2. Cristea, V.: Energy consumption of mobile phones. Unpublished Masters Degree Dissertation, Leeds Beckett University, UK (2015)
3. Kharchenko, V., Kondratenko, Y., Kacprzyk J. (eds.): Green IT Engineering: Concepts, Models, Complex Systems Architectures. Studies in Systems, Decision and Control, vol. 74. Springer, Cham (2017a). https://doi.org/10.1007/978-3-319-44162-7
4. Kharchenko, V., Kondratenko, Y., Kacprzyk, J. (eds.): Green IT Engineering: Components, Networks and Systems Implementation. Studies in Systems, Decision and Control, vol. 105. Springer, Cham (2017b). https://doi.org/10.1007/978-3-319-55595-9
5. Kondratenko, Y., Korobko, V., Korobko, O., Kondratenko, G., Kozlov, O.: Green-IT approach to design and optimization of thermoacoustic waste heat utilization plant based on soft computing. In: Kharchenko, V., Kondratenko, Y., Kacprzyk, J. (eds.) Green IT Engineering: Components, Networks and Systems Implementation. Studies in Systems, Decision and Control, vol. 105, pp. 287–311. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55595-9_14
6. Kuchuk, G., Kovalenko, A., Kharchenko, V., Shamraev, A.: Resource-oriented approaches to implementation of traffic control technologies in safety-critical I&C systems. In: Kharchenko, V., Kondratenko, Y., Kacprzyk, J. (eds.) Green IT Engineering: Components, Networks and Systems Implementation. Studies in Systems, Decision and Control, vol. 105, pp. 313–337. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55595-9_15
7. Emarketer.com: Smartphone Users Worldwide Will Total 1.75 Billion in 2014—eMarketer. [online] Available at: http://www.emarketer.com/Article/Smartphone-Users-Worldwide-Will-Total-175-Billion-2014/1010536. Accessed 15 Apr 2015
8. www.statista.com: Topic: App Stores. [online] Available at: http://www.statista.com/topics/1729/app-stores/. Accessed 15 Apr 2015 (2014)
9. Protalinski, E.: Microsoft Confirms Windows Phone Store Passed 300,000 Apps. The Next Web. [online] Available at: http://thenextweb.com/microsoft/2014/08/08/microsoft-confirms-windows-phone-store-300000-apps/. Accessed 15 Apr 2015 (2014)
10. news.microsoft.com: Microsoft by the numbers. [online] Available at: http://news.microsoft.com/bythenumbers/ms_numbers.pdf. Accessed 15 Apr 2015
11. Webb, M.: SMART 2020: Enabling the Low Carbon Economy in the Information Age (2008)
12. GeSI: Smarter2030: ICT Solutions for 21st Century Challenges. URL: http://gesi.org/report/detail/smarter-2030. Accessed 20 Mar 2018 (2015)
13. Pathak, A., Hu, Y., Zhang, M.: Where is the energy spent inside my app? In: Proceedings of the 7th ACM European Conference on Computer Systems—EuroSys '12 (2012)

14. Hao, S., Li, D., Halfond, W., Govindan, R.: Estimating mobile application energy consumption using program analysis. In: 2013 35th International Conference on Software Engineering (ICSE) (2013)
15. Jung, W., Kang, C., Yoon, C., Kim, D., Cha, H.: DevScope. In: Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis—CODES + ISSS '12 (2012)
16. Carroll, A., Heiser, G.: An Analysis of power consumption in a smartphone. In: Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference, pp. 21–21, Boston, MA, 23–25 June 2010
