Leeds Beckett University

Faculty of Arts, Environment & Technology

PERCCOM Master Program

Master's Thesis in

Pervasive Computing & COMmunications

for sustainable Development

Cristea Vlad Vasile

Energy Consumption of Applications on Mobile Phones

2015

<u>Supervisor(s) :</u> Professor Colin Pattinson (Leeds Beckett University)

Doctor Ah-Lian Kor (Leeds Beckett University)

<u>Examiners:</u> *Eric Rondeau* (University of Lorraine)

Jari Porras (Lappeenranta University of Technology)

Karl Andersson (Luleå University of Technology)

This thesis is prepared as part of an European Erasmus Mundus programme PERCCOM - Pervasive Computing & COMmunications for sustainable development.

This thesis has been accepted by partner institutions of the consortium (cf. UDL-DAJ, n°1524, 2012 PERCCOM agreement).

Successful defence of this thesis is obligatory for graduation with the following national diplomas:

- Master in Master in Complex Systems Engineering (University of Lorraine)
- Master of Science in Computer Science and Engineering, Specialization: Pervasive Computing and Communications for Sustainable Development (Lulea University of Technology)
- Master of Science (Technology) in Computer Science (Lappeenranta University of Technology)

Cristea Vlad Vasile

Title: Energy Consumption of Mobile Phones

Faculty of Arts, Environment & Technology

Leeds Beckett University, Leeds LS1 3HE, United Kingdom

Pervasive Computing and Communications for Sustainable development Master Program

Examiners: Professor Eric Rondeau,

Professor Jari Porras,

Professor Karl Andersson

134 Pages, 25 Appendixes, 26 Charts, 108 Tables, 113 Figures

Keywords: energy efficiency, Windows Phone, smartphone's energy consumption

Abstract

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 an iOS operating systems. Furthermore, another reason is the increasing number of Windows Phone users. The objective of this research is to categorise the energy consumption parameters (e.g. use of one thread or several thread for the same output). The result for each group of experiment 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 by the experiment, the expected 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.

Acknowledgment

This work was supported by the PERCCOM programme and Erasmus+ programme.

I would like to express my gratitude to my supervisors Colin Pattinson and Ah-Lian Kor for the useful comments, remarks and engagement through the learning process of this master thesis.

Contents

1. Introduction

In recent years, the smartphones market had 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 (Emarketer.com, 2015). This ascending trend has determined the same publication to predict that the number of smartphone users will be around 2.5 billion in 2017. This means that around 30% from the world's population will own such a device. The main producers of smartphones in the last quarter of 2014, according to International Data Corporation (IDC) (www.idc.com, 2015) are: Samsung with 19.9% of the market, Apple with 19.7%, Lenovo with 6.5%, Huawei with 6.3% and Xiaomi with 4.4%. 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 was 76.6%, while the smartphones which support iOS represent 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. Staistica 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 can be a competitor for Android and iOS operating systems.

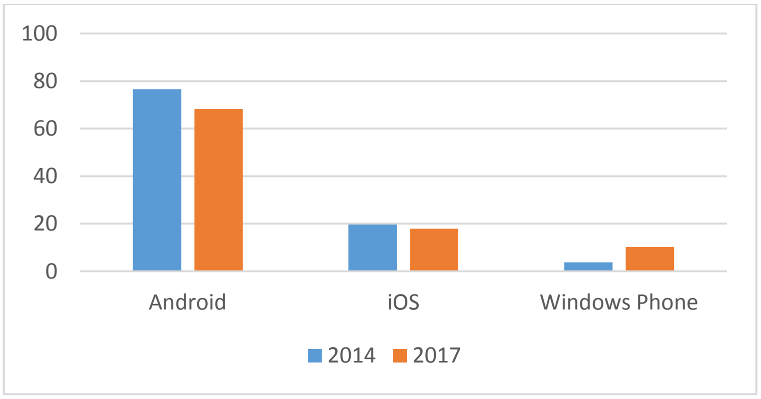


Chart 1 Operating systems distribution

According to Statistica portal in October 2014 (www.statistica.com, 2014) there were 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 (Protalinski, 2014) 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 newest statistics from Microsoft (news.microsoft.com, 2015), in March 2015, there was a number of 585.000 applications in Windows Store. It can be noticed that the increasing rate of applications' development is very high, promoting Windows Store to become a competitor for App Store and Google Play. This is the reason for the objective of this thesis: to analyze in details the concepts and controls used by the developers of Windows Phone.

According to Smart2020 report (Webb, 2008) the information technology and communication (ICT) consumes around 2% of the world's energy. This number can be compared to the total energy consumed by airline industry. The mobile phones will represent in 2020 1% from the ICT footprint and the mobile network will represent 13%. It is very difficult to calculate very precise 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 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. According to (Spectrum.ieee.org, 2015), the total amount of energy used by a smartphone in a year is bigger than the amount of energy consumed by two new Energy Star refrigerators in the same time frame. The smartphone's energy consumption is the second reason of this research.

A smartphone's battery is discharged by the applications that are used every day by the user. As it was mentioned before, the number of applications from stores is growing really fast and if developers neglect to optimize the battery consumption, the effect will be seen in the total energy consumption. The objective of this thesis 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 that wants an energy efficiency application. There will be a predefined number of rules that will be tested and which will cover the UI part, the processing part and the network part.

1.1. Aim and Research Objectives

The goal of this research is to create a set of 25 evidence-based rules that aim to improve the energy consumption of mobile phone applications. The following research objectives will help achieve this aim:

- Research Objective 1: Create a set of 25 hypotheses (tabulated in Table 2) relating to the front-end, back-end and web services of mobile phone applications.
- Research Objective 2: Write two applications for each hypothesis.
- Research Objective 3: Collect data, analyze and evaluate the energy consumption of each application.
- Research Objective 4: Evaluate the hypotheses tabulated in Table 2 based on findings in Research Objective 3.

1.2. Contributions

This thesis makes the following contribution:

- 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 the energy consumption of a mobile application.

1.3. Dissertation structure

This thesis has the following structure:

- Chapter 2 will present some researches that are related to the current one;
- Chapter 3 will discuss the methodology used for obtaining the results, the tools that were used and the concepts that were tested;
- Chapter 4 will contain a brief overview of each experiment and a general discussion about all the experiments;
- Chapter 5 will included the conclusions of this thesis and the future work;
- Appendix presents each experiment in details.

2. Related Work

The previous chapter introduces the topic of this thesis: an analysis of the energy consumption of different controls offered by Windows Phone SDK combined with different concepts used in programming. Smartphones' energy efficiency is a new research domain 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.

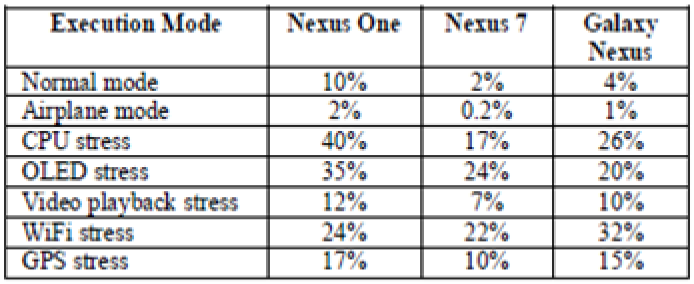
Related studies with the current paper are in the following directions: tools that measure energy consumption, comparisons between different network types, cloud services, and an overview analysis of an application.

2.1. Tools

The tool described in (Pathak, Hu and Zhang, 2102) shows how can be implemented a software that measure the energy consumption of an application. They validated this tool by analyzing the energy consumption of ten popular applications stored in Google Play, including Angry Birds, Facebook and Android browser. Their analyze shows that third-party advertisement module consumes between 65% and 75% of the total energy, the clean termination of long lived TCP sockets consume between 10% and 50% of the total energy, tracking user data consumes between 20% and 30% and the processing algorithms consume between 10% and 30% of the total energy. Another tool used for measuring the energy is called eLens (Hao et al., 2013) and combines program analysis and per-instruction energy modeling. In the same category it can be placed the tool called DevScope and described in (Jung et al., 2012).

2.2. Overall consumption

Measuring the energy consumption of an application can be done in two ways: using a multimeter or using a software. The first method is difficult implement and is not specific. Using the second approach, the paper (Corral et al., 2013) presents an overview of energy consumption for a mobile application. The objective of this paper is to stress different components of a smartphone and to see the amount of energy consumed. For this experiment were used three smartphones: HTC Nexus One cell phone, with Android 2.3.7, powered by a Li-Ion 1400 milliampere-hour (mAh) battery, Samsung Galaxy cell phone, with Android 4.0.4, powered by a Li-Ion 1750 mAh battery and Nexus 7 tablet, operated by Android 4.2.1, powered by a Li-Ion 4325 mAh battery. The results of this study are presented in Figure 1.



Execution Mode	Nexus One	Nexus 7	Galaxy Nexus
Normal mode	10%	2%	4%
Airplane mode	2%	0.2%	1%
CPU stress	40%	17%	26%
OLED stress	35%	24%	20%
Video playback stress	12%	7%	10%
WiFi stress	24%	22%	32%
GPS stress	17%	10%	15%

Figure 1 Percentage of battery discharged in 2 hours. (Corral et al., 2013)

A similar study was made in (Xia et al.,2013) for an iMate KJam smartphone and the results are the following ones: CPU - 35%, GSM - 25%, Wi-Fi – 25%, Backlight – 3%, Bluetooth – 7% and other – 5%. In (Chen et al., 2013) is presented a detailed study on the energy consumed by the display in different applications for Android Operating System.

Study (Carroll and Heister,2010) tries to measure the energy consumption of a mobile application by taking physical power measurements at the component level on a piece of real hardware. For this they used a Samsung S3 mobile phone. They took these measurements for different scenarios and the results obtained are the following:

- For audio playback: 58% of the power is consumed by the codec and 42% consumed by amplifier;
- For video playback: the CPU is the biggest single consumer of power;
- For text messaging: the power is consumed mostly by the display components;
- For a phone call: the GSM consumes the most part of the energy;
- For e-mailing: the GSM is the main energy consumer;
- For web browsing: most of the energy is consumed by GPRS.

Another approach in measuring the energy consumption is to measure each function in the application. This approach is presented in (Hahnel et al., 2012) using Running Average Power Limit and HAECER (Highly-Adaptive Energy-Efficient Systems – Energy Reader).

2.3. Cloud services

In this paper (Namboodiri and Ghose, 2012) there is an analysis of energy consumption for cloud and non-cloud services. For this experiment they used a HTC Desire smartphone with Android 2.1 operating system. They compared three types of applications: documents, video and chees, revealing the following conclusions: the cloud services are energy efficient for applications that are computation intensive only if they run locally and energy inefficient if the applications are computation intensive regardless where they run. A graphical representation of these results can be seen in Figure 2.

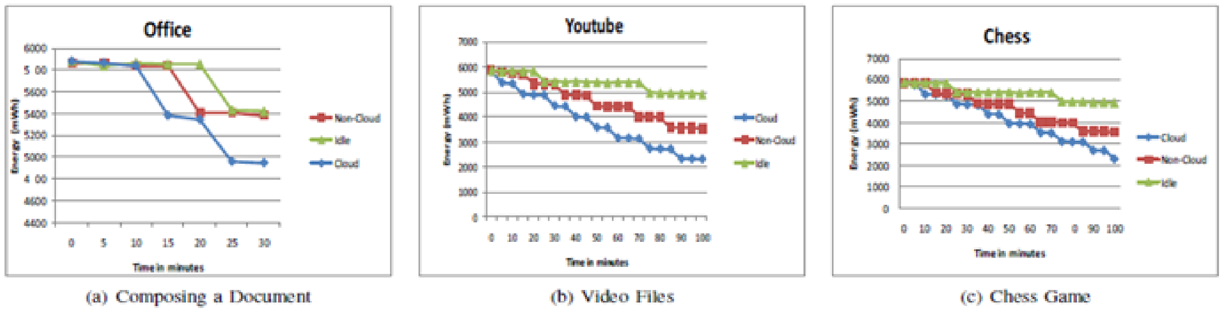


Figure 2 Battery capacity over time while composing a document, playing video files or games on a mobile phone (Namboodiri and Ghose, 2012)

2.4. Network measurements

This study (Metri et al., 2012) tests different aspects for an iPhone and for an Android phone. The tests that were made are presented in Figure 5. These tests show that, for both iPhone and Samsung, a Wi-Fi network consumes less energy than a 3G network. A detailed description of the iPhone's energy consumption can be found in Figure 3 and Figure 4.

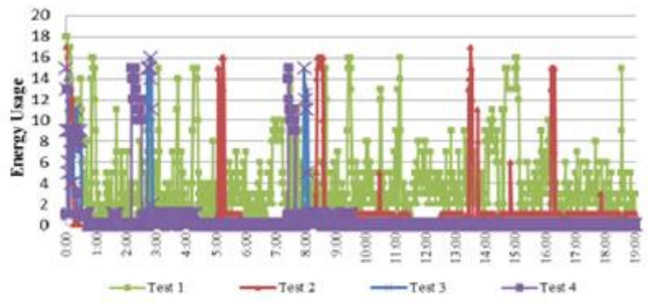


Figure 3 Energy usage of iPhone using Wi-Fi (Metri et al., 2012)

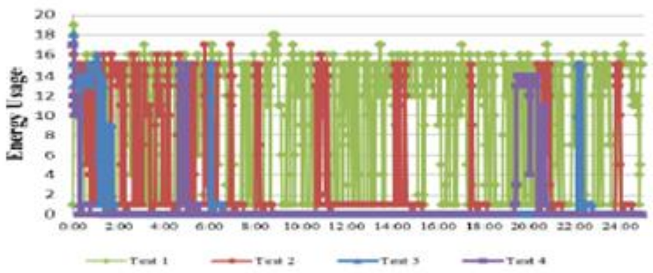
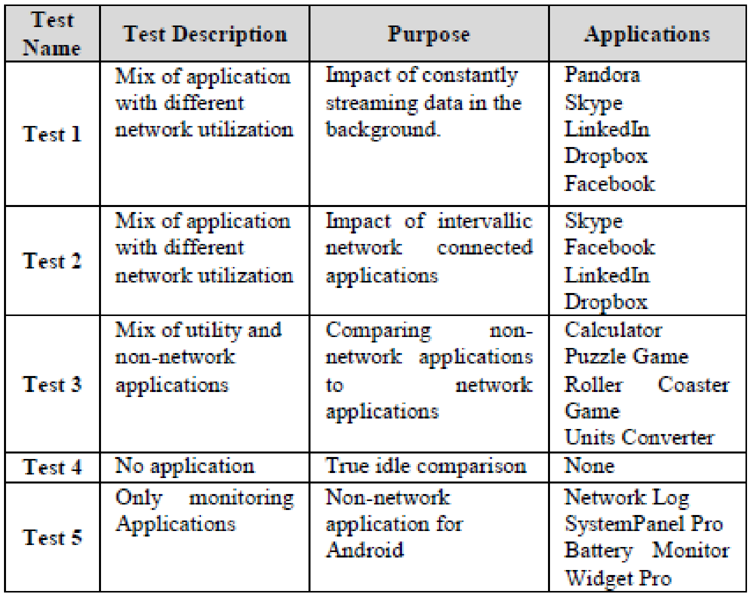


Figure 4 Energy usage of iPhone using 3G (Metri et al., 2012)



Test Name	Test Description	Purpose	Applications
Test 1	Mix of application with different network utilization	Impact of constantly streaming data in the background.	Pandora Skype LinkedIn Dropbox Facebook
Test 2	Mix of application with different network utilization	Impact of intervallic network connected applications	Skype Facebook LinkedIn Dropbox
Test 3	Mix of utility and non-network applications	Comparing non-network applications to network applications	Calculator Puzzle Game Roller Coaster Game Units Converter
Test 4	No application	True idle comparison	None
Test 5	Only monitoring Applications	Non-network application for Android	Network Log SystemPanel Pro Battery Monitor Widget Pro

Figure 5 Types of test (Metri et al., 2012)

This study (Wilke et al., 2013) measures the energy consumption of e-mailing and web browsing in different conditions. For e-mailing the following test cases are taken into consideration: setup mail account, drop mail account, check for mails, read, write, forward and delete mails. Each action which is related to e-mails was tested in the following conditions: a long email, a short email, a mail with a picture attached, with a text file attached and with an audio file attached. The applications tested are K9, MailDroid and MailDroidPro. The results of the tests are presented in Figure 6. In the second case, web browsing, the following situations were considered: open a web page, open an image, download a file, and performing a web search. All of these actions were made using three applications: Easy browser, NineSky browser and Droid Surfing, and the result are presented in Figure 7.

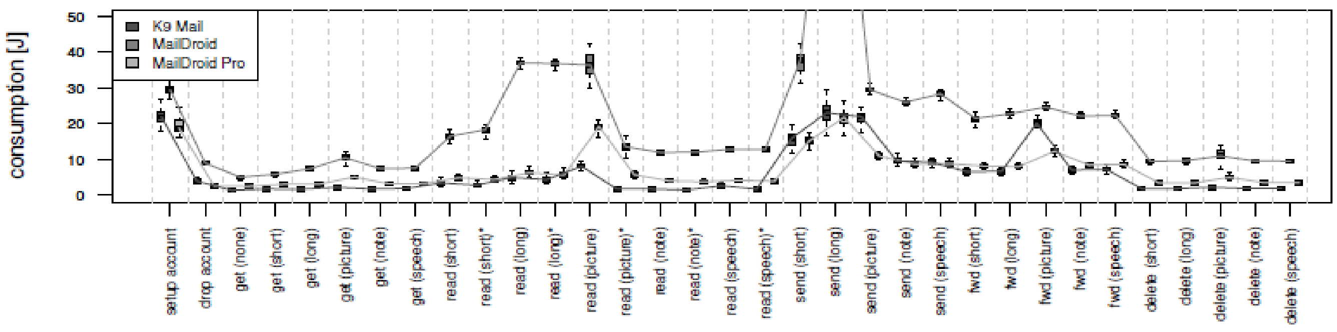


Figure 6 Median energy consumption for Android email clients (Wilke et al., 2013)

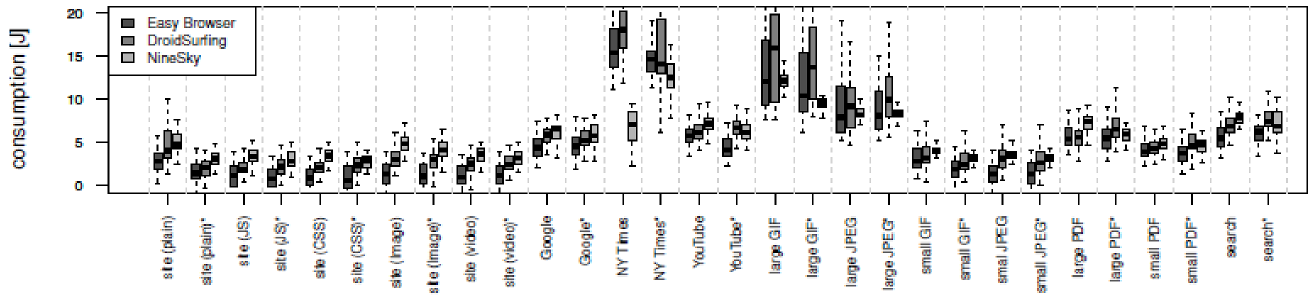
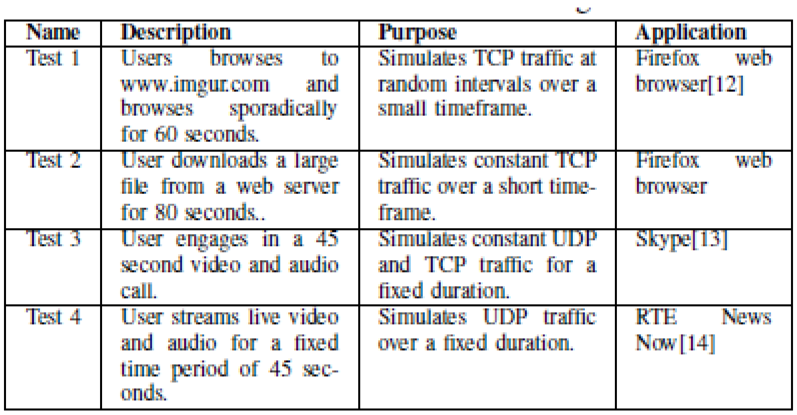


Figure 7 Median energy consumption for Android web browsers (Wilke et al., 2013)

The last study (Andreucetti et al., 2014) analyses the energy consumption of Wi-Fi network and 3G network using a Samsung Galaxy phone which runs Android 2.3.3 operating system. The tests that were performed are presented in Figure 8 and the results of the study in Figure 9.



Name	Description	Purpose	Application
Test 1	Users browses to www.imgur.com and browses sporadically for 60 seconds.	Simulates TCP traffic at random intervals over a small timeframe.	Firefox web browser[12]
Test 2	User downloads a large file from a web server for 80 seconds..	Simulates constant TCP traffic over a short time-frame.	Firefox web browser
Test 3	User engages in a 45 second video and audio call.	Simulates constant UDP and TCP traffic for a fixed duration.	Skype[13]
Test 4	User streams live video and audio for a fixed time period of 45 seconds.	Simulates UDP traffic over a fixed duration.	RTE News Now[14]

Figure 8 Details of the tests (Andreucetti et al., 2014)

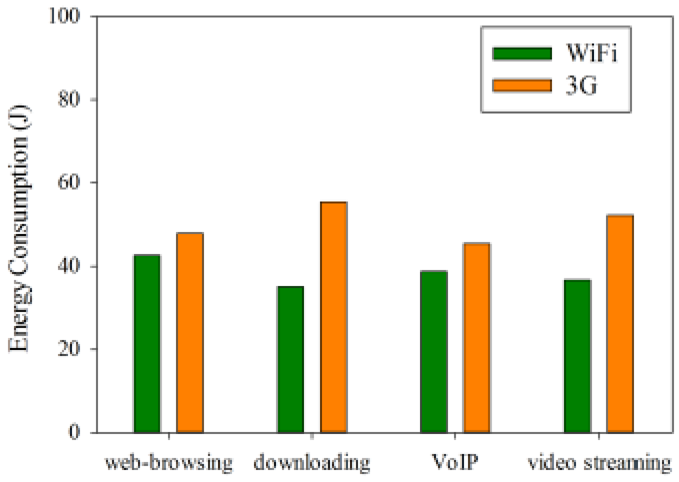


Figure 9 Results of the tests (Andreucetti et al., 2014)

This chapter presents some studies that are related to the current research. The findings presented in this paper can be compared to similar results, presented in this chapter. It can be noticed that most of the studies focus on the hardware components or on the network. The software component is not analyzed in detail in none of the papers. 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 current research comes to complete these studies. It tries to go one layer deeper and to analyze different factors that can influence the energy consumption of a mobile application. From (Corral et al., 2013) it is known the fact that the display component is one of the component that consumes most energy in an application. What is not known is why this phenomenon and how to improve the energy consumption. The purpose of this paper is to identify a part of the elements that consumes most of the energy and to come with solutions for each element. The following chapter will describe the methodology used and the hypotheses that are be tested in this thesis.

From the researcher’s knowledge, there is no existing published results on the impact of the various components in a mobile application (front-end; back-end; web service) and the battery/energy consumption. There are some recommendations related to the performance optimization (Blogs.msdn.com, 2015) and (Msdn.microsoft.com, 2015) but little on energy efficient/battery friendly application development. However, some of the existing work relates to: energy efficient mobile applications assistance (Kelenyi et al., 2014), energy-efficient mobile techniques (Siebra et al., 2012) and energy consumption estimation (Hao et al., 2013).

3. Methodology

In this chapter the method used for completing our research will be discussed. As it was already mentioned in the Chapter 1, 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 consume less energy. 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 master thesis focuses only on one specific operating system, Windows Phone 8.1, a product of Microsoft Company released in April 2014.

3.1. Application Development Tools

For the development of this master thesis three tools were used: Visual Studio 2013, Windows Phone Application Analysis and Microsoft Expression Design.