17. Corral, L., Georgiev, A., Sillitti, A., Succi, G.: A method for characterizing energy consumption in Android smartphones. In: 2013 2nd International Workshop on Green and Sustainable Software (GREENS) (2013)
18. Xia, F., Hsu, C., Liu, X., Liu, H., Ding, F., Zhang, W.: The power of smartphones. Multimedia Syst. **21**(1), 87–101 (2013)
19. Carroll, A., Heister, G.: An Analysis of Power Consumption in a Smartphone (2010)
20. Hahnel, M., Dobel, B., Volp, M., Hartig, H.: Measuring energy consumption for short code paths using RAPL. ACM SIGMETRICS Perform. Eval. Rev. **40**(3), 13 (2012)
21. Namboodiri, V., Ghose, T.: To cloud or not to cloud: a mobile device perspective on energy consumption of applications. In: 2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM) (2012)
22. Wilke, C., Piechnick, C., Richly, S., Poschel, G., Gotz, S., Abmann, U.: Comparing mobile applications' energy consumption. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing—SAC '13 (2013)
23. Andreucetti, R., Chen, S., Yuan, Z., Muntean, G.: Smartphone energy consumption of multimedia services in heterogeneous wireless networks. In: 2014 International Wireless Communications and Mobile Computing Conference (IWCMC) (2014)
24. Malik, M.Y.: Power Consumption Analysis of a Modern Smartphone. URL: https://arxiv.org/abs/1212.1896. Accessed 20 Mar 2018 (2012)
25. Lee, M., Kim, D., Lee, J.: Analysis of characteristics of power consumption for context-aware mobile applications. Information **5**(4), 612–621 (2014). https://doi.org/10.3390/info5040612
26. Yurur, O., Liu, C., Moreno, W.: Modeling battery behavior on sensory operations for context-aware smartphone sensing. Sensors (14248220) **15**(6), 12323–12341 (2015). https://doi.org/10.3390/s150612323
27. Abreu, D.P., Villapol, M.E.: Measuring the energy consumption of communication interfaces on smartphones using a moderately-invasive technique. In: Global Information Infrastructure and Networking Symposium (GIIS) 2012, 1–6 Dec 2012, Choroni, Venezuela. https://doi.org/10.1109/giis.2012.6466662
28. Parkkila, J., Porras, J.: Improving battery life and performance of mobile devices with cyber foraging. In: 2011 IEEE 22nd International Symposium on Personal, Indoor and Mobile Radio Communications (2011)
29. Drouant, N., Rondeau, E., Georges, J., Lepage, F.: Designing green network architectures using the ten commandments for a mature ecosystem. Comput. Commun. **42**, 38–46 (2014)
30. Pattinson, P., Robinson, L.: A study of the effectiveness of "Wake up on LAN" as a means of power management. In: ICE-B 2008, pp. 73–76 (2008)
31. Li, S., Mishra, S.: Optimizing power consumption in multicore smartphones. J. Parallel Distrib. Comput. **95**, 124–137 (2016). https://doi.org/10.1016/j.jpdc.2016.02.004
32. Ukhanova, A., et al.: Power consumption analysis of constant bit rate video transmission over 3G networks. Comput. Commun. **35**(14), 1695–1706 (2012). https://doi.org/10.1016/j.comcom.2012.05.010
33. Elliott, J., Kor, A.L., Omotosho, A.: Energy consumption in smartphones: an investigation of battery and energy consumption of media related applications on android smartphones. In: International SEEDS Conference, 2017, Leeds, UK
34. Microsoft: Windows Phone 8.1 Mobile Device Management Overview. Available at: https://www.microsoft.com/en-gb/download/details.aspx?id=42508. Accessed date: 15 Apr 2015 (n.d.)