3.1.1. Visual Studio 2013

The development of the applications for Windows Phone 8.1 can be made using Microsoft Visual Studio 2013. This software is an IDE (integrated development environment) from Microsoft. It can be used for developing desktop applications, web sites, web services, Windows applications and mobile applications. As programming languages, Microsoft Visual Studio 2013 includes C, C++, VB .NET (Visual Basic), C# and F#. First version of Visual Studio was released in 1995 and the latest version, Visual Studio 2015, was announced in 2014. 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 UI part, each application can be opened in Microsoft Blend, which is a specialized tool in UI design. Figure 10 presents a basic Windows Phone application open in Visual Studio 2012:

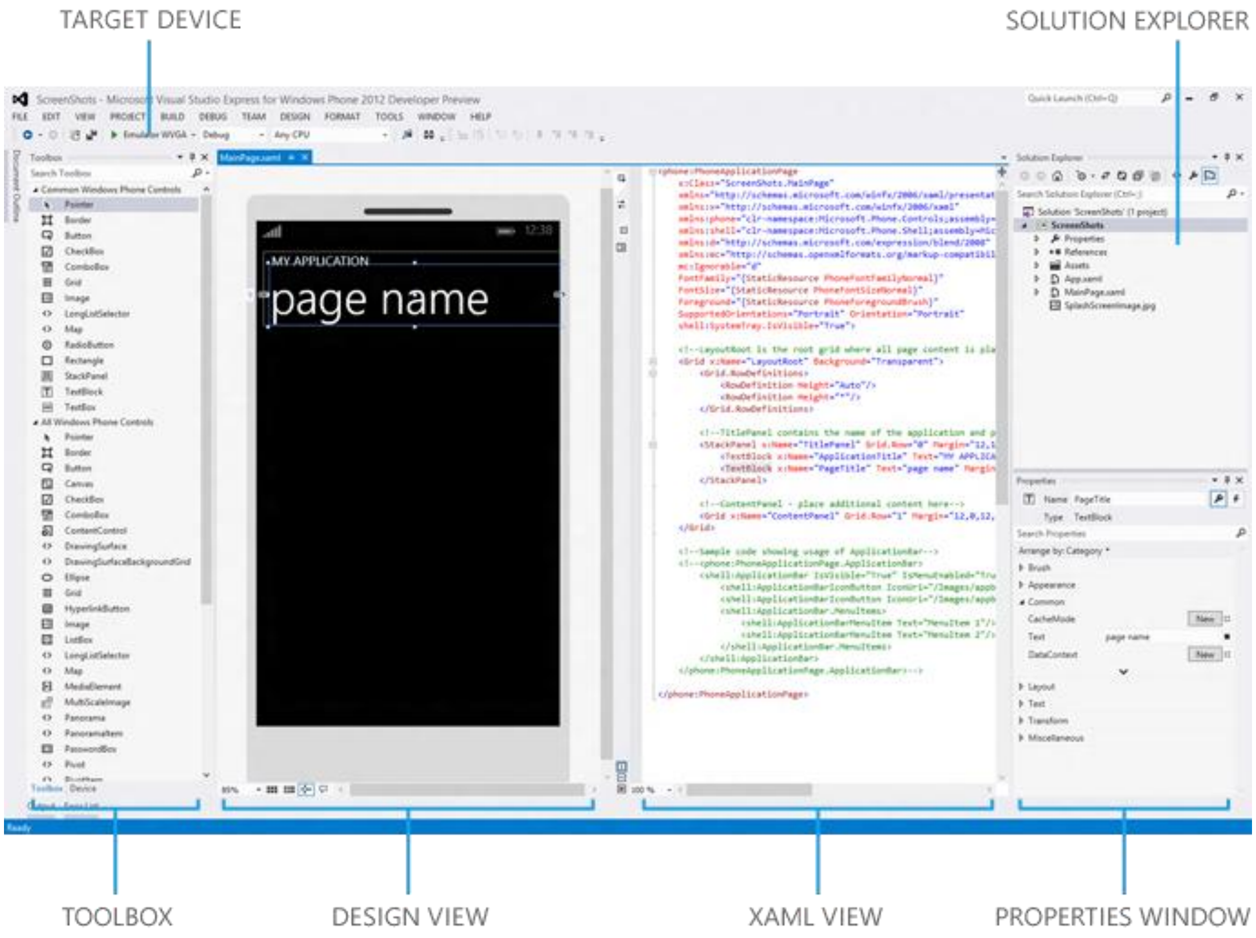


Figure 10 Visual Studio 2012 for Windows Phone (Msdn.microsoft.com, 2015)

The main components (Msdn.microsoft.com, 2015) 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.

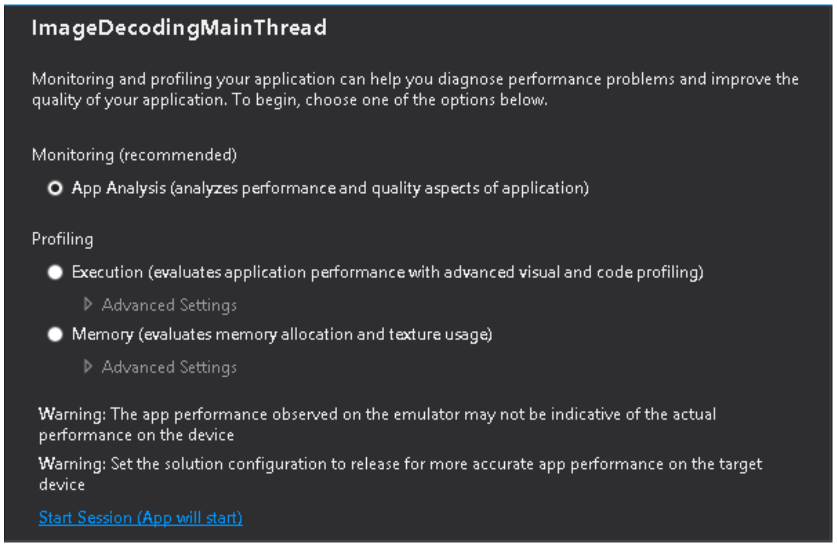
- Properties Windows – offers the possibility to see and to modify the properties of different controls or files.
- Solution Explored – shows all the projects and files that are included in the current solution, in a hierarchical way.
- Target Device – offers the possibility to choose the device on which the application will run. This device can be a virtual emulator or a real device. The virtual emulator it 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.

3.1.2. Windows Phone Application Analysis

Another tool that is really useful is Windows Phone Application Analysis tool. 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 interface of this tool looks like in Figure 11:



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

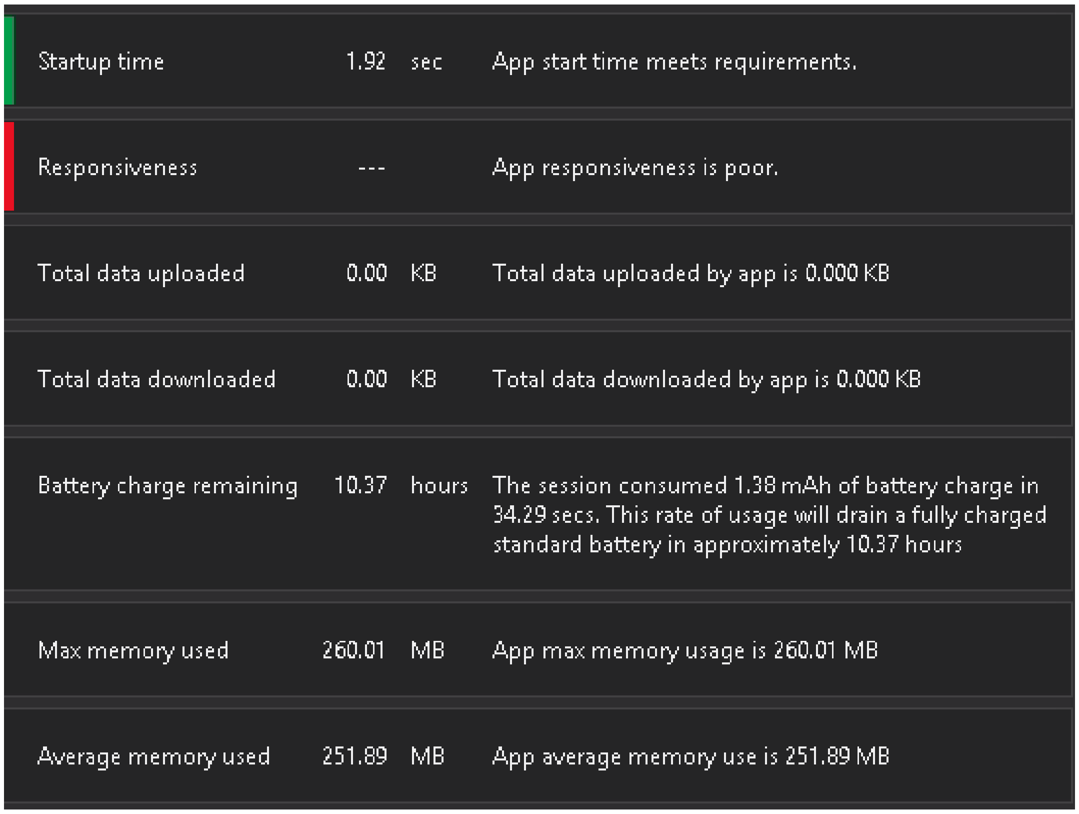
Warning: The app performance observed on the emulator may not be indicative of the actual performance on the device

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

Start Session (App will start)

Figure 11 Windows Phone Application Analysis tool interface

The output generated by this tool can be general or in detail. 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.



🟩	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

Figure 12Windows Phone Application Analysis tool – general output

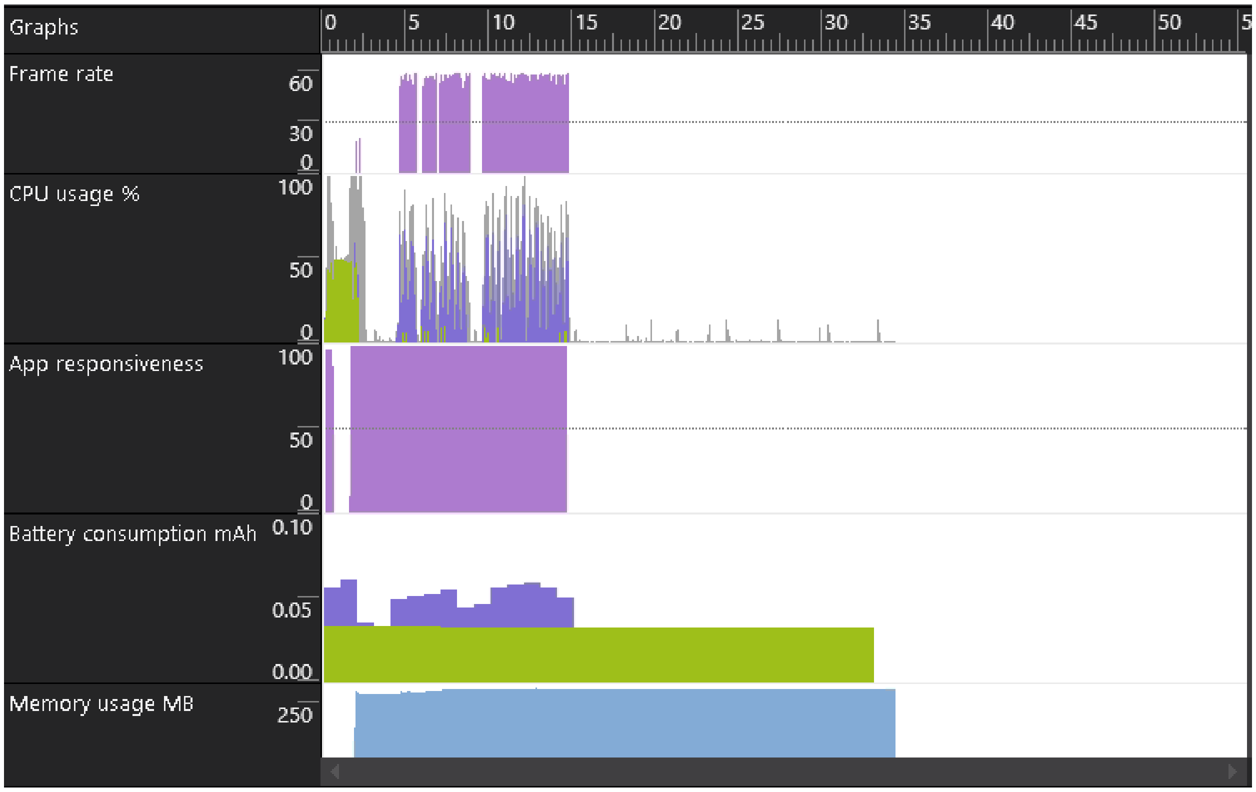


Figure 13 Windows Phone Application Analysis tool - detailed output

3.1.3. 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.

3.2. Experimental approach

The set of rules that are 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 called "Windows Phone 8: Performance and Optimization for Developers" (Channel 9, 2012) 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. There are two reasons for the importance of battery consumption: first reason is the time a user can spend in front of his/her device, while the second reason is the energy that is consumed by the device. Consequently, we propose to analyze some of these controls and concepts from energy point of view and see if they have a better consumption or not.

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 at the moment there can be made only assumptions whether the new controls are more efficient than the old ones, or whether one concept is more efficient than another one.

3.2.1. Experiment components

The main criterion that is applied in the selection of the elements, which is part in the experiments, is the diversity. 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 enumerated above in the following three groups:

Frontend components	VirtualizedStackPanel, StackPanel, ListBox, LongListSelector, ProgressBar, Opacity, Visibility,

	Storyboard, Image background creation, background property
Backend components	Assembly, recursive function, iterative function, page constructor, onNavigatedTo event, Thread, multithread, for, while, base64 string format, Image build action, synchronous loading, asynchronous loading, image decoding, image format
Web Services components	Clouds

Table 1 Elements

The next step in writing the hypothesis is to group all these elements based on their functionality. We will choose similar concepts and based on them we will formulate one hypothesis for each group. The output of the grouping operation is the following one:

- Frontend components:
  - **Group 1**: Background property
  - **Group 2**: StackPanel and VirtualizationStackPanel
  - **Group 3**: LongListSelector and ListBox
  - **Group 4**: ProgressBar: Indeterminate Progress bar and Determinate Progress Bar
  - **Group 5**: Visibility property and opacity property
  - **Group 6**: Storyboard
  - **Group 7**: PNG and JPG file format
  - **Group 8**: Image creation
  - **Group 9**: Storyboard and image
  - **Group 10**: XAML representation and image representation
- Backend components:
  - **Group 11 :** base 64 representation and image representation
  - **Group 12:** For and While instructions
  - **Group 13:** Assemblies
  - **Group 14:** OnNavigatedTo and page constructor
  - **Group 15:** Single threading and multi-threading
  - **Group 16:** Iterative and recursive

- **Group 17:** Image build action
- **Group 18:** Image decoding
- **Group 19:** Synchronous loading and Asynchronous loading

- Web Services Components
  - **Group 20:** Image stored in clouds and image stored locally
  - **Group 21:** Video stored in clouds and video stored locally
  - **Group 22:** Audio file stored in clouds and audio file stored locally
  - **Group 23:** Image format in clouds
  - **Group 24:** Image downloading and image accessing in clouds
  - **Group 25:** Processing locally and processing in clouds

3.2.2. Hypotheses

After having decided the use of experiments in our research, the next step is to identify the hypothesis. Due to the fact that the controls and concepts that we want to test, are used in different contexts, it is impossible to have only one hypothesis. For this reason, we have grouped our components based on their functionality and formulate a hypothesis for each group. Based on these groups we are able to obtain a number of 25 hypothesis which are tested and discussed in this thesis. The hypotheses are presented in Table 2:

	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 2 Hypotheses

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 we use Windows Phone Application Analysis software. The data that are collected are: battery charge remaining, the execution time and the battery consumption. After we obtain the battery consumption, we transform it into energy consumption. For this transformation we use 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, we assume 3.7 Volts as the voltage for Nokia Lumia 1320 which is used throughout the experiments.

3.2.3. Experiment template

Having all of these data for one experiment, we fill the following experiment template, which is used for all the experiments:

Experiment number x

Aim: This section contains the aim of the experiment.

Equipment: This section contains the required equipment.

Experiment procedure: This section contains the steps that are required in order to complete the experiment. One or several snapshots of the applications will be included in this part.

Results: This section contains the results of the experiment. The results section contains a table that contains the numerical results and two or several charts that will illustrate the battery consumption for each option that is tested.

Example:

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (Wh)**
Option 1	x	x	x	x
Option 2	x	x	x	x

Table 3 Results table

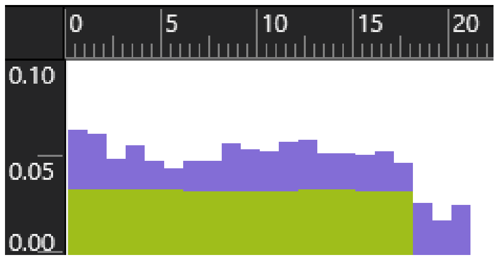


Figure 14 Chart 1

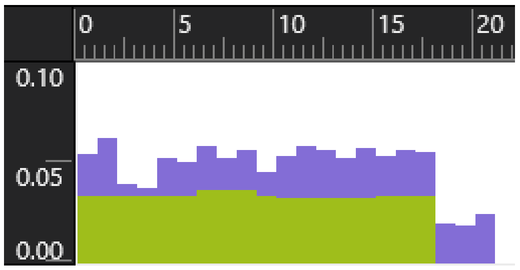


Figure 15 Chart 2

Thread	**Color**	**Description**
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table 4 Threads description

Axis	**Description**
X	Time (s)
Y	Battery consumption (mAh)

Table 5 Axis description

Conclusions: This section contains the conclusions of the experiment.

3.2.4. Experiment configurations

The experiments that are proposed for this thesis are device dependent. This means that the collected results are specific for a device. However, the rule abstracted are generalizable. The configurations that are used for the experiments can be found in the following table:

Property	**Value**

Battery voltage	3.8V
Nominal voltage	3.7V
Battery type	BV-4BW
Emulator type	720p
Emulator resolution	1280x720
Brightness	100%

Table 6 Device configuration

As it can be noticed in the above table, the only dependencies are related to the battery and screen resolution. This means that we should obtain some numbers for a specific emulator but the rules that will be obtained can be applied to any device. This phenomenon appears because we measure three threads: UI thread, application thread and network thread. The only difference in numbers is for the UI thread that is dependent on the resolution screen. The battery properties are important for the transformation of battery consumption in energy consumption. Since the battery is the same type for a specific device it does not influence the final result.

3.2.5. Experiment description

Once we have grouped the elements, formulated the hypothesis and defined all the elements that are dependent on the device, the next step is to write one or several applications for each experiment and to obtain the results. A detailed description of each experiment can be found in the Appendix of this thesis. The experiments are grouped into categories shown in Table 7.

Experiment	**Appendix**
Experiment 1 – Background Color	Appendix 1
Experiment 2 – Image format (JPG vs PNG)	Appendix 2
Experiment 3 - Visual object storing	Appendix 3
Experiment 4 – Decoding threads	Appendix 4
Experiment 5 – Animated vs static object	Appendix 5
Experiment 6 – Image decoding	Appendix 6
Experiment 7 – Image loading	Appendix 7
Experiment 8 – Control hiding	Appendix 8

Experiment 9 – ProgressBar consumption	Appendix 9
Experiment 10 – List control	Appendix 10
Experiment 11 – Build type property	Appendix 11
Experiment 12 – Image format	Appendix 12
Experiment 13 – Loop instructions	Appendix 13
Experiment 14 - Threads	Appendix 14
Experiment 15 - Method for data loading	Appendix 15
Experiment 16 – Function type	Appendix 16
Experiment 17 – StackPanel control	Appendix 17
Experiment 18 - Assemblies	Appendix 18
Experiment 19 - Animations	Appendix 19
Experiment 20 – Storing images	Appendix 20
Experiment 21 – Playing videos	Appendix 21
Experiment 22 – Playing audio files	Appendix 22
Experiment 23 – Image format (JPG vs PNG) in clouds	Appendix 23
Experiment 24 – Images – Multiple access	Appendix 24
Experiment 25 – Heavy processing operations	Appendix 25

Table 7 Name of the experiment and related annex

3.2.6. Elements used in experiments

The controls and concepts that were tested are the following ones:

3.2.6.1. Frontend

- **Background property (Msdn.microsoft.com, 2015):** This property is used for setting or getting a brush that is used for the background of a control. The brush object can have different types of output: SolidColorBrush which fills the area with a solid color, LinearGradientBrush which fills the area with a linear gradient, RadialGradientBrush which fills the area with a radial gradient, ImageBrush that fills the area with an image, DrawingBrush which fills the area with a drawing (vector or bitmap objects) and VisualBrush which fills the area with a visual object**.**

- **Image background creation (Blogs.msdn.com, 2015):** In the usual way, the decoding of an image is made by the UI thread. There is one property for the "Image" control that moves the decoding to a different thread. The property is called "CreateOptions" and its value has to be set to "BackgroundCreation". Below, there is an example of how to use this property:

```
<Image Height="100" Width="100" Margin="12,0,9,0" >
    <Image.Source>
        <BitmapImage UriSource="" CreateOptions="BackgroundCreation"/>
    </Image.Source>
</Image>
```

- **Storyboard (Msdn.microsoft.com, 2015):** A storyboard is a container that is used for animated objects. A storyboard is applied to the properties of an object, like color, width or height. For these properties we set the initial value, the final value and the period of time that is required for this transaction. This control offers the possibility to start, pause, stop and seek. The following piece of code shows how a storyboards is declared:

```
<StackPanel>
    <StackPanel.Resources>
        <!-- Animates the rectangle's opacity. -->
        <Storyboard x:Name="myStoryboard">
            <DoubleAnimation
    Storyboard.TargetName="MyAnimatedRectangle"
    Storyboard.TargetProperty="Opacity"
    From="1.0" To="0.0" Duration="0:0:1"
    AutoReverse="True"/>
        </Storyboard>
    </StackPanel.Resources>
    <Rectangle MouseLeftButtonUp="Rectangle_Tapped"
x:Name="MyAnimatedRectangle"
Width="300" Height="200" Fill="Blue" />
</StackPanel>
```

- **Visibility property (Msdn.microsoft.com, 2015):** This property can be applied to any control and has as effect the hiding of the control. There are two possible values: Visible, which means that the control will be visible and Collapsed, which means that the control is not visible. When the value is changed from Visible to Collapsed means the object is not kept in memory anymore, cannot trigger any event and any processing related to the control is impossible. When the value is changed from Collapsed to Visible it means that the control will be redrawn. The value of this property can be set in the XAML page: `<Button x:Name="button" Visibility="Collapsed"/>` , or in the backend code: `button.Visibility = Visibility.Collapsed;`

- **Opacity property (Msdn.microsoft.com, 2015):** As the visibility property, opacity is also used for making controls visible or invisible. It can take values from 0, which means invisible, to 1, which means visible. When the value of this property is set to 0, an image of the control is saved in the memory and it does not have to be redrawn when the property will be set to a value different than 0. Even with opacity set to 0 a control can participate into events and it is possible to process the content of the element. The value of this property can be set in the XAML page: `<Button x:Name="button" Opacity="1"/>`, or in the backend code: `button.Opacity = 0.0;`
- **ProgressBar (Msdn.microsoft.com, 2015):** is a control that is used for showing the progress of an operation. There are two types of ProgressBar: determinate, which is used when the total amount of time/work is known and it is displayed as a solid bar that moves from left to right, and indeterminate, which is used when the duration of an operation is unknown. In the second case there are three animated dots that move from left to right and have a repetitive behavior until the operation is done. The following examples show how to declare a determinate and an indeterminate ProgressBar:

```
<ProgressBar IsIndeterminate="False">

</ProgressBar>
```

```
<ProgressBar IsIndeterminate="True">

</ProgressBar>
```

- **ListBox (Msdn.microsoft.com, 2015):** This control is used for displaying a collection of items vertically. Usually it has a fixed dimension, which is set by the developer, and allows the scrolling through the elements. There are two properties that can be used for setting its content: Items and ItemsSource.

```
<ListBox ItemsSource="{Binding Items}" ScrollViewer.ManipulationMode ="Control" Height="652" Canvas.Top="80">
</ListBox>
```

- **LongListSelector (Msdn.microsoft.com, 2015):** This is a new control that was introduced for the first time in Windows Phone 8.0 SDK. It is similar with the ListBox control but comes with some new features, like grouping and searching. It also offer more templates which can be used in displaying data.

```
<phone:LongListSelector x:Name="AddrBook"/>
```

- **StackPanel control (Msdn.microsoft.com, 2015):** A StackPanel control is a collection of other UI controls. All of the controls will be the control children's. All the elements which are inside the StackPanel will be created when the page is called.

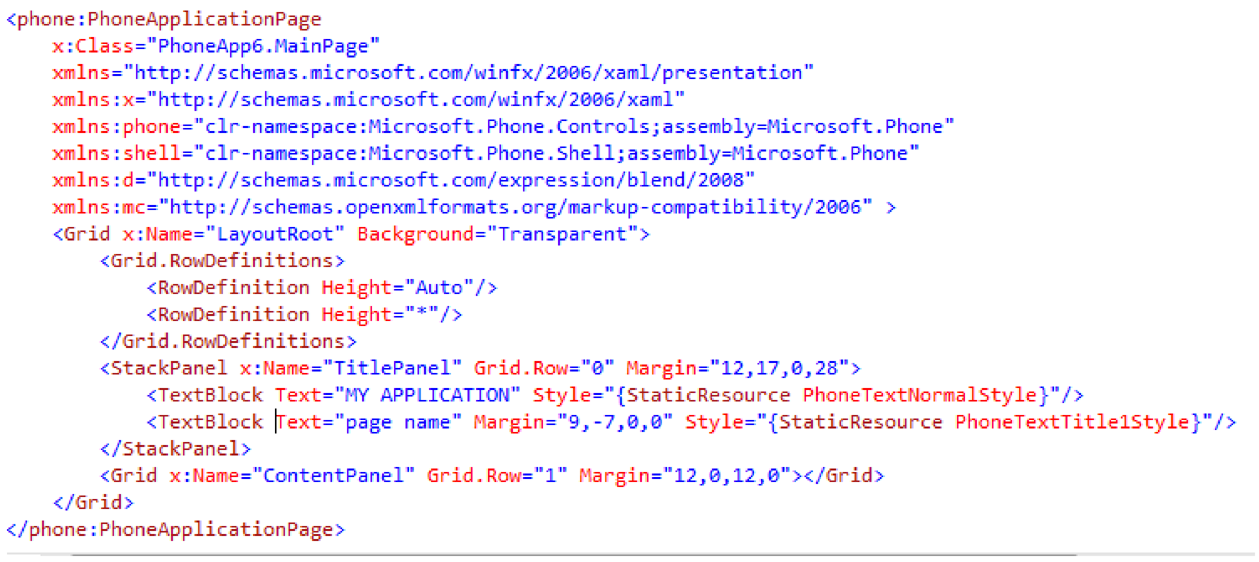
```
<StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
</StackPanel>
```

- **VirtualizedStackPanel control (Msdn.microsoft.com, 2015):** A VirtualizedStackPanel control has the same properties as the StackPanel. The difference between it and the StackPanel control is the fact that the elements inside a VirtualizedStackPanel control will not be created if they are not visible to user. All of these elements will be loaded only when the user scrolls through the application and become visible. Also a VirtualizedStackPanel can be placed only inside an ItemsControl element.

```
<VirtualizingStackPanel .../>
```

3.2.6.2. Backend:

- **File format:** This concept refers to the way in which information is encoded in a file. There are two roles a file format has: first role is to specify if the file is binary, or ASCII file, while the second role to is to specify how the information is organized. In our research we will work with three formats:
    - **PNG (W3.org, 2015):** A PNG file format is a lossless compression file format transmitted across the internet. It supports indexed-color, grayscale and true color images.
    - **JPG (Whatis.techtarget.com, 2015):** The JPG file format was specially created for storing photographic images and it is a lossy compressed file format. A JPG file includes a sequence of segments and each of this segment begins with a marker. The marker begins with a 0xFF byte followed by a byte that indicates the type of the marker.
    - **XAML (Msdn.microsoft.com, 2015):** The XAML extension was developed by Microsoft and it is a XML-based markup language. It is included in Windows Phone applications, Silverlight applications and Windows Presentation Foundation. The purpose of this format is to create user interfaces and includes elements as: text, images, shapes, animations or grids. The code that is used by the XAML file is stored in the same file but with an extra extension: .cs.



```
<phone:PhoneApplicationPage
    x:Class="PhoneApp6.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" >
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock Text="MY APPLICATION" Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock Text="page name" Margin="9,-7,0,0" Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"></Grid>
    </Grid>
</phone:PhoneApplicationPage>
```

- **Image decoder to size (Msdn.microsoft.com, 2015):** By default, an image is decoded in its natural resolution. Many times an application needs an image in a custom resolution. This can be realized specifying in the decoding instruction the width and height that are desired:
  image.Source = PictureDecoder.DecodeJpeg(jpgStream, 194, 256);
- **Synchronous loading:** Loading images using a synchronous method means the UI thread will take care of all operations that are required for decoding, resizing and displaying the picture. The following instruction is used for this approach:
  BitmapImage.SetSource(Stream); (the image is loaded from stream)
- **Asynchronous loading:** Loading the images using an asynchronous method means the UI thread will take care of the decoding while the other operations related to image processing are realized in a separate thread. The following instruction is used for this approach:
  BitmapImage.UirSource = urisource; (the image is loaded via URI)
- **Image build action (Developers.de, 2015):** This property of an image refers to the way in which an image will be stored when the application is deployed. There are two possible values:

- **Resource:** When this value is used, it means that the picture is stored in the assembly file. When this picture is used in the application as source, it will be referenced as:

```
<Image Name="imgResource" Source="../concierge bell1.jpg" Margin="0, 300,0,0" Width="200" Height="200"/>
```

- **Content:** This value is used if a developer wants to store the image along the application file (XAP).

```
<Image Name="imgContent" Source="/concierge bell.jpg" Margin="0,-200,0,0" Width="200" Height="200" />
```

- **Base64 string format:** Base64 is an encoding scheme that transforms binary data to base 64 representation. "…is design to represent arbitrary sequence of octets in a form that allows the use of both upper- and lowercase letters but that need not be human readable" (Tools.ietf.org, 2015). This encoding scheme can be also applied to images. Below, there are representation of an image in base64 format and in PNG representation.

Figure 16 The representation in PNG format

```
iVBORw0KGgoAAAANSUhEUgAAABAAAAAQCAYAAAAf8/9hAAAC+klEQVR42m2TW0sbURDHZ7PZTbyg5LImrTY2F2MTrRrMWvDSQikqFbxg1WDf+gUEn/
TJR+2DgtCHClJ89AtYEEoLKr0Y26g1UauxaVFIskkVzHWTTXrOEaVUDwx7zjLzm/nPmUPBDctus9G1druD02rNMpqGs7Mz/47X5/FsbUr/+1L/HqwWCzs8ODhqMplGuDKdvqi4GGgEyGYyEIvFgi
cnx7Ov5+dnvnk84jWA0+Hg+nt6lqqqrXyl0QRyuRxkMhkxChvyiQgCCOGw+83CQtfb5WXhCnCnvJx9PvBszWSp4hEABdEkMwHQMnK+hIWDQYhGIu6X09Ot2zs7IgG4+nrHbul0k81tD0GhVJCMkAdg
WJaALmH4K+UkONjdg0g0Oj4+MTFFlWk0dHdH+zHDMvoHzS2odAZUGjWo1Br4/StAguS0HGgkCe9xxv3dXSwx+GpuroKqrbY6W5t4dy6fgwrDXdBwHJye/oHaunrCOEjE4yR7SWkpZEQRYufnIITCIGfk8GFl
laf4hrqhtqamRRF1WpIkON8uB2VBAVSazfADZbrf4ACsZ8/rxapQcAgYhgEWydv2+lwUX1831P6obTEtZiCNMojIGIUCGpw8CbLabJDP52H900dIp9JECssyoEA+B0c/XVQNktD/tNONLySVTiG7gNSgzJtf
N6C+0XkFSCaSqMEU6oUCAVjYO/TzlFatpl+4Bo41KpU+mUoBtlQ6DSVqLZmPwJEfcrkcGIxG2Pf50D5PgKyCDa5+Wa8g19jb2T72uKV5EktIJJMEch6Lk0okFJxB/RGRRHyFuE85BMhK0vjSu/dTBMBpNOxwX8/
aPYuZT6AyMSSRSkIKacbVXPQmA5lsBrLZLL5O98rn9dagIIhXo2wyGLjujidLVqORxwHxBK7kohoMwv+woczuja3vXYeBgHDtMeFKWvjGUbu1aqSooFCPZyOZRHLQLMQTiWA4Ep11b27NhCIR8cbXeLlwY3Wc1l
GoVJqxB4L4Q5GoR4hGrz3nv6YYa/2B9ROlAAAAAElFTkSuQmCC
```

Figure 17 The representation in base64 string format

- **For instruction (Msdn.microsoft.com, 2015):** A "for" loop runs a block of instructions repetitively until it meets a certain condition that is set to "false". It is usually used for iterating collections.

```
for (int i = 0; i < 10; i++)
{
}
```

- **While instruction (Msdn.microsoft.com, 2015):** A while instruction runs a block of instructions repetitively until it meets a certain condition set to "false".

```
int n = 1;
while (n < 6)
{
    n++;
}
```

- **Thread:** A thread is a concurrent execution of a block of instructions. This means that the instructions are executed from the first instruction to the last one and the nth instruction will not be executed until the n-1th instruction is completed.

- **Multithreading:** In a multithreading application there are several threads defined and each of them will execute a specific block of instruction. This means that the instructions are executed in parallel, and one instruction does not have to wait until another one finishes its execution.

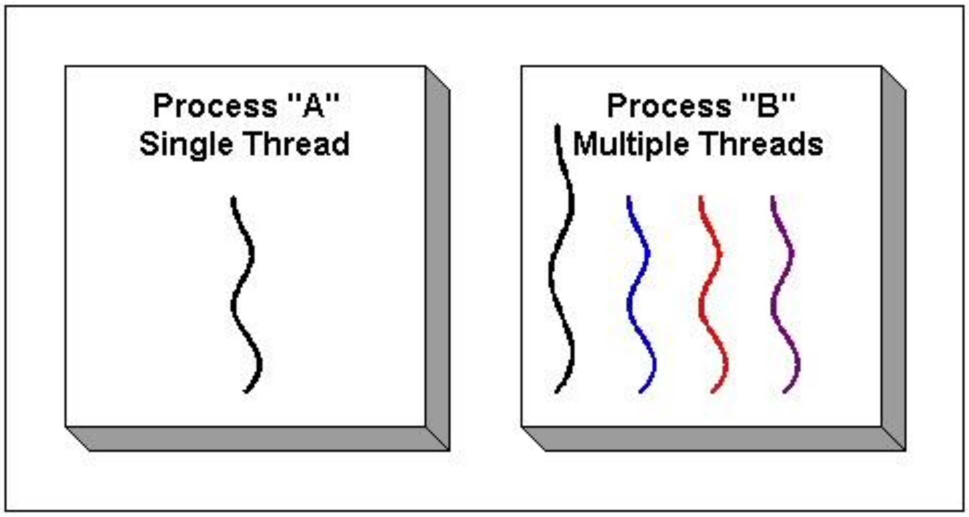


Figure 18 Thread concept

- **OnNavigatedTo method (Msdn.microsoft.com, 2015):** This method is the first method that is called after a page becomes active. If the page is called multiple times, this method is called every time. This method has to be overridden when a developer wants to place some code in it. Below, there is an example of how to override this method:

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
}
```

- **Page constructor:** The constructor initializes a new instance of the page and it is the first method, which is called a page that is requested. Usually all the components are initialized in this method. The classic declaration of a constructor looks like:

```
public MainPage()
{

}
```

- **Recursive function:** Any function that calls itself, it is called a recursive function. Its working principle is to split a problem into smaller programs and in the end the results to be combined. There are numerous examples of problems that can be solved using a recursive way, like Fibonacci number or factorial number. In each recursive function, an “If –else “condition has to be found.

- **Iterative function:** Any function that does not call itself is called an iterative function. In this function, there can be calls to other functions and any other instructions.
- **Assembly:** An assembly is a code library that is used for deployment. It is defined by Microsoft and it is available in the latest developed technologies. One assembly can contain one or more files that are executed by the .NET runtime environment.

3.2.6.3. Network

- **Clouds:** “Clouds computing is a general term for the delivery of hosted services over the internet” (SearchCloudComputing, 2015). This allows to the user to store files, to expose some services, to store important data or to publish applications that can be used by any other user. According to the same source there are three types of clouds: private, public and hybrid.

The following chapter will contain the obtained results and a discussion regarding these results for each experiment.

4. Results

This chapter will present the results that are obtained from the execution of the experiments. 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 were made. This chapter will present the results, in the form of a table, for each experiment and a discussion regarding the expected results compared to the actual results. Each experiment is presented in detail in Appendix. As a consequence, in this chapter it will present only the results of the experiments. Table 7, in Chapter 3, presents the appendix that corresponds to each experiment.

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (Wh)**
Black	23.36	0.56	17.52	0.002072
Purple	24.42	0.82	12.48	0.003034
Red	24.36	0.87	11.62	0.003219
Pink	22.16	1.27	7.26	0.004699
White	21.06	1.37	6.39	0.005069
Dark Blue	22.16	0.69	13.33	0.002553

Table 8 Experiment 1 - Background color - energy consumption

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (Wh)**
JPG format	10.53	0.29	15.07	0.001073
PNG format	10.58	0.29	15.02	0.001073

Table 9 Experiment 2 - Image format - 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

Table 10 Experiment 3 - Visual object storing - energy consumption

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (Wh)**
Background thread	33.49	1.27	10.96	0.004699
UI thread	34.19	1.38	10.37	0.005106

Table 11 Experiment 4 - Decoding threads - energy consumption

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (Wh)**
Animated	20.56	0.56	15.63	0.002072
Static	20.12	0.45	18.69	0.001665

Table 12 Experiment 5 - Animated and static object - energy consumption

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (Wh)**
Decoder to size	11.30	0.29	16.14	0.001073
Default decoder	11.57	0.31	15.79	0.001147

Table 13 Experiment 6 - Image decoding - energy consumption

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (Wh)**
Synchronous	21.28	0.64	13.78	0.002368
Asynchronous	22.17	0.65	14.20	0.002405

Table 14 Experiment 7 -Image loading - 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

Table 15 Experiment 8 - Control hiding - 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

Table 16 Experiment 9 - ProgressBar - energy consumption

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (Wh)**
ListBox	20.64	1.08	7.99	0.003996
LongListSelector	20.68	1.09	7.84	0.004033

Table 17 Experiment 10 - List control - energy consumption

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (Wh)**
Resource	22.03	0.65	14.09	0.002405
Content	22.35	0.66	14.13	0.002442

Table 18 Experiment 11 - Build type property - 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

Table 19 Experiment 12 - Image format - 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

Table 20 Experiment 13 - Loop instructions - 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

Table 21 Experiment 14 - Threads - energy consumption

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (Wh)**
Constructor	**32.14**	**1.19**	**11.25**	**0.004403**
OnNavigateTo	**31.78**	**1.18**	**11.18**	**0.004366**

Table 22 Experiment 15 - Method for data loading - 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 23 Experiment 16- Function type - energy consumption

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (Wh)**
StackPanel (without scrolling)	22.56	0.71	13.73	0.002627
VirtualizingStackPanel (without scrolling)	20.57	0.55	17.72	0.002035
StackPanel (with scrolling)	20.76	1.38	6.26	0.005106
VirtualizingStackPanel (with scrolling)	20.85	1.31	6.64	0.004847

Table 24 Experiment 17 - StackPanel control - energy consumption

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (Wh)**
One assembly (without navigation)	22.46	0.46	18.59	0.001702
Two assemblies (without nagivation)	20.55	0.46	18.67	0.001702
One assembly (with navigation)	26.31	0.61	18.49	0.002257
Two assemblies (with navigation)	26.98	0.61	18.45	0.002257

Table 25 Experiment 18 - Assemblies - 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

Table 26 Experiment 19 - Animations - 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

Table 27 Experiment 20 - Storing images - energy consumption

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (Wh)**
From internet	45.57	1.80	10.72	0.00666
Stored locally	46.77	1.89	10.29	0.006993

Table 28 Experiment 21 - Playing video - energy consumption

	Time (s)	Battery consumption (mAh)	Battery charge remaining (h)	Energy consumption (Wh)
From internet	93.32	2.38	16.35	0.008806
Stored locally	93.10	2.32	16.72	0.008584

Table 29 Experiment 22 - Playing audio files - 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

Table 30 Experiment 23 - Image format in cloud - energy consumption

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

Table 31 Experiment 24 - Images - multiple access - 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

Table 32 Experiment 25 - Heavy processing operation - energy consumption

This chapter presents the results of each experiment performed. It can be observed that in some cases, there are big differences regarding the energy consumption between the concepts analyzed, while in other cases the studied concepts consume the same amount of energy. For example, a case where the difference is big is Experiment 1, where the difference between black color and white color is 0.81 mAh or Experiment 14, where the difference between multithreading and single thread is 0.72 mAh. In experiments like Experiment 2, Experiment 12, Experiment 13, Experiment 18, Experiment 19, there is the same amount of energy consumed by the concepts under investigation. In the rest of the experiments it can be noted a difference in the total amount of consumed energy. From the total number of 25 experiments, the assumed hypothesis is true in 14 cases. The hypothesis is not relevant in 5 experiments and it is false in 4 cases. Two hypotheses are inconclusive. Table 27 presents a summary of the results obtained from these experiments (note the third column is the energy efficiency rule).

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
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

Table 33 Rules obtained after running the experiments

The next chapter will contain the conclusions of this dissertation and some aspects that could be considered for future work in this domain.

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. From the comparative analysis in the experiments, we abstract rules relating to software energy consumption, which are hardware independent. This study reveals the fact that there are some concepts, such as single threading, which consume more energy than similar concepts 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. The second reason for this study is the 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 percent will grow, because the ICT domain is in a continuous development, so it is very important to reduce the energy in all the aspects. In Chapter 1, it presents the trend of the mobile applications development. This trend is ascending and thousands of applications are released every day. Not all the developers are aware of the impact that their applications have on the environment. In this case, they will use a concept that is faster or a concept that is known by them. That is why it is very important to offer them a “green” alternative when they are making these decisions. If all the applications release from now on would follow some “green” rules, the total impact on the world’s energy consumption would be totally different. In the second case, there are developers that are aware of the environmental problems, but they do not have the necessary time to investigate the energy consumption of each control that they use. In this case it is important to have these rules, so they can use them in the development process.

There are studies conducted in this domain, but most of them are focused on Android phones or on iOS phones. Windows Phone is not very popular at the moment, but, according to the sources presented in Chapter 1, 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 see exactly the differences between them. Another direction of further study can be in finding 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. The third direction of this work can be the development of a mobile applications framework that use these rules. Even though in this thesis there were developed some “green” rules for writing mobile phone applications it could be interesting to investigate and develop small applications that can be integrated in the operating system. An example of an application would be a “fade to dark” functionality.

References

Andreucetti, R., Chen, S., Yuan, Z. and Muntean, G. (2014). Smartphone energy consumption of multimedia services in heterogeneous wireless networks. *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*.

Blogs.msdn.com, (2015). *Off-thread decoding of images on Mango, how it impacts your application ? - Silverlight for Windows Phone Performance team blog - Site Home - MSDN Blogs*. [online] Available at: http://blogs.msdn.com/b/slmperf/archive/2011/06/13/off-thread-decoding-of-images-on-mango-how-it-impacts-you-application.aspx [Accessed 15 Apr. 2015].

Carroll, A. and Heister, G. (2010). An analysis of power consumption in a smartphone.

Channel 9, (2012). *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].

Chen, X., Chen, Y., Ma, Z. and Fernandes, F. (2013). How is energy consumed in smartphone display applications?. *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications - HotMobile '13*.

Corral, L., Georgiev, A., Sillitti, A. and Succi, G. (2013). A method for characterizing energy consumption in Android smartphones. *2013 2nd International Workshop on Green and Sustainable Software (GREENS)*.

Developers.de, (2015). *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].

Emarketer.com, (2015). *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].

Hahnel, M., Dobel, B., Volp, M. and Hartig, H. (2012). Measuring energy consumption for short code paths using RAPL. *ACM SIGMETRICS Performance Evaluation Review*, 40(3), p.13.

Hao, S., Li, D., Halfond, W. and Govindan, R. (2013). Estimating mobile application energy consumption using program analysis. *2013 35th International Conference on Software Engineering (ICSE)*.

Jung, W., Kang, C., Yoon, C., Kim, D. and Cha, H. (2012). DevScope. *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis - CODES+ISSS '12*.

Metri, G., Agrawal, A., Peri, R. and Weisong Shi, (2012). What is eating up battery life on my SmartPhone: A case study. *2012 International Conference on Energy Aware Computing*.

Msdn.microsoft.com, (2015). *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].

Msdn.microsoft.com, (2015). *for (C# Reference)*. [online] Available at: https://msdn.microsoft.com/en-us/library/ch45axte.aspx [Accessed 15 Apr. 2015].

Msdn.microsoft.com, (2015). *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].

Msdn.microsoft.com, (2015). *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].

Msdn.microsoft.com, (2015). *Page.OnNavigatedTo Method (System.Windows.Controls)*. [online] Available at: https://msdn.microsoft.com/library/system.windows.controls.page.onnavigatedto(VS.95).as px [Accessed 15 Apr. 2015].

Msdn.microsoft.com, (2015). *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].

Msdn.microsoft.com, (2015). *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].

Msdn.microsoft.com, (2015). *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].

Msdn.microsoft.com, (2015). *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].

Msdn.microsoft.com, (2015). *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].

Msdn.microsoft.com, (2015). *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].

Msdn.microsoft.com, (2015). *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].

Msdn.microsoft.com, (2015). *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].

Msdn.microsoft.com, (2015). *What is XAML?*. [online] Available at: https://msdn.microsoft.com/en-us/library/cc295302.aspx [Accessed 15 Apr. 2015].

Msdn.microsoft.com, (2015). *while (C# Reference)*. [online] Available at: https://msdn.microsoft.com/en-us/library/2aeyhxcd.aspx [Accessed 15 Apr. 2015].

Namboodiri, V. and Ghose, T. (2012). To cloud or not to cloud: A mobile device perspective on energy consumption of applications. *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*.

news.microsoft.com, (2015). *Microsoft by the numbers*. [online] Available at: http://news.microsoft.com/bythenumbers/ms_numbers.pdf [Accessed 15 Apr. 2015].

Pathak, A., Hu, Y. and Zhang, M. (2012). Where is the energy spent inside my app?. *Proceedings of the 7th ACM european conference on Computer Systems - EuroSys '12*.

Protalinski, E. (2014). *Microsoft Confirms Windows Phone Store Passed 300,000 Apps*. [online] The Next Web. Available at: http://thenextweb.com/microsoft/2014/08/08/microsoft-confirms-windows-phone-store-300000-apps/ [Accessed 15 Apr. 2015].

SearchCloudComputing, (2015). *What is cloud computing? - Definition from WhatIs.com*. [online] Available at: http://searchcloudcomputing.techtarget.com/definition/cloud-computing [Accessed 15 Apr. 2015].

Spectrum.ieee.org, (2015). *A Smart Phone Uses as Much Energy as a Refrigerator? - IEEE Spectrum*. [online] Available at: http://spectrum.ieee.org/energywise/energy/environment/smart-phones-uses-as-much-energy-as-a-refrigerator [Accessed 15 Apr. 2015].

Tools.ietf.org, (2015). *RFC 4648 - The Base16, Base32, and Base64 Data Encodings*. [online] Available at: https://tools.ietf.org/html/rfc4648#section-4 [Accessed 15 Apr. 2015].

W3.org, (2015). *Portable Network Graphics (PNG) Specification (Second Edition)*. [online] Available at: http://www.w3.org/TR/PNG/#1Scope [Accessed 15 Apr. 2015].

Webb, M. (2008). SMART 2020: Enabling the low carbon economy in the information age.

Whatis.techtarget.com, (2015). *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].

Wilke, C., Piechnick, C., Richly, S., Poschel, G., Gotz, S. and Abmann, U. (2013). Comparing mobile applications' energy consumption. *Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC '13*.

www.idc.com, (2015). *IDC: Smartphone Vendor Market Share*. [online] Available at: http://www.idc.com/prodserv/smartphone-market-share.jsp [Accessed 15 Apr. 2015].

www.statista.com, (2014). *Topic: App Stores*. [online] Available at: http://www.statista.com/topics/1729/app-stores/ [Accessed 15 Apr. 2015].

Xia, F., Hsu, C., Liu, X., Liu, H., Ding, F. and Zhang, W. (2013). The power of smartphones. *Multimedia Systems*, 21(1), pp.87-101.

Blogs.msdn.com, (2015). *Image Tips for Windows Phone 7 - Stefan Wick's Weblog - Tips for WP7, Silverlight, WPF and Tablet PC - Site Home - MSDN Blogs*. [online] Available at: http://blogs.msdn.com/b/swick/archive/2011/04/07/image-tips-for-windows-phone-7.aspx?CommentPosted=true#commentmessage [Accessed 20 Jul. 2015].

Hao, S., Li, D., Halfond, W. and Govindan, R. (2013). Estimating mobile application energy consumption using program analysis. *2013 35th International Conference on Software Engineering (ICSE)*.

KelÃ©nyi, I., Nurminen, J., Siekkinen, M. and Lengyel, L. (2014). Supporting Energy-Efficient Mobile Application Development with Model-Driven Code Generation. *Advanced Computational Methods for Knowledge Engineering*, pp.143-156.

Msdn.microsoft.com, (2015). *App performance considerations for Windows Phone 8*. [online] Available at: https://msdn.microsoft.com/en-us/library/windows/apps/ff967560(v=vs.105).aspx [Accessed 20 Jul. 2015].

Siebra, C., Costa, P., Miranda, R., Silva, F. and Santos, A. (2012). The software perspective for energy-efficient mobile applications development. *Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia - MoMM '12*.

Appendix

Appendix 1. Experiment 1 – Background color

Aim: To investigate the impact background colors of an application have on energy consumption.

Equipment: For this experiment the following components are necessary:

- PC
  - Operating system: Windows 8.1
  - Development tool: Microsoft Visual Studio 2013
- Mobile phone
  - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. We develop an application using Visual Studio tool and C# programming language, which displays "Hello world" on the screen.

Step 2. During this experiment we change the "Background" property of the main grid. For this experiment, we use the following values: red (#FF0000), black (#00000), purple (#800080), pink (#FFC0CB), white (#FFFFFF) and dark blue (#00008B).

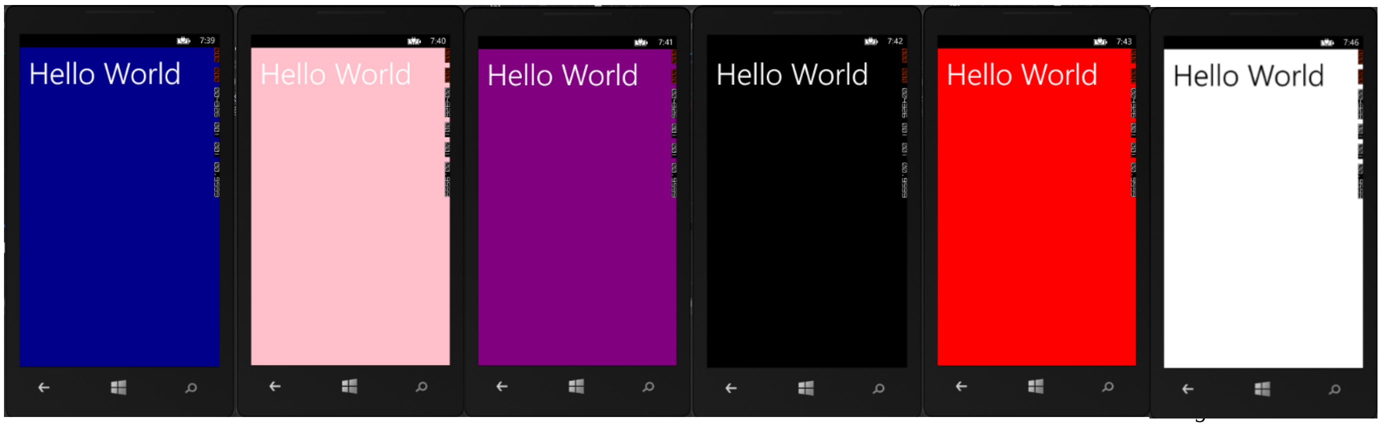


Figure A1.1. Application snapshots

Step 3. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 4. After we obtain the battery consumption, we transform it into energy consumption. For this transformation, we use the following formula:

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

The voltage value depends on the phone that is used. Consequently, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (wh)**
Black	23.36	0.56	17.52	0.002072
Purple	24.42	0.82	12.48	0.003034
Red	24.36	0.87	11.62	0.003219
Pink	22.16	1.27	7.26	0.004699
White	21.06	1.37	6.39	0.005069
Dark Blue	22.16	0.69	13.33	0.002553

Table A1.1 Background color – energy consumption

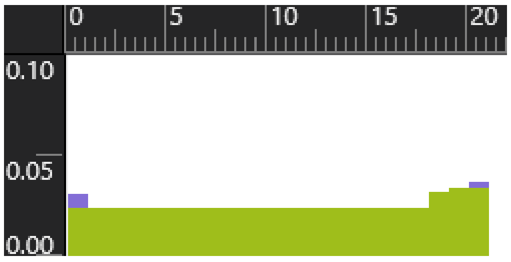


Figure A1.2 Chart for black color

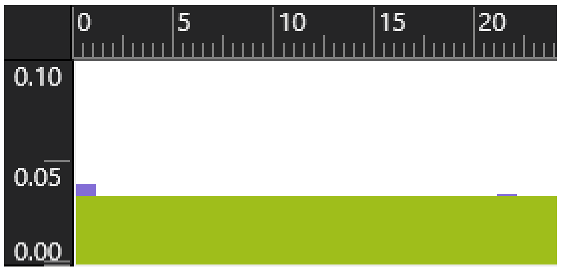


Figure A1.3. Chart for purple color

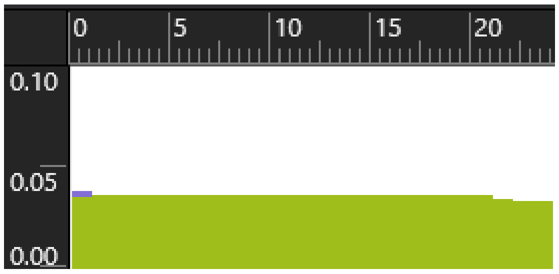


Figure A1.4. Chart for red color

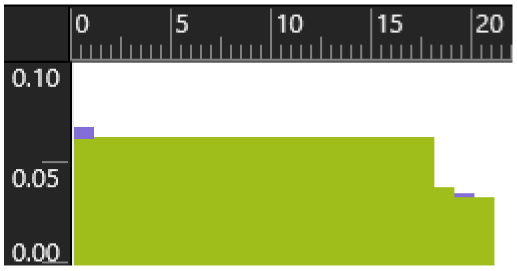


Figure A1.5. Char for pink color

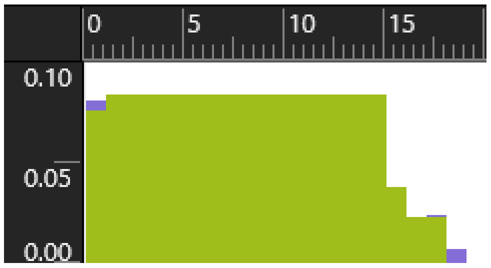


Figure A1.6. Chart for white color

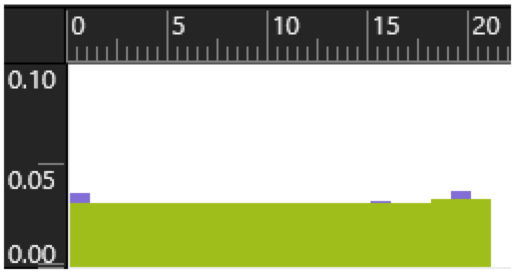


Figure A1.7. Chart for dark clue color

Axis	Description
X	Time (s)
Y	Battery consumption (mAh)

Table A1.3. Axis description

Thread	Color	Description
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A1.2. Threads description

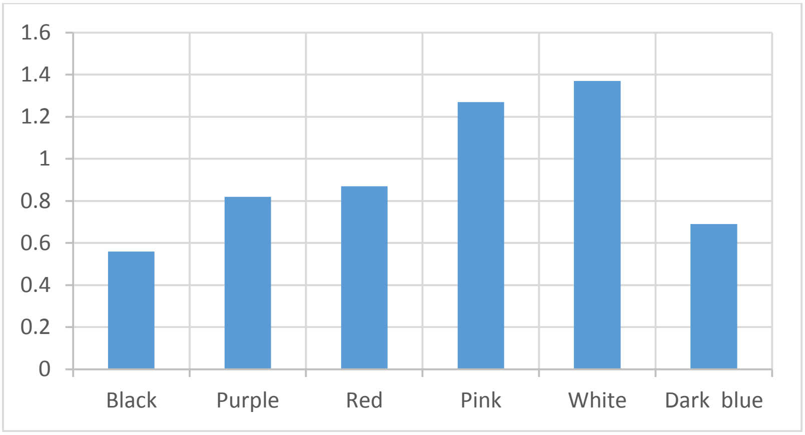


Chart A1.1. Background color – energy consumption

Conclusions: As we can see from the table above, the background color plays an important role in the energy consumption of a mobile application. Running the same experiment with different colors we obtained totally different results. The darker colors consume much less energy than the other colors. Having black or dark blue background, an operator can use his/her phone two times longer than using pink or white background. In the charts above we can see that the energy consumed by application thread (purple color) it is similar in all the cases and the energy consumed by this thread is generated when the application is launched and when it is terminated. The big difference that can be noticed here is related to UI thread (green color). In the case of black color we see a small constant energy consumption while in the case of white color, the energy consumed by UI thread is almost three times more. This rule is not generally valid because of the screen properties. Nokia supports AMOLED (active-matrix organic light-emitting diode) screens which do not have a solid backlight. That is the reason we can save battery, changing the background color, using a Nokia phone.

Appendix 2. Experiment 2 – Image format (JPG vs PNG)

Aim: To investigate the impact of displaying a PNG (Portable Network Graphics) file format and a Joint Photographic Experts Group (JPG) file format on energy consumption.

Equipment: For this experiment the following components are necessary:

- PC
  - Operating system: Windows 8.1
  - Development tools: Microsoft Visual Studio 2013
- Mobile phone
  - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. In the first step, we prepare our images. We work with the same two images, but one of them will be in a PNG format (dimension: 4288x2848, size: 2.78MB) and the other in a JPG format (dimension: 4288x2848, size: 2.78MB). The PNG format is a lossless compression file format while the JPG file, which is an extension of JPEG format is a lossy compressed file format.

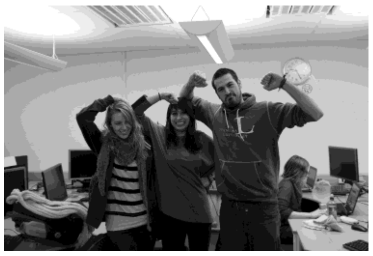


Figure A2.1. The PNG file format

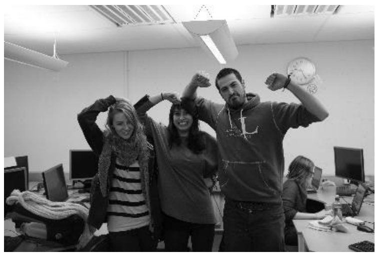


Figure A2.2. The JPG file format

Step 2. Using each of these two pictures, we develop an application using Visual Studio 2013 as development tool and C# as development language. In the application there is an

“Image” control whose source will be set once to the PNG file, and after that, to the JPG file. The application has the background set to transparent.

Figure A2.3. Application snapshot

Step 3. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 4. After we obtain the battery consumption, we transform it into energy consumption. For this transformation we use 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, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (wh)**
JPG format	10.53	0.29	15.07	0.001073
PNG format	10.58	0.29	15.02	0.001073

Table A2.1. Image format – energy consumption

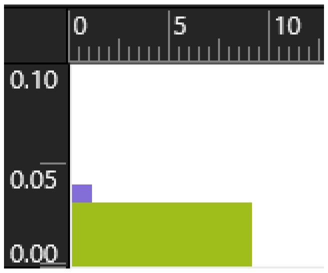


Figure A2.4. Chart for PNG format

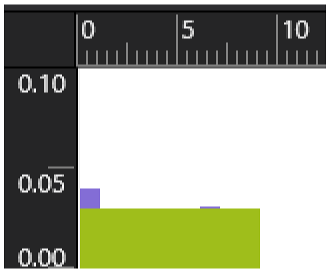


Figure A2.5.19 Chart for JPG format

Thread	**Color**	**Description**
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A2.2 Threads description

Axis	**Description**
X	Time (s)
Y	Battery consumption (mAh)

Table A2.3. Axis description

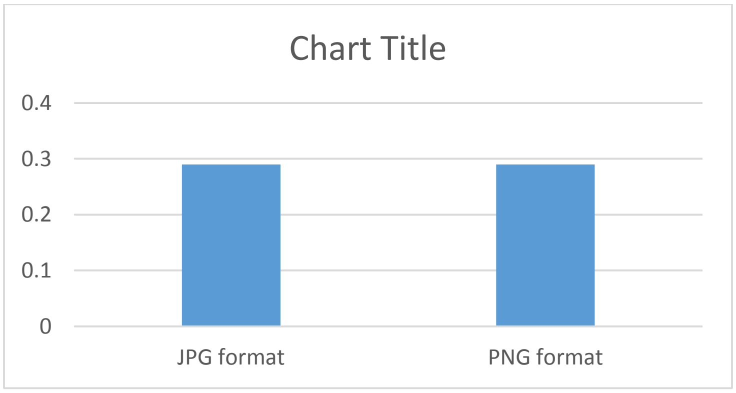


Chart A2.1. Image format – energy consumption

Conclusions: After this experiment, it can be observed that the format of the picture is not relevant if the pictures are stored locally. The amount of energy used for rendering these pictures is the same even though the JPG file is loaded a bit faster than the PNG file. This happens because the size of the images is the same and the quality is similar. The Figure A2.4 and Figure A2.5 shows us the energy distribution of the main threads: UI thread and application thread. We can see that the application thread (purple color) consume energy only when the application is launched and for the processing of the picture. The energy consumed by the UI thread (green color) it is constant over the execution of the application because it displays the same content.

Appendix 3. Experiment 3 – 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.

Objective: In this experiment we will test if it is more energy efficient to store a visual object as XAML or as a picture.

Equipment: For this experiment the following components are necessary:

- PC
    - Operating system: Windows 8.1
    - Development tools:
        - Microsoft Visual Studio 2013
        - Microsoft Expression Design 4
- Mobile phone
    - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. We create an image file and a XAML file. In this step we create two different files: a XAML file and a Portable Network Graphics (PNG) file (dimension: 640x480, size: 55.8 KB). They are created using Microsoft Expression Design 4. This is a tool used by developers to create graphic interfaces. The XAML files are Microsoft extensions of Extensible Markup Language and are used for creating User Interface pages. The PNG is a graphic file format which supports a lossless compression. The output obtained using this software is the following:

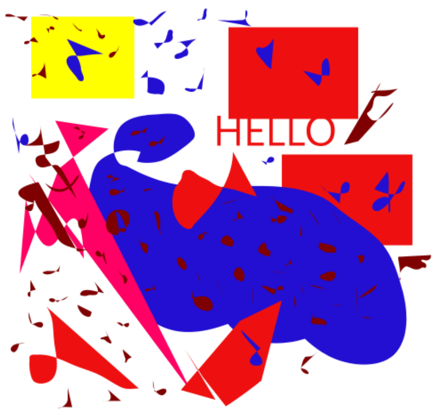
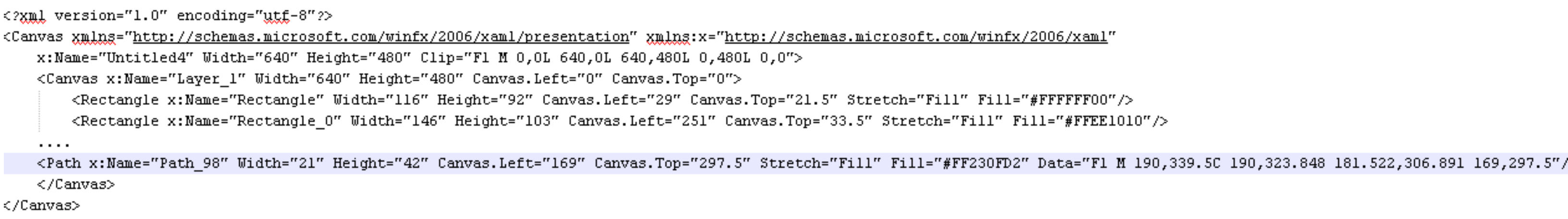


Figure A3.1. The PNG format