35. Msdn.microsoft.com: Welcome to Visual Studio 2013 [online]. Available: https://msdn.microsoft.com/en-us/library/dd831853(v=vs.120).aspx. Accessed date: 15 Apr 2015
36. Msdn.microsoft.com: Visual Studio Express 2012 for Windows Phone 8. [online] Available at: https://msdn.microsoft.com/en-us/library/windows/apps/ff630878(v=vs.105).aspx. Accessed 15 Apr 2015
37. Channel 9: Windows Phone 8: Performance & Optimization for Developers (Channel 9). [online] Available at: http://channel9.msdn.com/Events/Build/2012/3-048. Accessed 15 Apr 2015 (2012)
38. Msdn.microsoft.com: VirtualizingStackPanel Class—Windows App Development. [online] Available at: https://msdn.microsoft.com/en-us/library/windows/apps/windows.ui.xaml.controls.virtualizingstackpanel. Accessed 15 Apr 2015
39. Msdn.microsoft.com: StackPanel Class (System.Windows.Controls). [online] Available at: https://msdn.microsoft.com/en-us/library/system.windows.controls.stackpanel(v=vs.110).aspx. Accessed 15 Apr 2015
40. Msdn.microsoft.com: ListBox Class (System.Windows.Controls). [online] Available at: https://msdn.microsoft.com/en-us/library/windows/apps/system.windows.controls.listbox(v=vs.105).aspx. Accessed 15 Apr 2015
41. Msdn.microsoft.com: LongListSelector Class (Microsoft.Phone.Controls). [online] Available at: https://msdn.microsoft.com/en-us/library/windows/apps/microsoft.phone.controls.longlistselector(v=vs.105).aspx. Accessed 15 Apr 2015
42. Msdn.microsoft.com: ProgressBar Class (System.Windows.Controls). [online] Available at: https://msdn.microsoft.com/en-us/library/windows/apps/system.windows.controls.progressbar(v=vs.105).aspx. Accessed 15 Apr 2015
43. Msdn.microsoft.com: UIElement.Opacity Property (System.Windows). [online] Available at: https://msdn.microsoft.com/en-us/library/system.windows.uielement.opacity(v=vs.110).aspx. Accessed 15 Apr 2015
44. Msdn.microsoft.com: UIElement.Visibility Property (System.Windows). [online] Available at: https://msdn.microsoft.com/en-us/library/system.windows.uielement.visibility(v=vs.110).aspx. Accessed 15 Apr 2015
45. Msdn.microsoft.com: Storyboard Class (System.Windows.Media.Animation). [online] Available at: https://msdn.microsoft.com/en-us/library/system.windows.media.animation.storyboard(v=vs.110).aspx. Accessed 15 Apr 2015
46. Msdn.microsoft.com: Control.Background Property (System.Windows.Controls). [online] Available at: https://msdn.microsoft.com/en-us/library/system.windows.controls.control.background(v=vs.110).aspx. Accessed 15 Apr 2015
47. Msdn.microsoft.com: Page.OnNavigatedTo Method (System.Windows.Controls). [online] Available at: https://msdn.microsoft.com/library/system.windows.controls.page.onnavigatedto(VS.95).aspx. Accessed 15 Apr 2015
48. Msdn.microsoft.com: Multithread. [online] Available at: https://msdn.microsoft.com/en-us/library/3c8c4cxa.aspx. Accessed 15 Apr 2015
49. Msdn.microsoft.com: while (C# Reference). [online] Available at: https://msdn.microsoft.com/en-us/library/2aeyhxcd.aspx. Accessed 15 Apr 2015
50. Tools.ietf.org: RFC 4648—The Base16, Base32, and Base64 Data Encodings. [online] Available at: https://tools.ietf.org/html/rfc4648#section-4. Accessed 15 Apr 2015
51. Developers.de: Windows Phone 7: Content vs. Resource Build Action - Damir Dobric Posts - developers.de. [online] Available at: http://developers.de/blogs/damir_dobric/archive/2010/09/18/windows-phone-7-content-vs-resource-build-action.aspx. Accessed 15 Apr 2015
52. Msdn.microsoft.com: PictureDecoder.DecodeJpeg Method (Stream, Int32, Int32) (Microsoft.Phone). [online] Available at: https://msdn.microsoft.com/en-us/library/windows/apps/ff708027(v=vs.105).aspx. Accessed 15 Apr 2015
53. W3.org: Portable Network Graphics (PNG) Specification (Second Edition). [online] Available at: http://www.w3.org/TR/PNG/#1Scope. Accessed 15 Apr 2015

54. Whatis.techtarget.com: What is JPG? What Opens a JPG? File Format List from WhatIs.com. [online] Available at: http://whatis.techtarget.com/fileformat/JPG-JPEG-bitmap. Accessed 15 Apr 2015
55. Msdn.microsoft.com: What is XAML? [online] Available at: https://msdn.microsoft.com/en-us/library/cc295302.aspx. Accessed 15 Apr 2015
56. SearchCloudComputing: What is cloud computing?—Definition from WhatIs.com. [online]. Available at: http://searchcloudcomputing.techtarget.com/definition/cloud-computing. Accessed 15 Apr 2015