```
<?xml version="1.0" encoding="utf-8"?>
<Canvas xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Name="Untitled4" Width="640" Height="480" Clip="F1 M 0,0L 640,0L 640,480L 0,480L 0,0">
    <Canvas x:Name="Layer_1" Width="640" Height="480" Canvas.Left="0" Canvas.Top="0">
        <Rectangle x:Name="Rectangle" Width="116" Height="92" Canvas.Left="29" Canvas.Top="21.5" Stretch="Fill" Fill="#FFFFFF00"/>
        <Rectangle x:Name="Rectangle_0" Width="146" Height="103" Canvas.Left="251" Canvas.Top="33.5" Stretch="Fill" Fill="#FFEE1010"/>
    ....
    <Path x:Name="Path_98" Width="21" Height="42" Canvas.Left="169" Canvas.Top="297.5" Stretch="Fill" Fill="#FF230FD2" Data="F1 M 190,339.5C 190,323.848 181.522,306.891 169,297.5"/
    </Canvas>
</Canvas>
```

Figure A3.2. The XAML format

Step 2. For each element we create an application which displays it. During this experiment we will develop two applications using Visual Studio tool and C# language. Both applications will look like the image below:

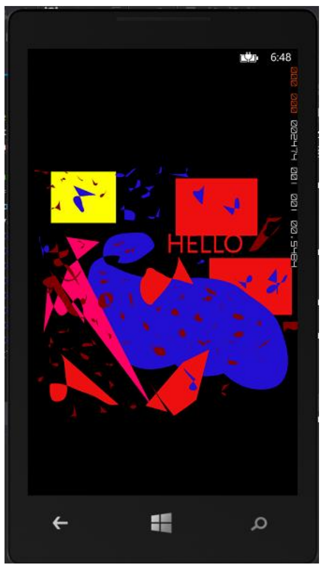


Figure A3.3. Application snapshot

Step 3. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 4. After we obtain the battery consumption we transform it into energy consumption. For this transformation we use 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, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	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

Table A3.1. Visual object storing – energy consumption

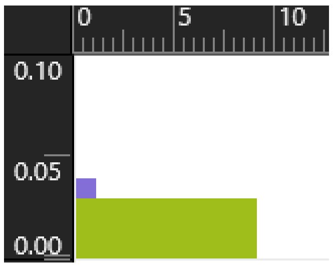


Figure A3.4. XAML chart

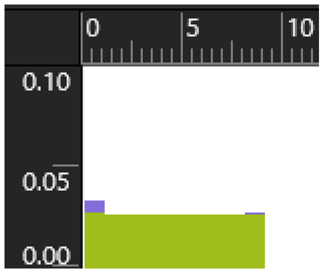


Figure A3.5. PNG chart

Thread	**Color**	**Description**
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A3.2. Threads description

Axis	Description
X	Time (s)
Y	Battery consumption (mAh)

Table A3.3. Axis description

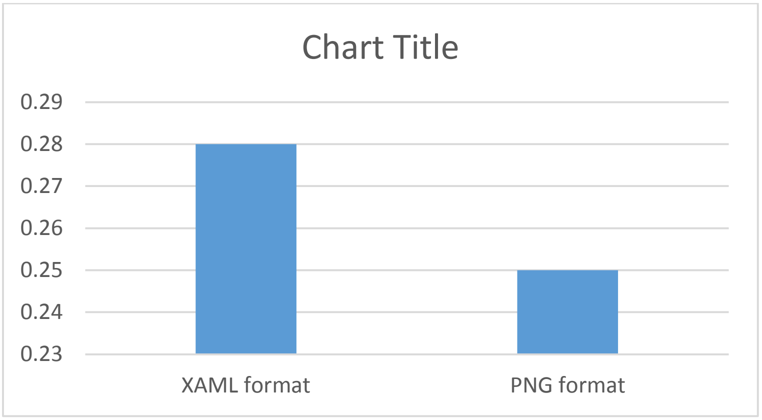


Chart A3.1. Visual object storing – energy consumption

Conclusions: Running these experiments we can notice 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 we are thinking to millions of applications that display images, this can be a considerable improvement. Also from the user's experience point of view, it is a big improvement considering 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. If we are looking at Figure A3.4 and Figure A3.5 we can notice that the energy consumed by the UI thread (green color) is the same in both cases. The only difference that can be notice is in the energy consumed by the application thread. In this case we can see that it requires more energy for creating the XAML object than to decode a picture.

Appendix 4. Experiment 4 – Decoding threads

Aim: To investigate the impact of displaying images using backgrounds threads and using the UI thread on energy consumption.

Equipment: For this experiment the following components are necessary:

- PC
    - Operating system: Windows 8.1

        - Development tools: Microsoft Visual Studio 2013
- Mobile phone
    - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. For this experiment we need a database of 15 images (downloaded from this website: http://wallpaperswide.com/music-desktop-wallpapers.html). The size and dimension of each picture are specific (Details in Table A4.1). Below there are some examples of the pictures that we use for this experiment:

Figure A4.1. Example of images

Image	Dimension	Size
Image 1	1920x1200	211 KB
Image 2	2560x1600	689KB
Image 3	2560x1600	1.34MB
Image 4	2880x1800	626KB
Image 5	1680x1050	267KB
Image 6	2560x1600	1.97MB
Image 7	2560x1600	687KB
Image 8	2880x1800	2.04MB
Image 9	2560x1600	1.01MB
Image 10	2560x1600	424KB
Image 11	1920x1200	681KB
Image 12	1920x1200	1.01MB
Image 13	2560x1600	806KB
Image 14	1680x1050	126KB
Image 15	4288x2848	2.78MB

Table A4.1 Dimensions and sizes of pictures

Step 2. The next step is the development of two applications, using Visual Studio 2013 development tool and C# programming language, which display these pictures in a list. The applications will display also a bigger picture that is in a different element. In one of the applications, all the pictures will have the attribute “CreateOption” set to “BackgroundCreation”. This attribute means that the image decoding is moved to the background threads. For the other application, the image decoding is made in the UI thread. The UI thread is the most important thread in an application because it has the responsibility to create the XAML objects, to draw all the visual objects and to execute the user's code.

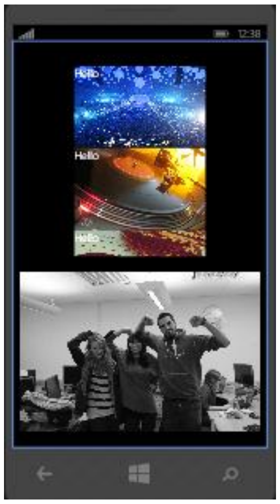


Figure A4.2. Application snapshot

Step 3. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 4. After we obtain the battery consumption we transform it into energy consumption. For this transformation we use 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, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	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

Table A4.2. Decoding threads – energy consumption

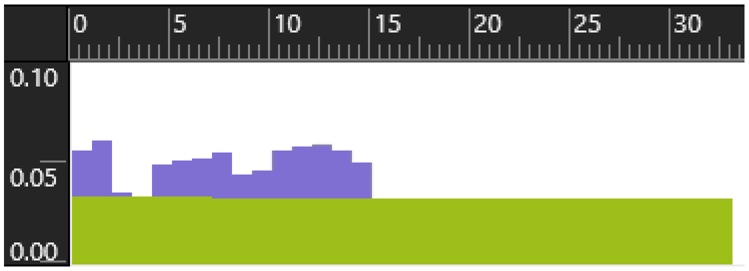


Figure A4.3. CreateOption Attribute set to BackgroundCreation

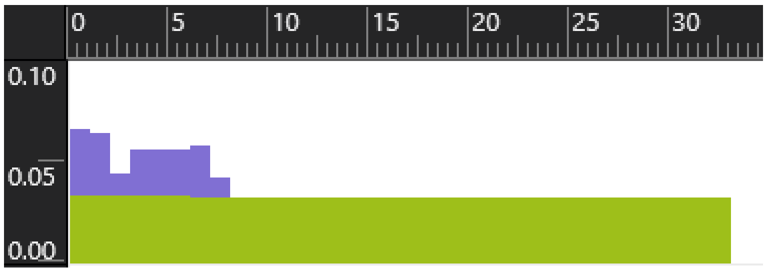


Figure A4.4. Without CreateOption attribute set

Thread	**Color**	**Description**
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A4.3. Threads description

Axis	**Description**
X	Time (s)
Y	Battery consumption (mAh)

Table A4.4. Axis description

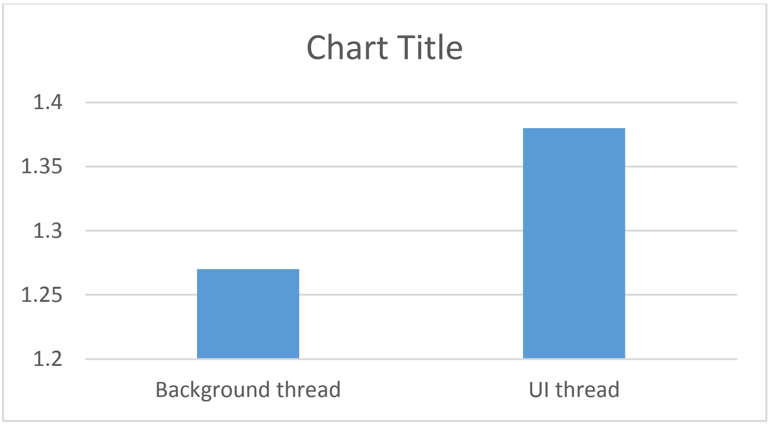


Chart A4.1. Decoding threads – energy consumption

Conclusions: This experiment shows that the energy consumed by these two applications is different. From Table A4.2, we can notice that decoding an image in a separate thread is more efficient than using only one thread. Regarding the energy distribution we can see that UI thread (green color) generates the same amount of energy in both cases while the application thread (purple color) generate less energy when we are 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 seconds while in the second case the application thread is working for 7 seconds. This happens because using more than one thread, the tasks are executed in a parallel way. When we have all the processing made by one thread it takes more time to decode all the pictures.

Appendix 5. Experiment 5 – Animated vs Static object

Aim: To investigate the energy impact of displaying an animated object compared to a static one.

Equipment: For this experiment the following components are necessary:

- PC
    - Operating system: Windows 8.1
    - Development tools: Microsoft Visual Studio 2013
- Mobile phone
    - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. Using Visual Studio 2013 development tool and C# programming language we write two applications, one displaying three animated objects, respectively one displaying the same objects in a static position. To begin with, we create three ellipses and one storyboard for each ellipse. A storyboard is a behavior which can be attached to an object to give it an animated effect. In the first application we start this behavior, while in the second case we do not. The objects move from one corner of the screen to the opposite one.

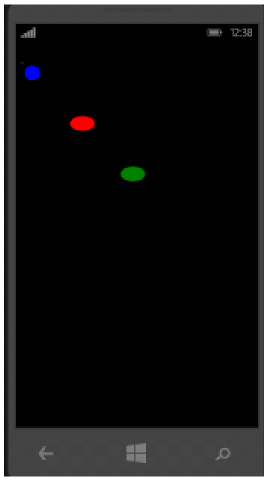


Figure A5.1. Application snapshot

Step 2. For each application, we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 3. After we obtain the battery consumption, we transform it into energy consumption. For this transformation we use 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, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	Time (s)	Battery consumption (mAh)	Battery charge remaining (h)	Energy consumption (wh)
Animated	20.56	0.56	15.63	0.002072
Static	20.12	0.45	18.69	0.001665

Table A5.1. Animated vs Static objects – energy consumption

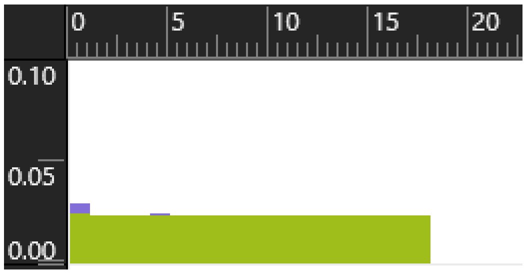


Figure A5.2. Static picture

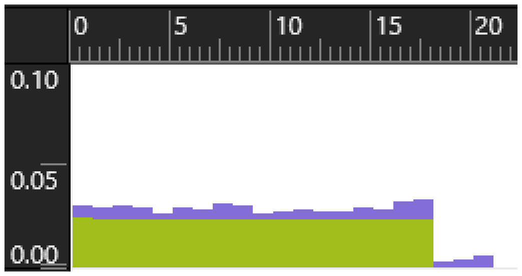


Figure A5.3. Animated picture

Thread	Color	Description
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A5.2. Threads description

Axis	Description
X	Time (s)
Y	Battery consumption (mAh)

Table A5.3. Axis description

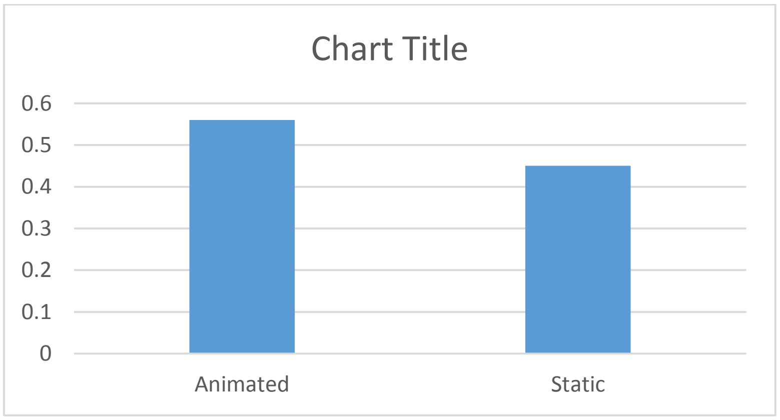


Chart A5.1. Animated vs Static objects – energy consumption

Conclusions: This experiment illustrates that static objects are more efficient from the point of view of energy consumption. This result is expected because, as we can see in the graph above, the animated images require also processing (purple color). If in the first graph the energy consumed for processing, by the static object, is almost 0, in the second case we see that it requires a constant energy for supporting the movement of the objects. The energy generated by the UI thread (green color) is the same in both cases because the same objects are displayed. For a very basic animation we see that the difference it is quite significant and we can improve the battery life with three hours by using static objects. There are cases when it is required to use animated objects, but on many occasions these objects are used just for the aspect of the application.

Appendix 6. Experiment 6 – Image decoding

Aim: To investigate the impact of displaying images using image decoder to size and using the default decoder on energy consumption.

Equipment: For this experiment the following components are necessary:

- PC
    - Operating system: Windows 8.1
    - Development tools: Microsoft Visual Studio 2013
- Mobile phone
    - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. For this experiment we need a database of 16 images (downloaded from this website: http://wallpaperswide.com/music-desktop-wallpapers.html).The size and dimension for each picture are specific (Details in Table A6.1). Below, there are some examples of the pictures that were used for this experiment:

Figure A6.1. Example of images

Image	**Dimension**	**Size**
Image 1	1920x1200	99.1 KB
Image 2	510x330	689KB
Image 3	2560x1600	1.34MB
Image 4	2880x1800	626KB
Image 5	1680x1050	267KB
Image 6	3888x2592	3.86MB
Image 7	2560x1600	687KB
Image 8	2880x1800	2.04MB
Image 9	2560x1600	1.01MB
Image 10	2560x1600	424KB
Image 11	1920x1200	681KB
Image 12	1920x1200	1.01MB
Image 13	2560x1600	806KB
Image 14	1680x1050	126KB
Image 15	510x330	96.7KB
Image 16	800x591	284KB

Table A6.1. Dimensions and sizes of the images

Step 2. The next step is the development of two applications, using Visual Studio 2013 development tool and C# programming language, which display these pictures in a table. For each application we create 16 images controls which have a predefined size of 100x100. In the first application, we use a default decoder, which means that the UI thread will resize them only in interface. In the second application, the images are resized before they are sent to the user interface. For this specific decoder we use the following instruction:

WriteableBitmap bitmap = PictureDecoder.DecodeJpeg(stream,width,height).

After this operation, we set the source of the image to this WriteableBitmap object. In the second case we set the source of the Image directly from the interface.

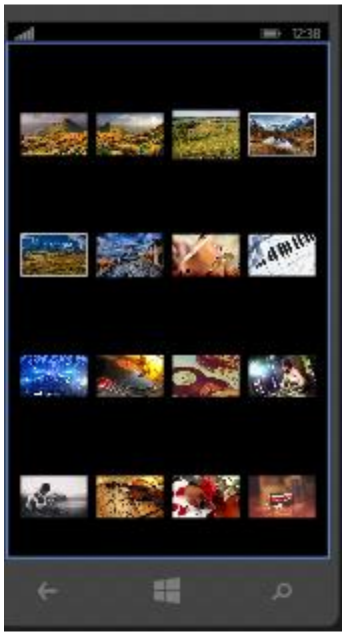


Figure A6.2. Application snapshot

Step 3. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 4. After we obtain the battery consumption, we transform it into energy consumption. For this transformation we use 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, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (wh)**
Decoder to size	11.30	0.29	16.14	0.001073
Default decoder	11.57	0.31	15.79	0.001147

Table A6.2. Image decoding – energy consumption

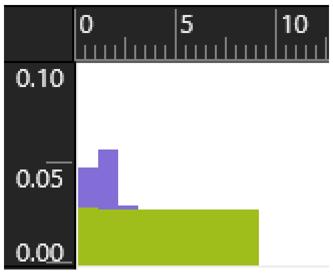


Figure A6.3 Default decoder

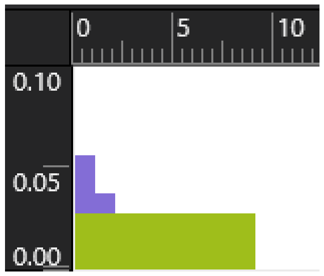


Figure A6.4. Custom decoder

Thread	**Color**	**Description**
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A6.3. Threads description

Axis	**Description**
X	Time (s)
Y	Battery consumption (mAh)

Table A6.4. Axis description

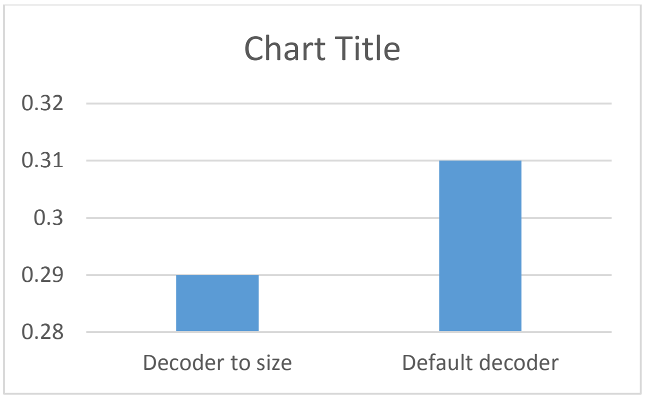


Chart A6.1. Image decoding – energy consumption

Conclusions: From the charts above we can notice that the energy consumed by the UI thread (green color) is the same for both applications, but there is a small difference in terms of energy distribution in the application thread (purple color). The custom decoder consumes more energy when the application is launched. The default decoder takes more energy because it has to process more the pictures that are given as input. Even though there is this small difference here, if we consider a situation when we have to resize hundreds of pictures, this difference will grow a lot.

Appendix 7. Experiment 7 – Image loading

Aim: To investigate the impact of displaying a set of images using synchronous and asynchronous methods on energy consumption.

Equipment: For this experiment the following components are necessary:

- PC
  - Operating system: Windows 8.1
  - Development tools: Microsoft Visual Studio 2013
- Mobile phone
  - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. For this experiment we need a database of 16 images (downloaded from this website: http://wallpaperswide.com/music-desktop-wallpapers.html).The size and dimension for each picture are specific (Details in Table A7.1). Below, there are some examples of the pictures that were used for this experiment:

Figure A7.1. Example of images

Image	Dimension	Size
Image 1	1920x1200	99.1 KB
Image 2	510x330	689KB
Image 3	2560x1600	1.34MB
Image 4	2880x1800	626KB
Image 5	1680x1050	267KB
Image 6	3888x2592	3.86MB
Image 7	2560x1600	687KB
Image 8	2880x1800	2.04MB
Image 9	2560x1600	1.01MB
Image 10	2560x1600	424KB
Image 11	1920x1200	681KB
Image 12	1920x1200	1.01MB
Image 13	2560x1600	806KB
Image 14	1680x1050	126KB
Image 15	510x330	96.7KB
Image 16	800x591	284KB

Table A7.1. Dimensions and sizes of the images

Step 2. We develop two applications, using Visual Studio 2013 development tool and C# programming language, which display these pictures in a table. For each application we create 16 “Image” controls that have a predefined size of 100x100. In the first application we load the images in a synchronous way, while in the second application we use an asynchronous manner. Loading the images using a synchronous method means the UI thread will take care of all operations that are required for decoding, resizing and displaying the picture. Loading the image in an asynchronous way does not mean that all the process will be done in separate threads, because the image decoding will be still made by UI thread. The instruction used for the asynchronous loading is:

BitmapImage.UirSource = urisource

while for the synchronous loading we use:

BitmapImage.SetSource(Stream).

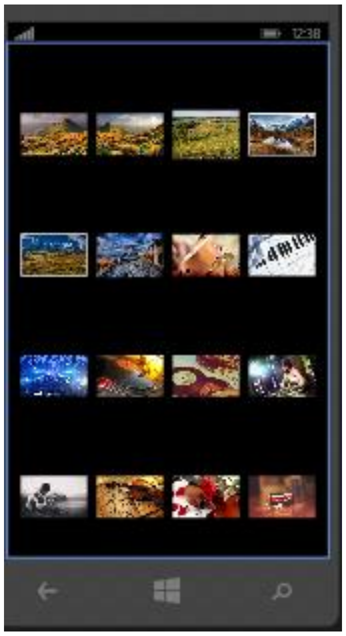


Figure A7.2. Application snapshot

Step 3. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 4. After we obtain the battery consumption, we transform it into energy consumption. For this transformation we use 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, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (wh)**
Synchronous	21.28	0.64	13.78	0.002368
Asynchronous	22.17	0.65	14.20	0.002405

Table A7.2. Image loading – energy consumption

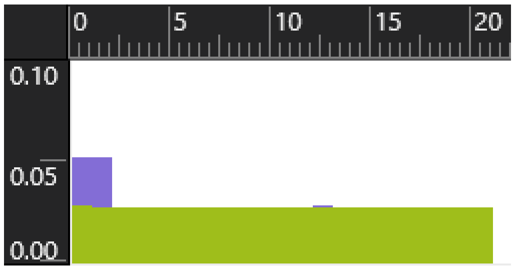


Figure A7.3. Sync loading

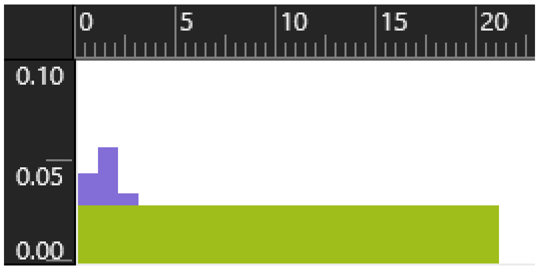


Figure A7.4. Async loading

Thread	**Color**	**Description**
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A7.3. Threads description

Axis	**Description**
X	Time (s)
Y	Battery consumption (mAh)

Table A7.4. Axis description

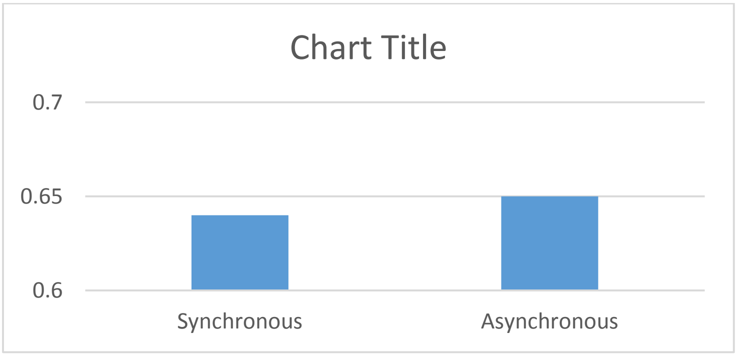


Chart A7.1. Image loading – energy consumption

Conclusions: The numbers we have obtained are quite close, so from energy point of view these two methods are similar. Nevertheless we can notice that the asynchronous method is more efficient and the battery will last a bit longer than using a synchronous method. We have this behavior because, as we already mentioned, not all the processing is made in a separate thread. Another thing that can be noticed, it is that using synchronous method will make our application to load slower. All the processing is made at the beginning of the application and that is not good for the UI thread, because it will become busy and will block the application. The asynchronous method will make our application faster and to load the pictures easier. From the Figure A7.3 and Figure A7.4 we can notice that the energy generated by the UI thread (green color) is the same in the both cases, while the energy generated by the application thread (purple) is different. For the synchronous loading it takes less time to load all the pictures but the processor works more, while for the asynchronous loading the processor is not so busy and the loading is made steadily.

Appendix 8. Experiment 8 – Control hiding

Aim: To investigate the impact of “visibility” property and “opacity” property on energy consumption.

Equipment: For this experiment the following components are necessary:

- PC
    - Operating system: Windows 8.1
    - Development tools: Microsoft Visual Studio 2013
- Mobile phone
    - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. We develop two application using Visual Studio 2013 development tool and C# programming language, which will display 150 stationary rectangles and two rectangles that will move across the screen. Every four seconds we set the “visibility” respectively “opacity” property of the rectangles to a different value. Both of these properties are used for making an UI element visible or invisible. If we are setting an object’s visibility to Collapsed means that XAML deletes the object form the memory. When the property is set to Visible the object is redrawn. When we are setting the opacity property to 0 it means that the object is not visible, but the representation of the object is still in the memory and the object is not redrawn when Opacity is set to a non-zero value.

Figure A8.1. Application snapshot

Step 2. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 3. After we obtain the battery consumption, we transform it into energy consumption. For this transformation, we use the following formula:

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

The voltage value depends on the phone that is used. Consequently, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	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

Table A8.1. Control hiding – energy consumption

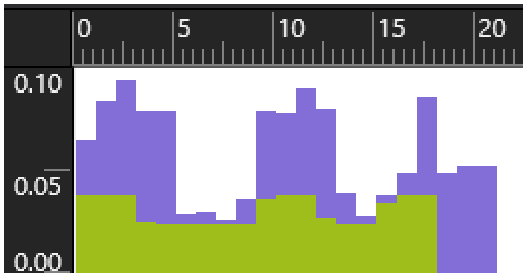


Figure A8.2. Visibility

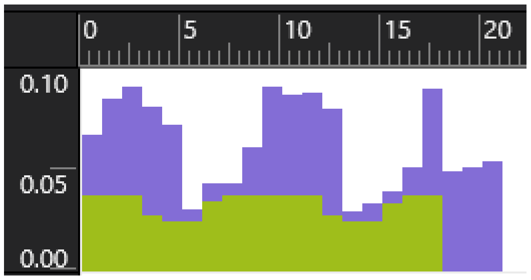


Figure A8.3. Opacity

Thread	Color	Description
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A8.2. Threads description

Axis	Description
X	Time (s)
Y	Battery consumption (mAh)

Table A8.3. Axis description

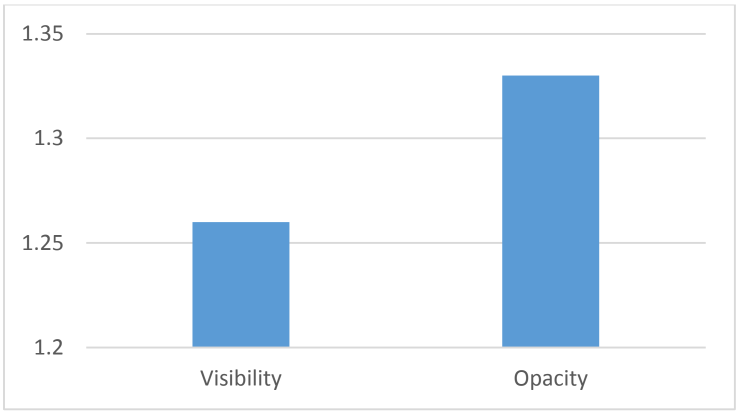


Chart A8.1. Control hiding – energy consumption

Conclusions: Both the applications are doing the same thing, but we observe that the energy consumption is different. We can see that 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 where we are using Opacity is faster, it costs more in terms of energy consumption. In the first graph we can see that the energy consumption of the UI thread is lower because the objects are deleted. In the second case, even if we cannot see the objects on the screen, they are stored in memory so more energy will be consumed. From the Figure A8.2 and Figure A8.3 we can observe some interesting facts: The energy consumed by the application thread (purple color) it 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 (green color). We see in Figure A8.2 that the UI thread consumes less energy while the objects are hidden. If we are setting the Opacity property the energy consumed by the UI thread does not drop live in the previous case.

Appendix 9. Experiment 9 – ProgressBar consumption

Aim: To investigate the energy efficiency of a determinate progress bar and an indeterminate progress bar.

Equipment: For this experiment the following components are necessary:

- PC
    - Operating system: Windows 8.1
    - Development tools: Microsoft Visual Studio 2013

- Mobile phone
    - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. For this experiment we develop two applications, using Visual Studio 2013 development tool and C# programming language, which display a progress bar, determinate, respectively indeterminate. A determinate progress bar means that we know how much time it will take for an operation to be completed, while an indeterminate progress bar means that we do not know about the time that it is required by an operation. We use the ProgressBar XAML control and we modify the property "IsIndeterminate" to true, respectively to false.

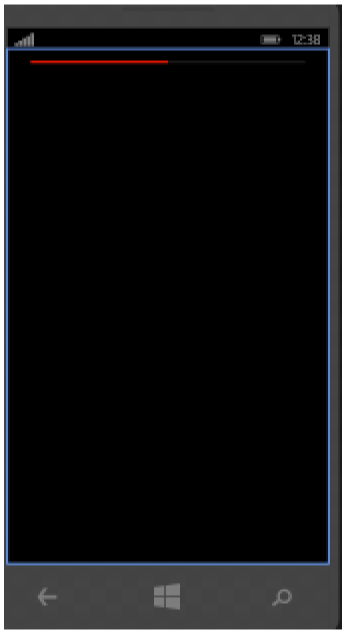


Figure A9.1. Application snapshot - determinate

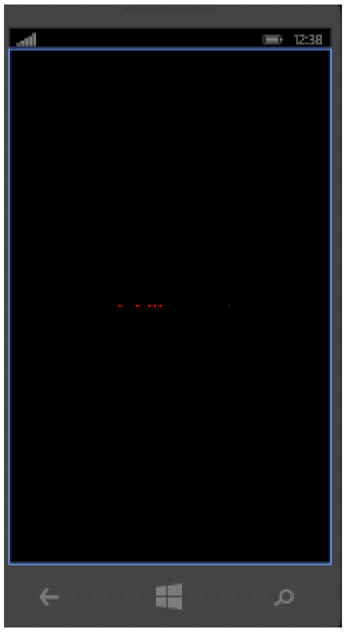


Figure A9.2. Application snapshot - indeterminate

Step 2. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 4. After we obtain the battery consumption, we transform it into energy consumption. For this transformation, we use the following formula:

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

The voltage value depends on the phone that is used. Consequently, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	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

Table A9.1. ProgressBar – energy consumption

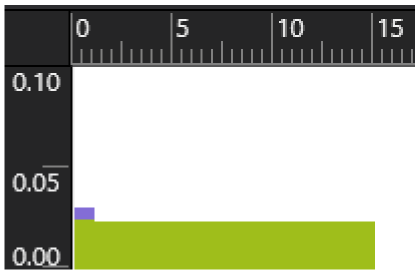


Figure A9.3. Determinate

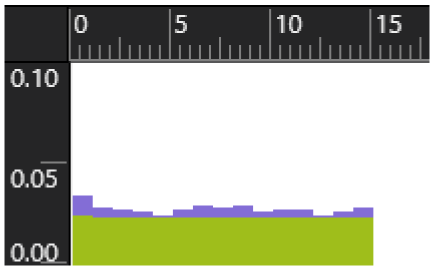


Figure A9.4. Indeterminate

Thread	**Color**	**Description**
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A9.2. Threads description

Axis	**Description**
X	Time (s)
Y	Battery consumption (mAh)

Table A9.4. Axis description

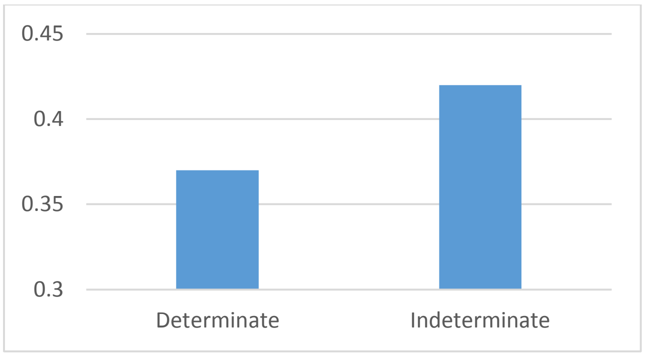


Chart A9.1. ProgressBar – energy consumption

Conclusions: As we can see 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 Figure A9.3 and Figure A9.4. The application thread (purple color) consumes more energy for an indeterminate progress bar because it supports the animation during the execution. In Figure A9.3 we can see 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. We use these controls in different cases, but if we can choose one of them in our application, that one should be determinate.

Appendix 10. Experiment 10 – List control

Aim: To investigate the energy efficiency of a “ListBox” control compared to a “LongListSelector” control.

Equipment: For this experiment the following components are necessary:

- PC
    - Operating system: Windows 8.1
    - Development tools: Microsoft Visual Studio 2013
- Mobile phone
    - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. For this experiment we develop two applications, using Visual Studio 2013 development tool and C# programming language, which display a list with 1000 elements. Each element contains an image (dimension: 256x256, size: 32,5 KB) and a text. For displaying these items we use two controls that are offered by Windows Phone: ListBox and LongListSelector. The difference between them is that LongListSelector supports extra operations like grouping or jumping directly to one group. During the experiment we keep scrolling through the application.

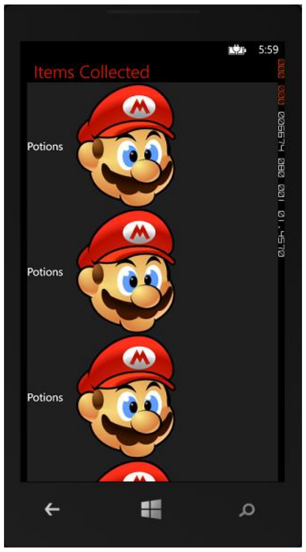


Figure A10.1. Application snapshot

Step 2. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 3. After we obtain the battery consumption, we transform it into energy consumption. For this transformation, we use the following formula:

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

The voltage value depends on the phone that is used. Consequently, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (wh)**
ListBox	20.64	1.08	7.99	0.003996
LongListSelector	20.68	1.09	7.84	0.004033

Table A10.1. List control – energy consumption

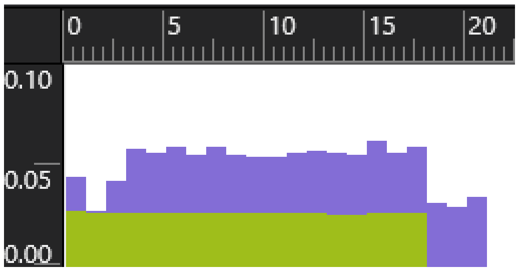


Figure A10.2. ListBox

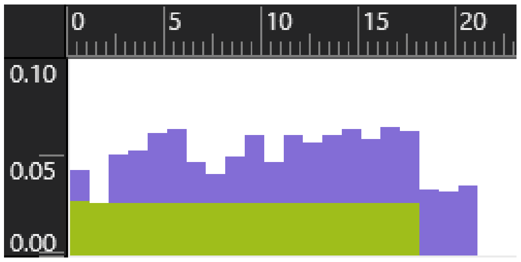


Figure A10.3. LongListSelector

Thread	**Color**	**Description**
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A10.2. Threads description

Axis	**Description**
X	Time (s)
Y	Battery consumption (mAh)

Table A10.3. Axis description

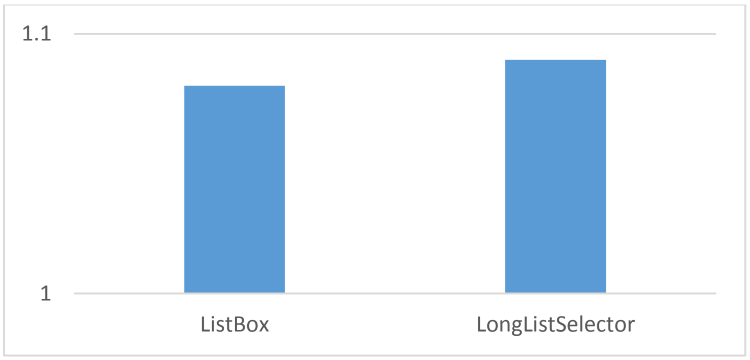


Chart A10.1. List control – energy consumption

Conclusions: LongListSelector is a new control in Windows Phone 8.1 and has some extra features like grouping and jumping from one group to another. Microsoft released this control because they considered that the performance is better and it is a faster control than ListBox. As it can be seen from the charts above, the difference is very small between these two controls from energy point of view. It can be observed that the ListBox is more efficient, even though the difference is very small. In Figure A10.2 and Figure A10.3 we can notice that the energy consumed by the UI thread (green color) is the same in both cases. The energy consumed by the application thread (purple color) it is quite constant for the ListBox. The LongListSelector has a fluctuating energy consumption.

Appendix 11. Experiment 11- Build type property

Aim: To investigate the impact of displaying images that have set their “Build type” to “Resource” and images that have set their “Build type” to ”Content” on energy consumption.

Equipment: For this experiment the following components are necessary:

- PC
    - Operating system: Windows 8.1
    - Development tools: Microsoft Visual Studio 2013
- Mobile phone
    - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. For this experiment we need a database of 16 images (downloaded from this website: http://wallpaperswide.com/music-desktop-wallpapers.html).The size and dimension for each picture are specific (Details in Table A11.1). Below, there are some examples of the pictures that were used for this experiment:

Figure A11.1. Example of images

Image	**Dimension**	**Size**
Image 1	1920x1200	99.1 KB
Image 2	510x330	689KB
Image 3	2560x1600	1.34MB
Image 4	2880x1800	626KB
Image 5	1680x1050	267KB
Image 6	3888x2592	3.86MB
Image 7	2560x1600	687KB
Image 8	2880x1800	2.04MB
Image 9	2560x1600	1.01MB
Image 10	2560x1600	424KB
Image 11	1920x1200	681KB
Image 12	1920x1200	1.01MB
Image 13	2560x1600	806KB
Image 14	1680x1050	126KB
Image 15	510x330	96.7KB
Image 16	800x591	284KB

Table A11.1. Dimensions and sizes of the images

Step 2. The next step is the development of two applications, using Visual Studio 2013 development tool and C# programming language, which display these pictures in a table. For each application we create 16 “Image” controls that have a predefined size of 100x100. In the first application we set to all the images the “build type” to “Content”, while in the second application we set this value to “Resource”. Setting this attribute to Content means that the images are included in XAP alongside the DLL, while setting it to Resource means that the images are embedded in DLL. Usually the type is set to Content if the developer wants a quick startup and to Resource when he wants a quick access to the images.

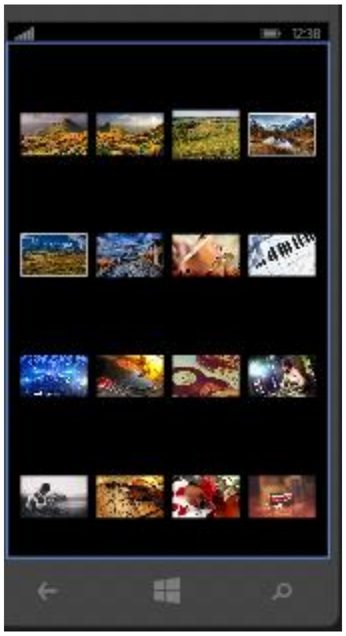


Figure A11.2. Application snapshot

Step 3. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 4. After we obtain the battery consumption, we transform it into energy consumption. For this transformation, we use the following formula:

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

The voltage value depends on the phone that is used. Consequently, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	Time (s)	Battery consumption (mAh)	Battery charge remaining (h)	Energy consumption (wh)
Resource	22.03	0.65	14.09	0.002405
Content	22.35	0.66	14.13	0.002442

Table A11.2. Build type property – energy consumption

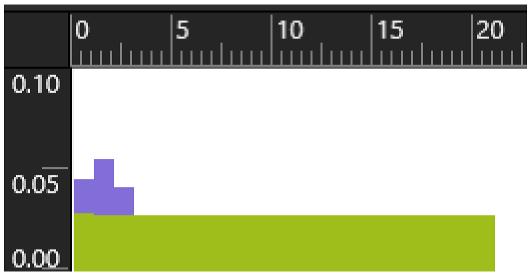


Figure A11.2. Resource

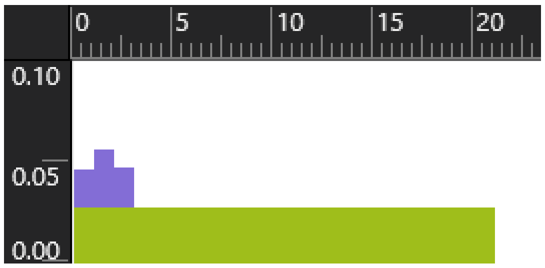


Figure A11.3. Content

Thread	Color	Description
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A11.3. Threads description

Axis	Description
X	Time (s)
Y	Battery consumption (mAh)

Table A11.4. Axis description

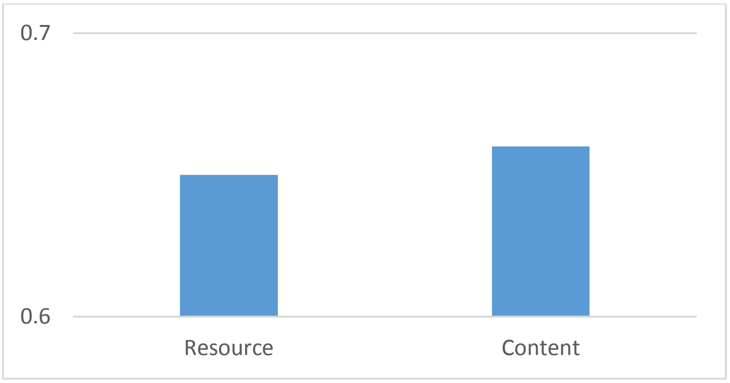


Chart A11.1. Build type property – energy consumption

Conclusions: As we can see from the table above there is a small difference, almost to 0, in terms of energy consumption. This happens because the application needs the same energy to decode and render all the images. It is not relevant where the images are stored, because the energy that is consumed in order to bring this images to UI is the same. From the Figure A11.2 and Figure A11.3 we can see that the energy consumed by the UI thread (green color) it is the same in the both cases. We can also notice that the application thread (purple color) consumes almost the same amount of energy and its distribution is very similar in both cases.

Appendix 12. Experiment 12 – Image format

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

Equipment: For this experiment the following components are needed:

- PC
  - Operating system: Windows 8.1
  - Development tools: Microsoft Visual Studio 2013
- Mobile phone
  - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. For this experiment we need a database of 16 images (downloaded from this website: http://wallpaperswide.com/music-desktop-wallpapers.html).The size and dimension for each picture are specific (details in Table A12.1). Below, there are some examples of the pictures that were used for this experiment:

Figure A12.1. Example of images

Image	Dimension	Size
Image 1	1920x1200	99.1 KB
Image 2	510x330	689KB
Image 3	2560x1600	1.34MB
Image 4	2880x1800	626KB
Image 5	1680x1050	267KB
Image 6	3888x2592	3.86MB
Image 7	2560x1600	687KB
Image 8	2880x1800	2.04MB
Image 9	2560x1600	1.01MB
Image 10	2560x1600	424KB
Image 11	1920x1200	681KB
Image 12	1920x1200	1.01MB
Image 13	2560x1600	806KB
Image 14	1680x1050	126KB
Image 15	510x330	96.7KB
Image 16	800x591	284KB

Table A12.1. Dimensions and sizes of the images

Step 2. Using Visual Studio 2013 development tool and C# programming language, we develop two applications which display these pictures in a table. For each application we create 16 images controls that have a predefined size of 100x100. In the first application we store all the pictures in a JPG format while in the second application we transform each picture into a string and store all the strings in a text file.

Base64 is an encoding scheme that transforms binary data to base 64 representation. For example the string "Hello" will be translated to "SGVsbG8=". The same representation can be applied to pictures. One disadvantage of this representation is the resizing of the pictures up to 33%.

Figure 12.2. Application snapshot

Step 3. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 4. After we obtain the battery consumption, we transform it into energy consumption. For this transformation, we use the following formula:

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

The voltage value depends on the phone that is used. Consequently, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	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

Table A12.2. Image format – energy consumption

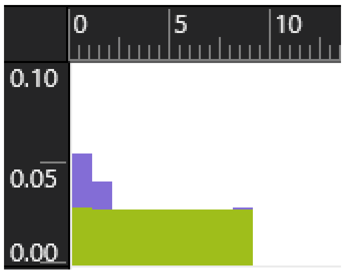


Figure A12.3. Base64

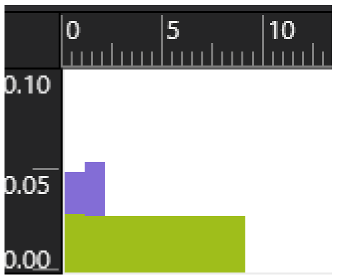


Figure A12.4 JPG

Thread	**Color**	**Description**
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A12.3. Threads description

Axis	**Description**
X	Time (s)
Y	Battery consumption (mAh)

Table A12.4. Axis description

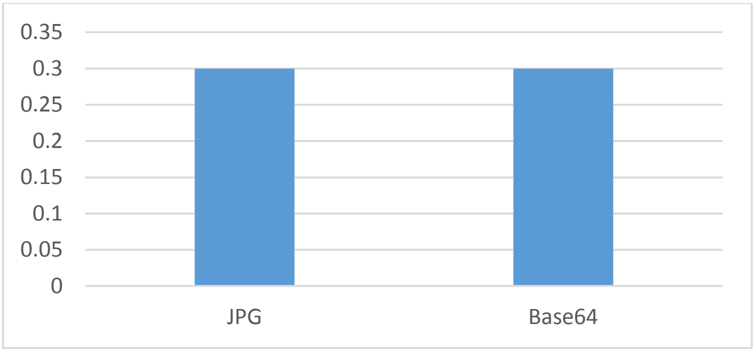


Chart A12.1. Image format – energy consumption

Conclusions: The battery consumption is equal in the both cases considered above, so it is not relevant if we keep images as JGP or as strings. Although the battery consumption is equal, we can notice the fact that the distribution of application thread is different. In “Figure A12.3” we can see that it requires a lot of energy for computation (purple color) at the beginning, but after it drops significantly. In the second case, we see that the time for all the computation is longer. The energy consumed by UI thread (green color) is similar in both cases.

Appendix 13. Experiment 13 – Loop instructions

Aim: To investigate the energy efficiency of two loops instructions: for and while.

Equipment: For this experiment the following components are necessary:

- PC
    - Operating system: Windows 8.1
    - Development tools: Microsoft Visual Studio 2013
- Mobile phone
    - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. Using Visual Studio 2013 development tool and C# programming language, we develop two applications, which count up to 500.000.000 and display a message after the loop is done. In the first application, a "for" loop is used, while in the second application we use a "while" loop. The difference between them is the syntax.

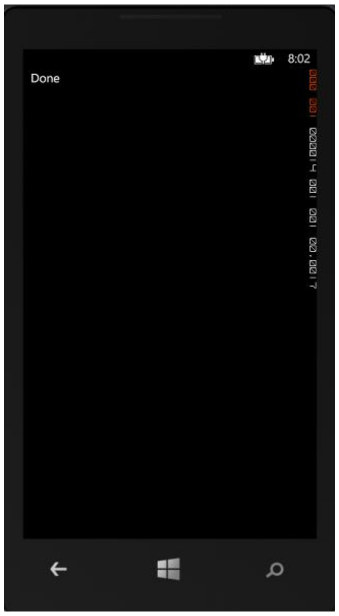


Figure A13.1. Application snapshot

Step 2. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 3. After we obtain the battery consumption, we transform it into energy consumption. For this transformation, we use the following formula:

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

The voltage value depends on the phone that is used. Consequently, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	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

Table A13.1. Loop instructions – energy consumption

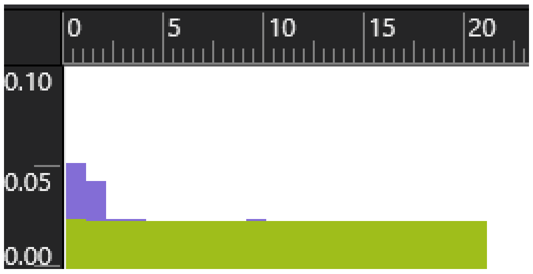


Figure A13.2. For loop

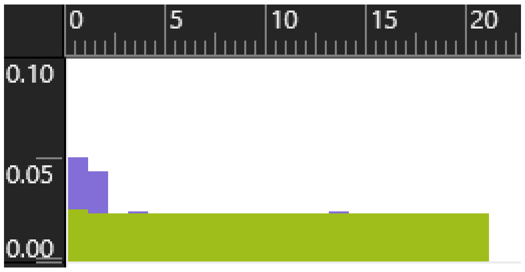


Figure A13.3. While loop

Thread	**Color**	**Description**
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A13.2. Threads description

Axis	**Description**
X	Time (s)
Y	Battery consumption (mAh)

Table A13.3. Axis description

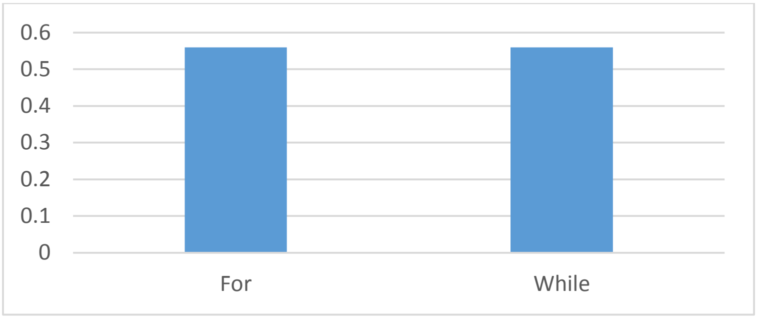


Chart A13.1. Loop instructions – energy consumption

Conclusions: As we can see in the results table, there is no difference between these two applications. This happens because, as we have already mentioned, the only difference between the two instructions is the syntax. From Figure A13.2 and Figure A13.3 we can see that the energy consumption distribution of both UI thread (green color) and application thread (purple color) is the same in both cases.

Appendix 14. Experiment 14 – Threads

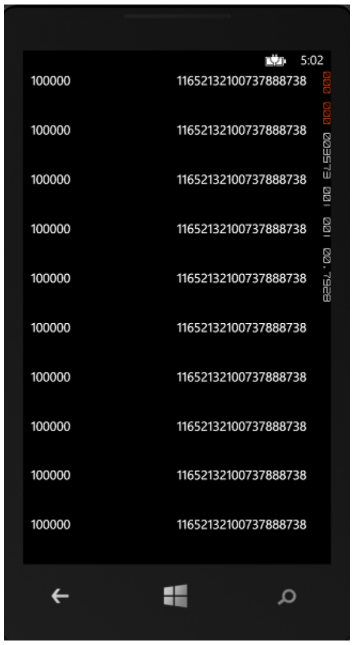
Aim: To investigate the energy efficiency of an application that uses one thread and of an application that uses more threads.

Equipment: For this experiment the following components are necessary:

- PC
  - Operating system: Windows 8.1
  - Development tools: Microsoft Visual Studio 2013
- Mobile phone
  - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1 Using Visual Studio 2013 development tool and C# programming language, we develop two applications, which display ten times the result of Fibonacci sequence with a length of 100.000 positions. In the first application we execute a calculation after the previous one is finished. In the second application, we use the multithreading concept thus each calculation is sent to a different thread. A thread ensures parallel execution of an operation, so all the operations are executed simultaneously.



```
100000                    11652132100737888738
100000                    11652132100737888738
100000                    11652132100737888738
100000                    11652132100737888738
100000                    11652132100737888738
100000                    11652132100737888738
100000                    11652132100737888738
100000                    11652132100737888738
100000                    11652132100737888738
100000                    11652132100737888738
```

Figure A14.1. application snapshot

Step 2. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 3. After we obtain the battery consumption, we transform it into energy consumption. For this transformation, we use the following formula:

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

The voltage value depends on the phone that is used. Consequently, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	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

Table A14.1. Threads – energy consumption

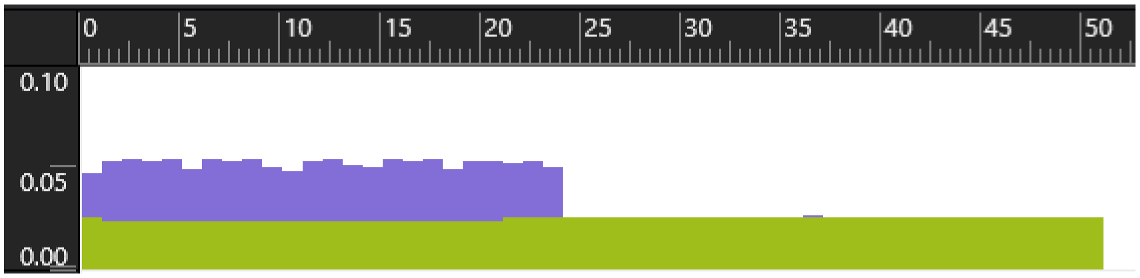


Figure A14.2 Single thread

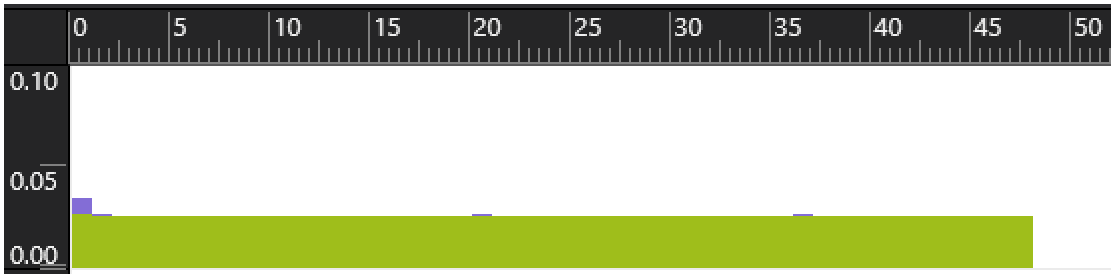


Figure A14.3. Multithreading

Thread	**Color**	**Description**
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A14.2. Threads description

Axis	**Description**
X	Time (s)
Y	Battery consumption (mAh)

Table A14.3. Axis description

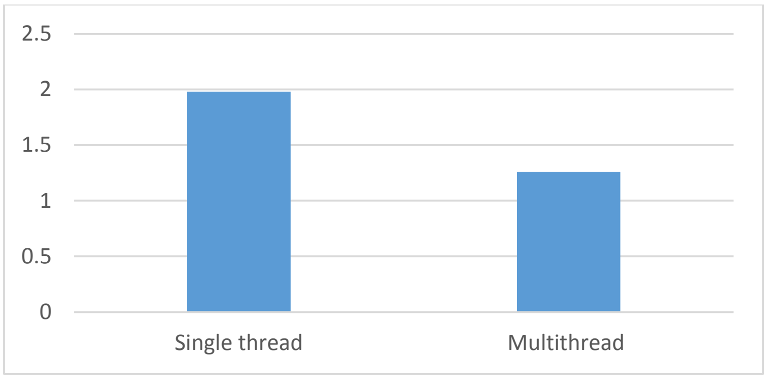


Chart A14.1. Threads – energy consumption

Conclusions: As we can see from the charts above, the difference between the two approaches is significant. From Figure A14.2 and Figure A14.3 we can notice that the energy used by the UI thread (green color) is the same in both cases. There is a big difference in application thread (purple color). For the single thread, we can observe 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. We can also notice that in the first case the application is frozen for the first 25 seconds, while the second application can be used immediately.

Appendix 15. Experiment 15 – Method for data loading

Aim: To investigate the energy efficiency of two applications that do heavy processing in constructor, respectively in OnNavigateTo event.

Equipment: For this experiment the following components are necessary:

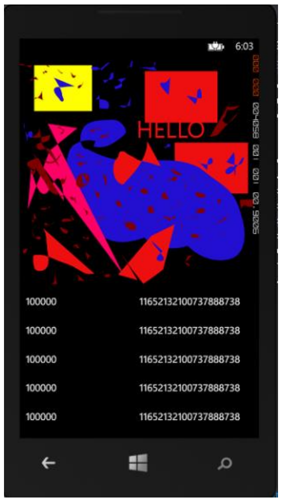
- PC
    - Operating system: Windows 8.1
    - Development tools:
        - Microsoft Visual Studio 2013
        - Microsoft Expression Design 4
- Mobile phone
    - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. Firstly, for these applications, we create a XAML file. This file is created using Microsoft Expression Design 4. This is a tool that is used by developers who create graphic interfaces. The XAML files are Microsoft extensions of Extensible Markup Language and are used for creating User Interface pages. The output that we have obtained using this software is the following:

Figure A15.1. The PNG format

Step 2. Next, we integrate this image with a heavy processing operation. For this experiment we develop two application, using Visual Studio 2013 development tool and C# programming language, which display five times the result of Fibonacci sequence, of 100.000 positions together with the image above. In the first application we will execute Fibonacci’s function in the page’s constructor, while in the second application we will execute it in OnNavigateTo event. The code in the page’s constructor is run before the first frame of the application is shown while the code in OnNavigateTo event is run after the page becomes active.



```
100000          1165213210073788738
100000          1165213210073788738
100000          1165213210073788738
100000          1165213210073788738
100000          1165213210073788738
```

Figure A15.2 Application snapshot

Step 3. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 4. After we obtain the battery consumption, we transform it into energy consumption. For this transformation, we use the following formula:

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

The voltage value depends on the phone that is used. Consequently, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (wh)**
Constructor	32.14	1.19	11.25	0.004403
OnNavigateTo	31.78	1.18	11.18	0.004366

Table A15.1. Data loading – energy consumption

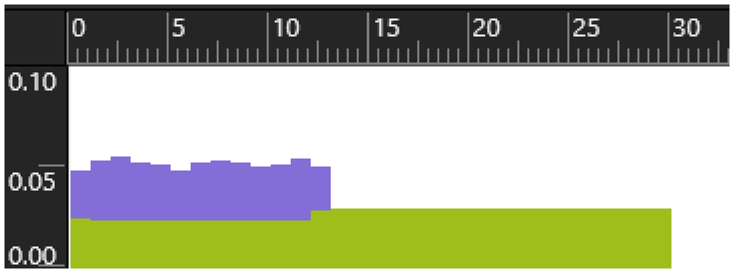


Figure A15.3 OnNavigateTo

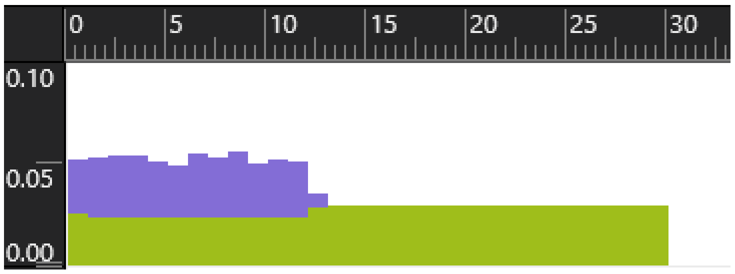


Figure A15.4. Constructor

Thread	**Color**	**Description**
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A15.2. Threads description

Axis	**Description**
X	Time (s)
Y	Battery consumption (mAh)

Table A15.3. Axis description

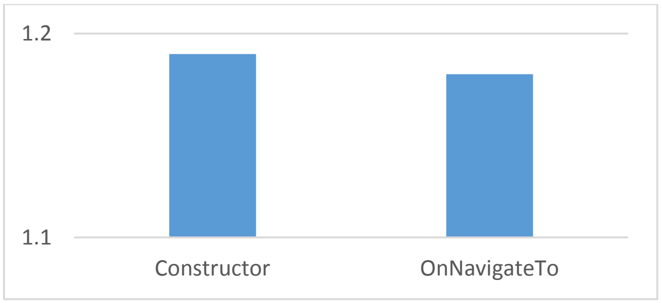


Chart A15.1. Data loading – energy consumption

Conclusions: In this experiment we determined that there is no difference in terms of energy consumption between loading heavy processing operation in the constructor or in OnNavigateTo event. From Figure A15.3 and Figure A15.4 we can see that distribution of energy consumption is similar in both cases. The energy consumed by UI thread (green color) has an identical distribution while the energy consumed by application thread (purple color) has a similar distribution. We can also notice that the time required for the data processing is similar in the both cases. This happens because the data processing is the same and it is of no importance where the operations are executed.

Appendix 16. Experiment 16 – Function type

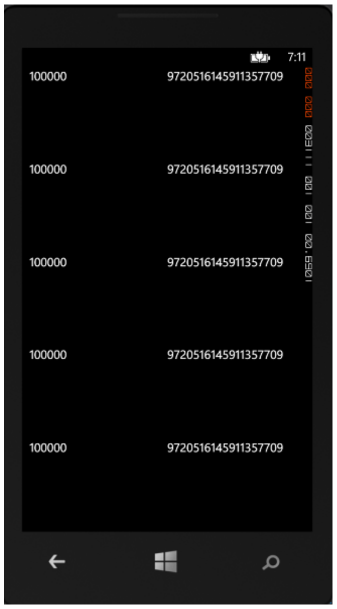
Aim: To investigate the energy efficiency of an application that uses an iterative function compared to an application that uses a recursive function.

Equipment: For this experiment the following components are necessary:

- PC
  - Operating system: Windows 8.1
  - Development tools: Microsoft Visual Studio 2013
- Mobile phone
  - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. Using Visual Studio 2013 development tool and C# programming language, we develop two applications, which display five times the 5.000 position of Fibonacci sequence. In the first application we execute each calculation using an iterative function. In the second application we use a recursive function that will do the same operation. A recursive function is a function that calls itself. The function that does not calls itself it is an iterative function.



```
7:11
100000          9720516145911357709

100000          9720516145911357709

100000          9720516145911357709

100000          9720516145911357709

100000          9720516145911357709
```

Figure A16.1 Application snapshot

Step 2. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 3. After we obtain the battery consumption, we transform it into energy consumption. For this transformation, we use the following formula:

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

The voltage value depends on the phone that is used. Consequently, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	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 A16.1. Function type – energy consumption

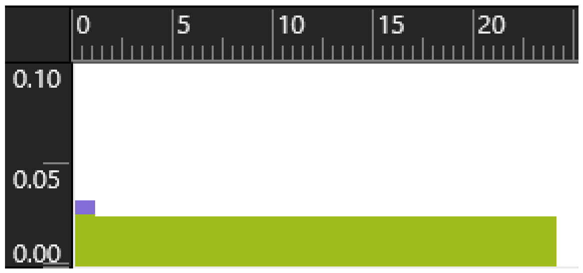


Figure A16.2 Iterative

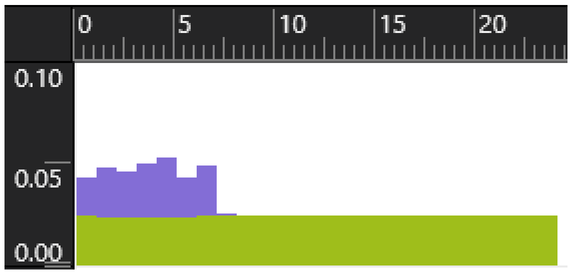


Figure A16.3. Recursive

Thread	**Color**	**Description**
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A16.2. Threads description

Axis	**Description**
X	Time (s)
Y	Battery consumption (mAh)

Table A16.3. Axis description

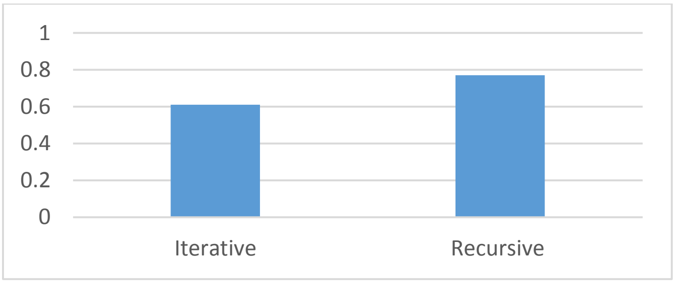


Chart A16.1. Function type – energy consumption

Conclusions: The application that uses an iterative function is more efficient according to the graphs above. We notice 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 and he can use the application. In the case of the iterative function the amount of energy that is required is very low. Furthermore, we notice in this case that the application is faster due to the fact that the thread is busy for less time. The energy consumed by the UI thread (green color) is similar in both cases.

Appendix 17. Experiment 17 – StackPanel control

Aim: To investigate the energy efficiency of a StackPanel control compared to a VirtualizingStackPanel control.

Equipment: For this experiment the following components are necessary:

- PC
  - Operating system: Windows 8.1
  - Development tools: Microsoft Visual Studio 2013
- Mobile phone
  - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. Using Visual Studio 2013 development tool and C# programming language, we develop two applications, which display a list with 400 elements. Each element contains an image (dimension: 256x256, size: 32.5KB) and a text. For displaying these items we use a ListBox control and as template for the list we use in one application a StackPanel control, while in the other application a VirtualizingStackPane. If we are using a StackPanel inside the ListBox means that all the items will be loaded when the list is loaded. In the other case, are loaded only the items that are visible for the user. When the user scrolls down, another request will be made.

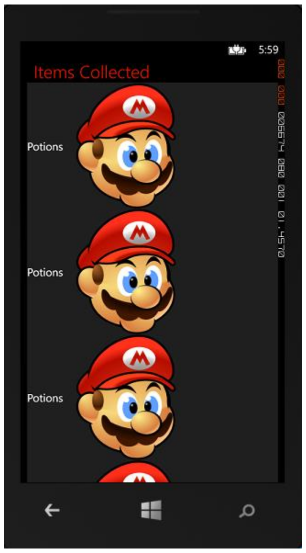


Figure A17.1. Application snapshot

Step 3. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 4. After we obtain the battery consumption, we transform it into energy consumption. For this transformation, we use the following formula:

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

The voltage value depends on the phone that is used. Consequently, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (wh)**
StackPanel (without scrolling)	22.56	0.71	13.73	0.002627
VirtualizingStackPanel (without scrolling)	20.57	0.55	17.72	0.002035
StackPanel	20.76	1.38	6.26	0.005106

(with scrolling)				
VirtualizingStackPanel (with scrolling)	20.85	1.31	6.64	0.004847

Table A17.1. StackPanel control – energy consumption

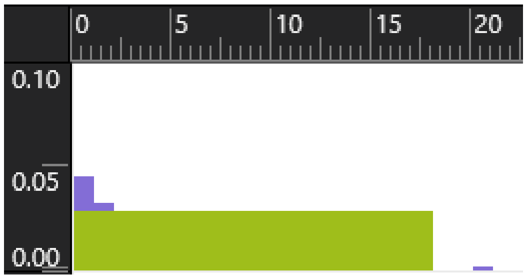


Figure A17.2 VirualizingStackPanel – without scrolling

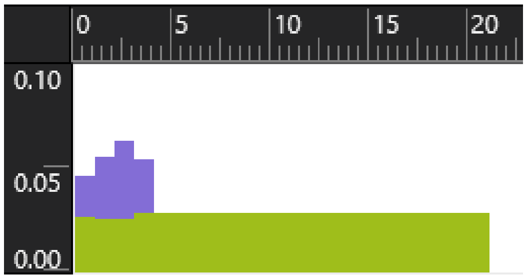


Figure A17.3 StackPanel – without scrolling

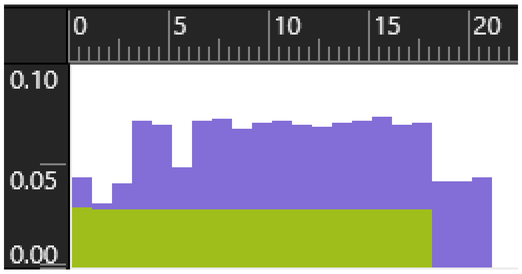


Figure A17.4. VirtualizingStackPanel- with scrolling

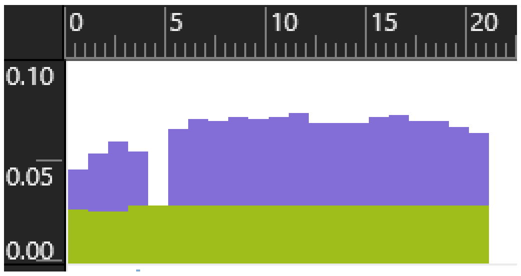


Figure A17.5. StackPanel – with scrolling

Thread	**Color**	**Description**
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A17.2. Threads description

Axis	**Description**
X	Time (s)
Y	Battery consumption (mAh)

Table A17.3. Axis description

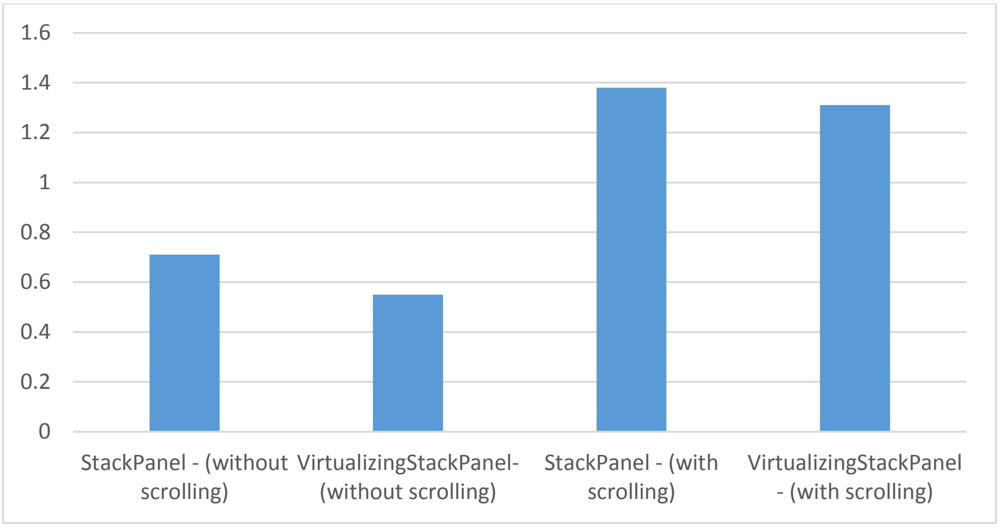


Chart A17.1. StackPanel control – energy consumption

Conclusions: We analyzed in this experiment two cases: when a user uses the scroll and when the user does not. We choose both cases because they are frequently in the behavior of the user. In the first case when the user does not scroll we see that the energy consumption between these two controls differs a lot. VirtualizingStackPanel is more efficient and also faster to load the data. This happens because only the data that are visible for the user are requested. We can notice that the UI thread (green color) consumes the same amount of energy in both cases, but the application thread (purple color) is different. For the VirtualizingStackPanel we have some energy consumed when the application is launched, while the StackPanel consumes more energy for a longer period of time. In the second case, we can observe that the difference decreases, because using the scroll, more data is requested all the time. So, when scrolling the energy consumption is less for the VirtualizingStackPanel, similar to the first case when the user does not scroll. We can see from Figure A17.4 and Figure A17.5 that the consumed energy is similar because the energy consumption distribution become similar. The startup of the application is faster when we are using VirtualizingStackPanel, but the scroll is faster when we are using StackPanel because all the data is already loaded.

Appendix 18. Experiment 18 – Assemblies

Aim: To investigate the energy efficiency of an application that stores external pages in another assembly compared to an application that stores the all the pages in the same assembly.

Equipment: For this experiment the following components are necessary:

- PC
    - Operating system: Windows 8.1
    - Development tools: Microsoft Visual Studio 2013
- Mobile phone
    - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. Using Visual Studio 2013 development tool and C# programming language, we develop two applications, which navigate through three pages. In the first application we have all of the pages in the same project and we load them when the application runs. In the second case we split the solution in two projects: one project that contains the main page and another project, of type Windows Phone Class Library, which contains two external pages. The pages from the latter project will be loaded only on demand. In our application we have two buttons that will navigate through the pages. For the data analyzing we chose two cases: when the user navigates through the pages and when the user does not.

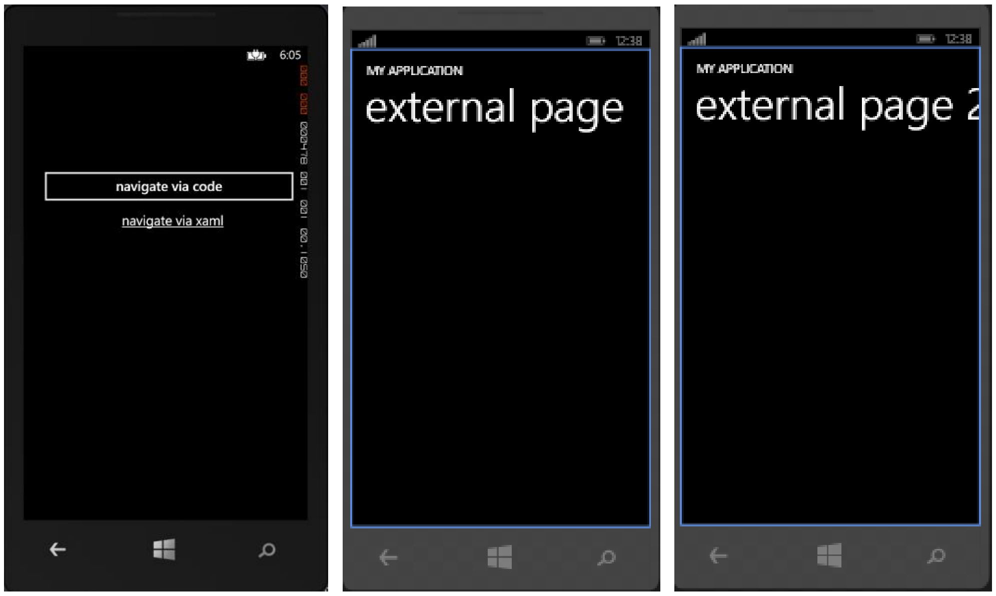


Figure A18.1 Application snapshot

Step 2. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 3. After we obtain the battery consumption, we transform it into energy consumption. For this transformation, we use the following formula:

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

The voltage value depends on the phone that is used. Consequently, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (wh)**
One assembly (without navigation)	22.46	0.46	18.59	0.001702
Two assemblies (without nagivation)	20.55	0.46	18.67	0.001702
One assembly (with navigation)	26.31	0.61	18.49	0.002257
Two assemblies (with navigation)	26.98	0.61	18.45	0.002257

Table A18.1. Assemblies – energy consumption

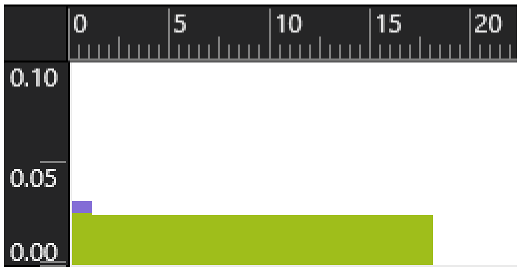


Figure A18.2. One assembly – without navigation

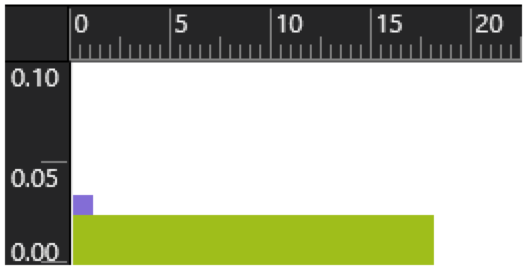


Figure A18.3Two assemblies – without navigation

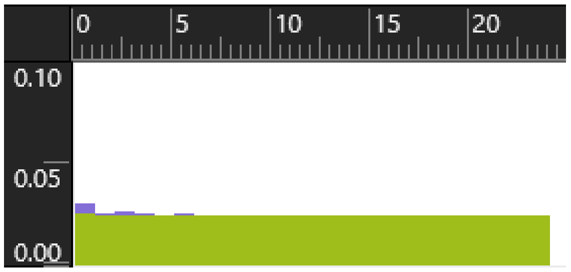


Figure A18.4. One assembly with navigation

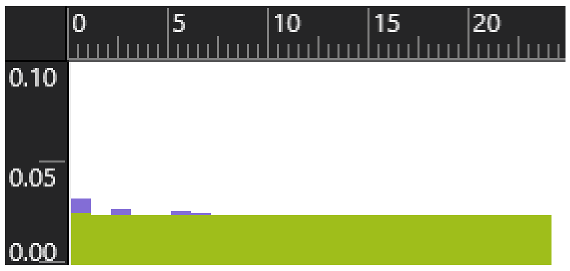


Figure A18.5 Two assemblies without navigation

Thread	**Color**	**Description**
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A18.2. Threads description

Axis	**Description**
X	Time (s)
Y	Battery consumption (mAh)

Table A18.3. Axis description

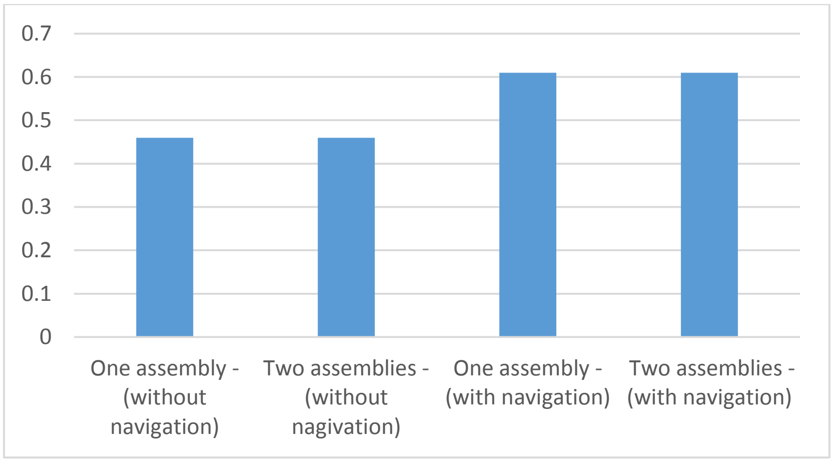


Chart A18.1. Assemblies – energy consumption

Conclusions: From the tables above we can observe that in both cases, when the user navigates through the pages and when the user does not, the energy consumption is the same. If we look very carefully at the first graph, we can see a very small difference, but it is not relevant for the result. It is possible that for applications with large numbers of pages, this difference to be more considerable. In the second case, we can see that the distribution of the energy is differs, but again not significantly.

Appendix 19. Experiment 19 – 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.

Equipment: For this experiment the following components are necessary:

- PC
  - Operating system: Windows 8.1
  - Development tools: Microsoft Visual Studio 2013
- Mobile phone
  - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. Using Visual Studio 2013 development tool and C# programming language, we develop two applications, which display an animation. In the first application we create the animation in the XAML file, while in the second application the animation is created in procedural code. The difference between these methods of creating an animation consists in the execution of the animation made by the composition thread, in the first case, and of the execution by the UI thread, in the second case.

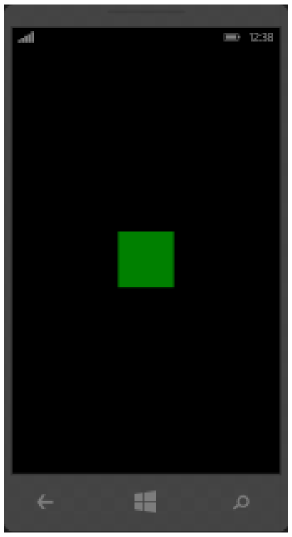


Figure A19.1 application snapshot

Step 2. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 3. After we obtain the battery consumption, we transform it into energy consumption. For this transformation, we use the following formula:

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

The voltage value depends on the phone that is used. Consequently, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	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

Table A19.1. Animations – energy consumption

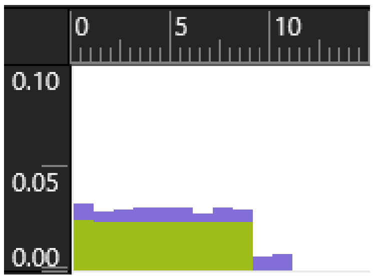


Figure A19.2. Code behind

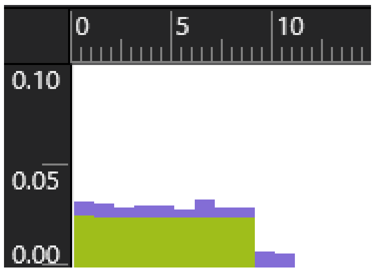


Figure A19.3 XAML

Thread	**Color**	**Description**
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A19.2. Threads description

Axis	**Description**
X	Time (s)
Y	Battery consumption (mAh)

Table A19.3. Axis description

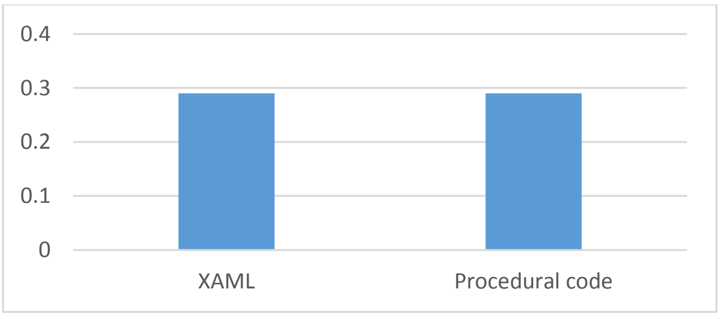


Chart A19.1. Animations – energy consumption

Conclusions: As we can observe from the charts above 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. We notice that running the animation in the composition thread or in the UI thread gives us the same effect. It might be possible to find some differences if the UI thread is overloaded. From the Figure A19.2 and Figure A20.3 we can observe 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.

Appendix 20. Experiment 20 – Storing images

Aim: To investigate the impact of displaying a set of images that are stored locally in comparison with a set of images that are stored in a web page.

Equipment: For this experiment the following components are necessary:

- PC
    - Operating system: Windows 8.1
    - Development tools: Microsoft Visual Studio 2013
- Mobile phone
    - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. For this experiment we need a database of 3 images (downloaded from this website: http://www.dannyst.com/). Each picture has specific size and dimension (details in Table A20.1). Below are the examples of the pictures used for this experiment:

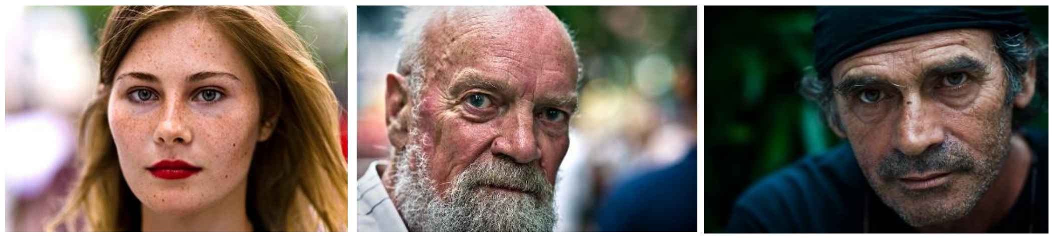


Figure A20.1. Example of images

Image	Dimension	Size
Image 1	875x581	345 KB
Image 2	875x581	437 KB
Image 3	875x581	278 KB

Table A20.1. Dimensions and sizes of the images

Step 2. Using Visual Studio 2013 development tool and C# programming language, we develop two applications which display these pictures in a table. For each application we create 3 "Image" controls. In the first application we store the images locally, while in the second application the images are stored in a web page.

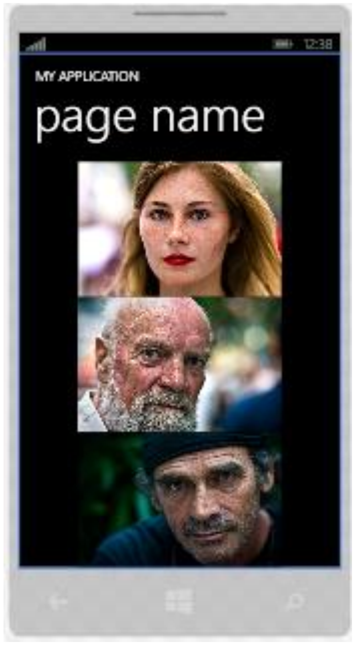


Figure A20.2. Application snapshot

Step 3. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 4. After we obtain the battery consumption, we transform it into energy consumption. For this transformation we use 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, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	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

Table A20.2. Storing images – energy consumption

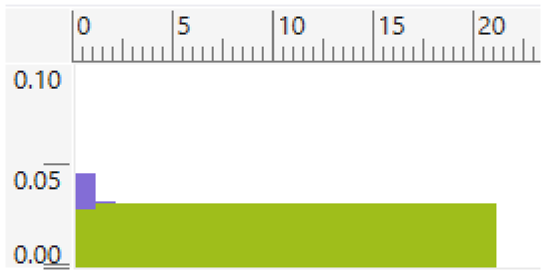


Figure 20.3. Local source

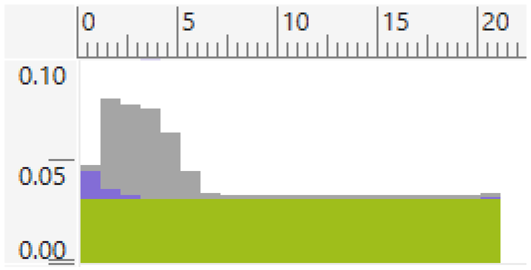


Figure A20.4 Internet source

Thread	**Color**	**Description**
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A20.3. Threads description

Axis	**Description**
X	Time (s)
Y	Battery consumption (mAh)

Table A20.4. Axis description

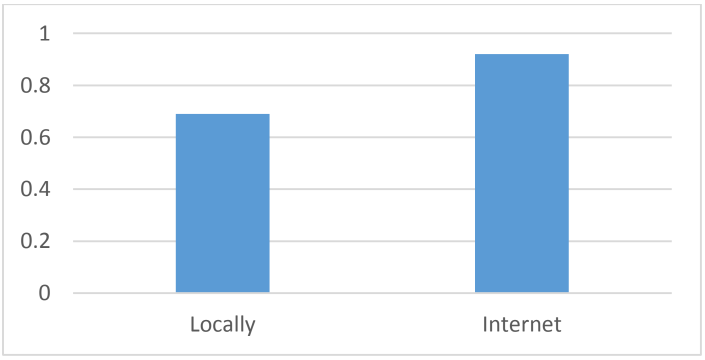


Chart A20.1. Storing images – energy consumption

Conclusions: 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 Figure A20.3 and Figure A20.4 we can notice 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 presented in Figure A20.3 shows there is no energy consumed by the network while the one in Figure A20.4 shows a significant amount of energy that is consumed by the network.

Appendix 21. Experiment 21 – Playing videos

Aim: To investigate the impact of playing a video stored locally compared to a video that is stored in a web page.

Equipment: For this experiment the following components are necessary:

- PC
  - Operating system: Windows 8.1
  - Development tools: Microsoft Visual Studio 2013
- Mobile phone
  - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. For this experiment we need a video (downloaded from this website: http://download.wavetlan.com/SVV/Media/HTTP/http-mp4.htm).

Step 2. Using Visual Studio 2013 development tool and C# programming language, we develop two applications which play a video. For each application we create a “MediaElement” control. In the first application we store the video locally, while in the second application the video is stored on a web page. A “MediaElement” control is an object that contains video, audio or both.

Step 3. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 4. After we obtain the battery consumption, we transform it into energy consumption. For this transformation we use 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, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	Time (s)	**Battery consumption (mAh)**	**Battery charge remaining (h)**	**Energy consumption (wh)**
From internet	45.57	1.80	10.72	0.00666
Stored locally	46.77	1.89	10.29	0.006993

Table A21.1 Playing videos – energy consumption

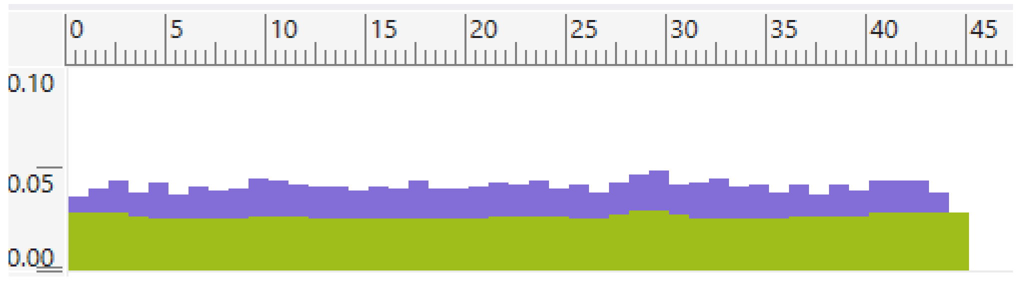


Figure A21.1 20Video stored locally

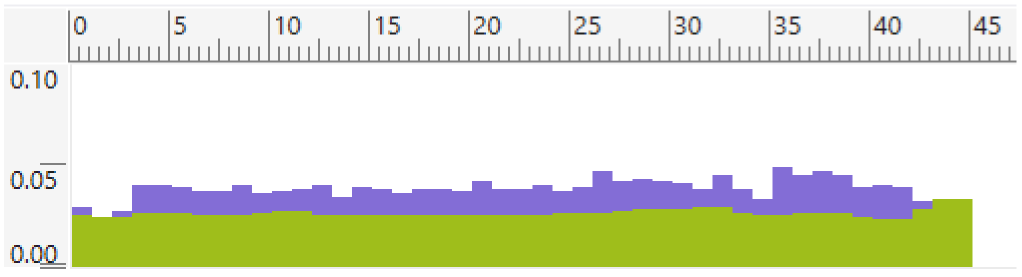


Figure A21.2 Video stored in a web page

Thread	Color	Description
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A21.2. Threads description

Axis	Description
X	Time (s)
Y	Battery consumption (mAh)

Table A21.3. Axis description

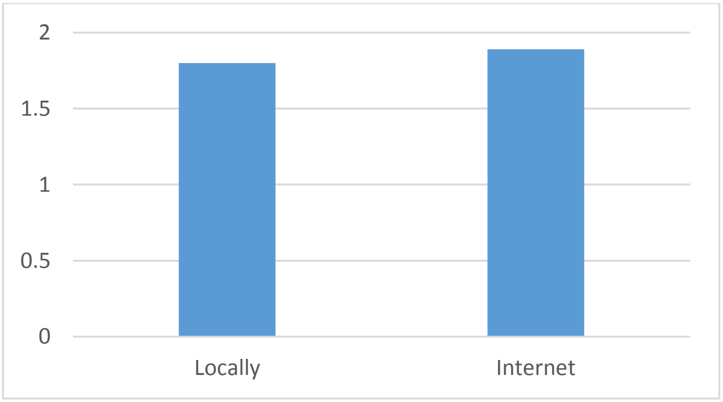


Chart A21.1. Playing videos – energy consumption

Conclusions: In Table A21.1 can be noticed that the energy consumption is similar in both cases. Analyzing Figure A21.1 and Figure A21.2 we can say that the energy consumed by the UI thread (green), the energy consumed by application thread (purple) and the energy consumed by network thread (gray) have similar values and distribution. We can also notice the fact that the energy consumed by the network is 0. This phenomenon appears because “MediaElement” control uses Windows Media Player internally for downloading the video. Being independent from our application, the energy consumed by this tool it is not included in the final result. We can just assume that the total amount of energy is bigger when the video is stored on a web page because there is extra energy consumed by the network.

Appendix 22. Experiment 22 – Playing audio files

Aim: To compare the impact of playing an audio file that is stored locally to an audio file that is stored in a web page.

Equipment: For this experiment the following components are necessary:

- PC
  - Operating system: Windows 8.1
  - Development tools: Microsoft Visual Studio 2013
- Mobile phone
  - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. For this experiment we need an audio file (downloaded from this website: http://www.tonecuffe.com/mp3/).

Step 2. Using Visual Studio 2013 development tool and C# programming language, we develop two applications which play an audio file. For each application we create a "MediaElement" control. In the first application we store the audio file locally, while in the second application the audio file is stored on a web page. A "MediaElement" control is an object that contains video, audio or both.

Step 3. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 4. After we obtain the battery consumption, we transform it into energy consumption. For this transformation we use 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, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	Time (s)	Battery consumption (mAh)	Battery charge remaining (h)	Energy consumption (wh)
From internet	93.32	2.38	16.35	0.008806
Stored locally	93.10	2.32	16.72	0.008584

Table A22.1. Playing audio files – energy consumption

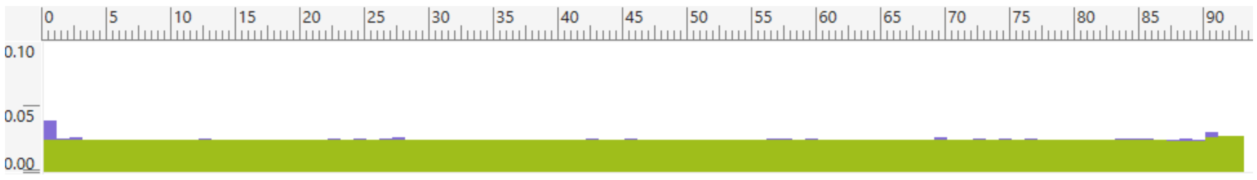


Figure A22.1 Audio file stored in a web page

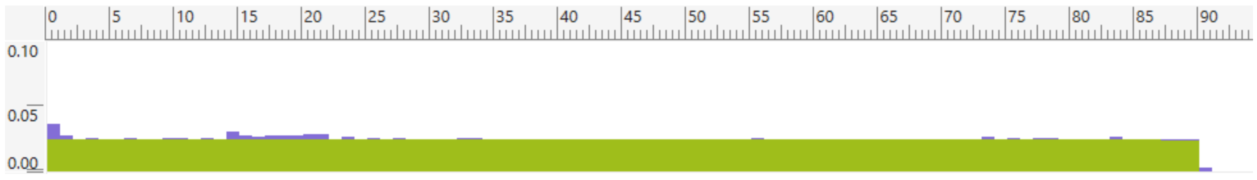


Figure A22.2 Audio file stored locally

Thread	Color	Description
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A22.2. Threads description

Axis	Description
X	Time (s)
Y	Battery consumption (mAh)

Table A22.3. Axis description

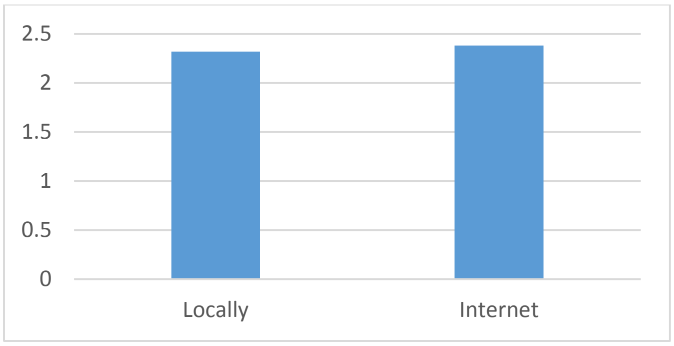


Chart A22.1. Playing audio files – energy consumption

Conclusions: In Table A22.1 can be noticed that the energy consumption is similar in both cases. Analyzing Figure A22.1 and Figure A22.2, we can say that the energy consumed by the UI thread (green), the energy consumed by application thread (purple) and the energy consumed by network thread (gray) have similar values and distribution. We can notice also the fact that the energy consumed by network is 0. This phenomenon appears because "MediaElement" control uses Windows Media Player internally for downloading the audio file. Being independent from our application the energy consumed by this tool it is not included in the final result. We can just assume that the total amount of energy is bigger when the audio file is stored on a web page because there is extra energy consumed by the network.

Appendix 23. Experiment 23 – Image format (JPG vs 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.

Equipment: For this experiment the following components are necessary:

- PC
  - Operating system: Windows 8.1
  - Development tools: Microsoft Visual Studio 2013
- Mobile phone
  - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. For this experiment we need a database of 6 images (stored on this website: https://www.tumblr.com/blog/vladcristeacont). The same image is stored in two different formats: JPG and PNG. Each picture has specific size and dimension. Below, there are the examples of the pictures used for this experiment:

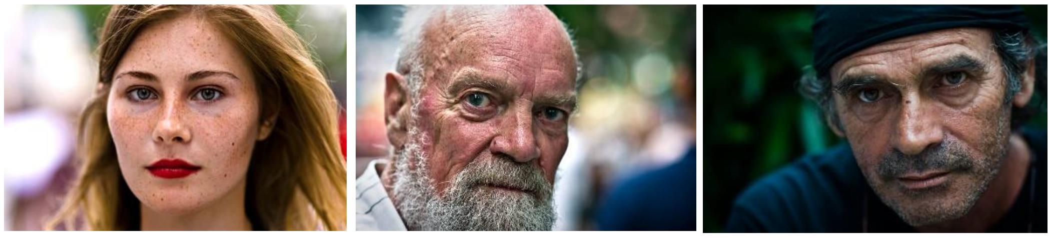


Figure A23.1. Example of images

Image	**Dimension**	**Size**
Image 1	875x581	345 KB
Image 2	875x581	437 KB
Image 3	875x581	278 KB
Image 4	875x581	623 KB
Image 5	875x581	583 KB
Image 6	875x581	476 KB

Table A23.1. Dimensions and sizes of the images

Step 2. Using Visual Studio 2013 development tool and C# programming language, we develop two applications which display these pictures in a table. For each application we create 3 "Image" controls. In the first application, we will use as source only images in JPG format, while in the second we will use images in PNG format. The original pictures were in JPG format and they were transformed in PNG format using this website: http://image.online-convert.com/convert-to-png.

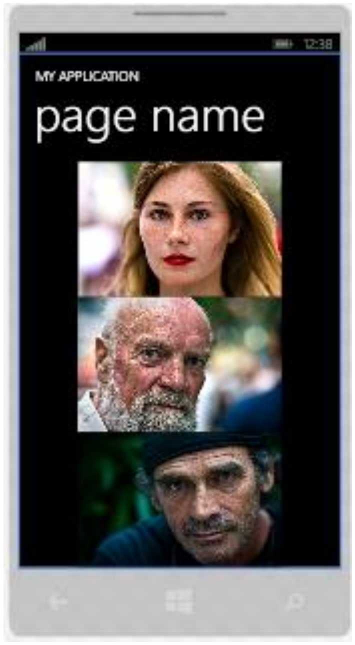


Figure A23.2 Application snapshot

Step 3. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 4. After we obtain the battery consumption, we transform it into energy consumption. For this transformation we use 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, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	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

Table A23.2 Image format in clouds – energy consumption

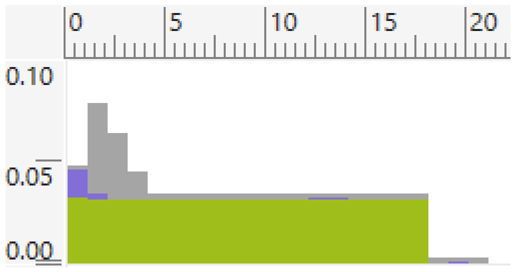


Figure A23.3 JPG format

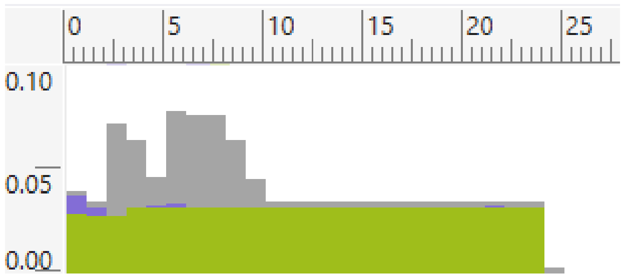


Figure A23.4 PNG source

Thread	Color	Description
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A23.3. Threads description

Axis	Description
X	Time (s)
Y	Battery consumption (mAh)

Table A23.4. Axis description

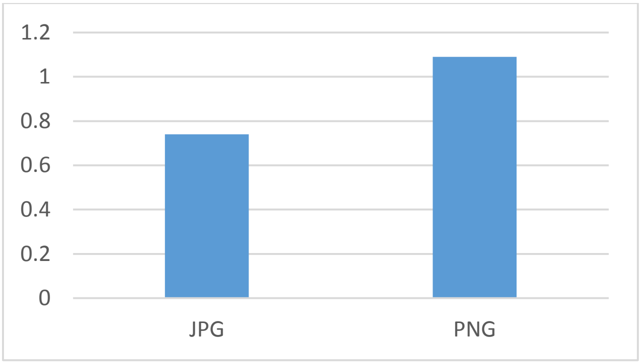


Chart A23.1. Image format in clouds – energy consumption

Conclusions: 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 A23.2, we can notice that the difference between these two formats is significant. If we are looking at Figure A23.3 and Figure A23.4, we can observe 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, 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.

Appendix 24. Experiment 24 – 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.

Equipment: For this experiment the following components are necessary:

- PC
  - Operating system: Windows 8.1
  - Development tools: Microsoft Visual Studio 2013
- Mobile phone
  - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1. For this experiment we need an image (dimension: 875x561, size: 278KB, stored on this website: https://www.tumblr.com/blog/vladcristeacont). Below, there is the picture that was used for this experiment:

Figure A24.1. Example of images

Step 2. Using Visual Studio 2013 development tool and C# programming language, we develop two applications which display this picture in a table. For each application we create 3 “Image” controls. In one application, firstly we will download and display the image. After 10 seconds we will display the image from the local source. We will repeat this again after another 10 seconds. In the other application we will set the source of the first control to a specific URL. After 10 seconds we will set the source for the second control and after another 10 seconds the source for the third control.

Figure A24.2 Application snapshot

Step 3. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 4. After we obtain the battery consumption, we transform it into energy consumption. For this transformation we use 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, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

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

Table A24.1. Images- multiple access – energy consumption

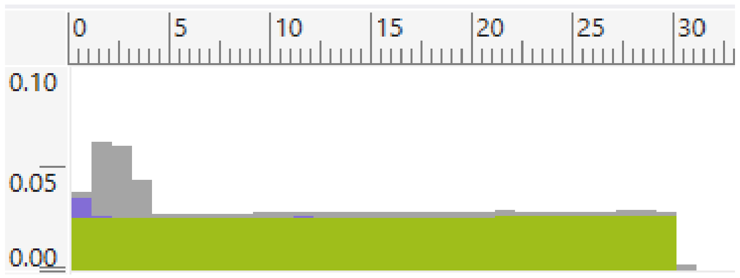


Figure A24.3 Application that display the image from the same URL

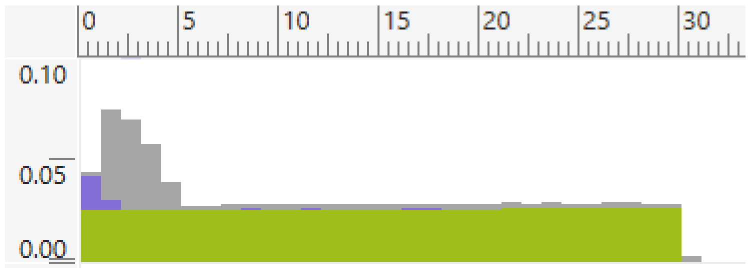


Figure A23.4 Application that save and display the images

Thread	Color	Description
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A24.2. Threads description

Axis	Description
X	Time (s)
Y	Battery consumption (mAh)

Table A24.3. Axis description

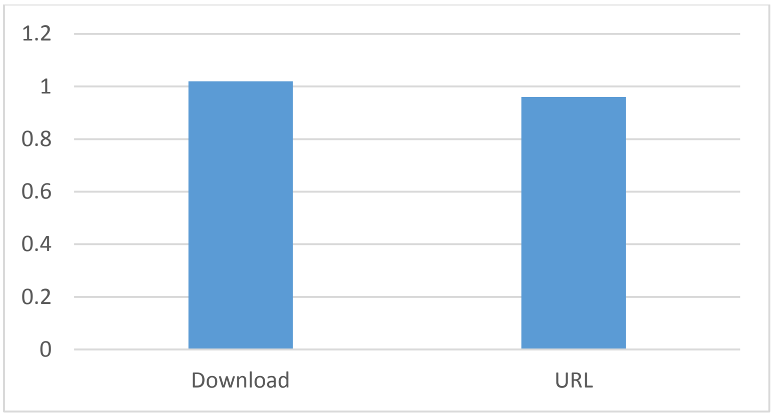


Chart A24.1. Images- multiple access – energy consumption

Conclusions: From this experiment we can notice that the application which displays the images without saving them consumes less energy than the application which downloads first the picture. If we are looking at Figure A24.3 and Figure A24.4, we can see that the energy consumed by the UI 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.

Appendix 25. Experiment 25 – Heavy processing operations

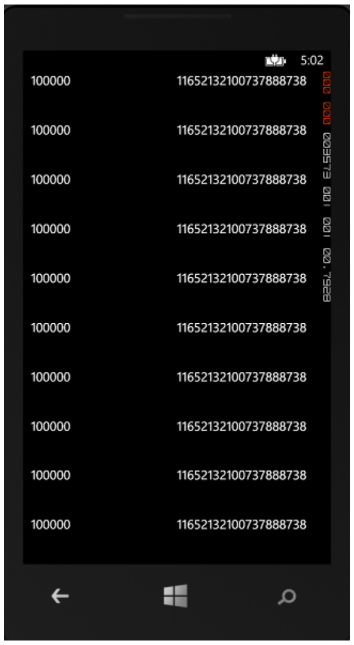
Aim: To compare the impact on energy consumption of an operation that is run locally to an operation that is run in clouds.

Equipment: For this experiment the following components are necessary:

- PC
  - Operating system: Windows 8.1
  - Development tools: Microsoft Visual Studio 2013
- Mobile phone
  - Operating system: Windows Phone 8.1

Experiment procedure:

Step 1 Using Visual Studio 2013 development tool and C# programming language, we develop two applications which display 10 times the result of Fibonacci sequence with a length of 100.000 positions. In the first application we will execute all of these operations locally, while in the second application we will run the operations in the clouds.



```
100000          1165213210073788873 8
100000          1165213210073788873 8
100000          1165213210073788873 8
100000          1165213210073788873 8
100000          1165213210073788873 8
100000          1165213210073788873 8
100000          1165213210073788873 8
100000          1165213210073788873 8
100000          1165213210073788873 8
100000          1165213210073788873 8
```

Figure A25.1 Application snapshot

Step 2. For each application we measure the battery consumption and the battery charge remaining. They are measured using Windows Phone Application Analysis tool, which is integrated in Visual Studio 2013 tool. The outputs of this analysis are the battery consumption measured in mAh (miliampere-hour) and the battery charge remaining, measured in h (hours).

Step 3. After we obtain the battery consumption, we transform it into energy consumption. For this transformation we use 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, we took this value for a specific phone: Nokia Lumia 1320. For this particular phone the voltage is 3.7 Volts.

Results:

	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

Table A25.1. Heavy processing operations – energy consumption

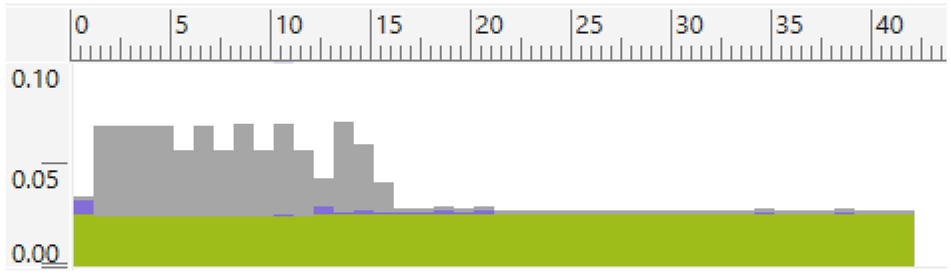


Figure A25.2. Cloud processing

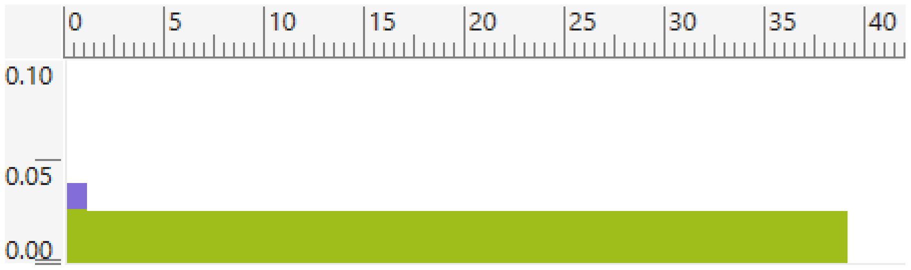


Figure A25.3. Locally execution

Thread	**Color**	**Description**
UI thread	Green	Energy consumption of the UI
Application thread	Purple	Energy consumption of the application that is not included in UI
Network thread	Grey	The network energy consumption

Table A25.2. Threads description

Axis	**Description**
X	Time (s)
Y	Battery consumption (mAh)

Table A25.3. Axis description

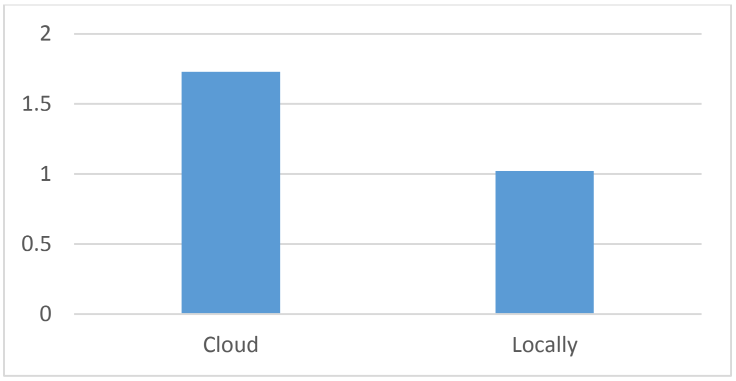


Chart A25.1. Heavy processing operations – energy consumption

Conclusions: The execution of some operations can influence significantly the energy consumption of an application. We can see in this experiment that executing some operations locally can save a lot of energy. From Table A25.1 we can notice that the difference between these two applications is significant. If we analyze Figure A25.2 and Figure A25.3, we can notice that the UI thread (green) consumes the same amount of energy in both cases. In Figure A25.3 we see that the application thread (purple) request some energy only at the begging while processing the data. For the other application the application thread consumes energy during the execution of the application because the data received from server has to be processed. The network thread (gray) makes the 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.