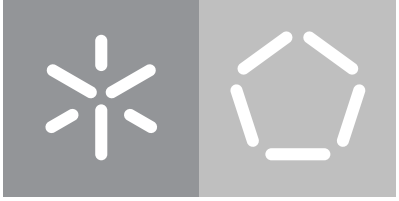
Universidade do Minho
Escola de Engenharia

Nelson Adriano Sequeira Gonçalves

Browser Energy Efficiency in Android

January, 2022



Universidade do Minho
Escola de Engenharia

Nelson Adriano Sequeira Gonçalves

Browser Energy Efficiency in Android

Master's Dissertation
Mestrado Integrado em Engenharia Informática

Work supervised by
João Alexandre Baptista Vieira Saraiva
Rui Alexandre Afonso Pereira

January, 2022

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the Universidade do Minho.

Acknowledgements

After I overcame the last challenge in my academic path, I want to thank the people who supported me and helped me achieve this goal.

First of all, I want to give my sincere thanks to my teacher and supervisor, João Saraiva, for his availability, knowledge, persistence, and above all, unconditional support in the obstacles that appeared along the way.

I also want to leave my thanks to my teacher and co-supervisor, Rui Pereira, for his knowledge in Green Software and help in all the problems and decisions overcome in the realization of this path.

My thanks also go to my teacher, Jácome Cunha, for his unconditional support, especially in this final stage, and for his knowledge in statistical analysis that allowed better research and conclusion of this path.

I also want to leave my thanks to all members of the Green Software Lab, especially to Rui Rua, for his willingness to help whenever necessary, without any barrier in communication, and for providing the tool reused by him, RERAN.

Finally, I want to leave my thanks for all the support given by my family, friends, and especially my girlfriend, who helped at all times to achieve this path.

Abstract

Nowadays, there is a massive growth in energy consumption in the IT sector, which is leaving a huge footprint in terms of energy consumption despite its benefits. With this, the topic of energy consumption and how to improve it has become one of the most talked-about topics today.

Several developments have been made to find the most efficient solutions to the various problems that users and developers encounter. But this is far from being an easy task for both, as there is still very little information available, or sometimes the solutions don't meet the needs of each one.

With this in mind, this dissertation aims to verify which Browser is more efficient in the Android environment since there is not much information in this area. For this, we selected seven browsers and ran four test scenarios in order to force the browsers. To test, we recorded a script for each Browser in each scenario, trying to mimic the use of a regular user. The RERAN tool was used to record and repeat each script five times, and the Trepn tool was used to monitor it. The results obtained allowed us to conclude which Browser was more efficient among the seven selected.

Keywords: Green Software, Green Computing, Android, Energy Efficiency, Web Browsers.

Resumo

Atualmente, existe um grande crescimento do consumo energetico do sector de IT, que apesar dos seus benefícios, está a deixar uma enorme pegada no que diz respeito ao consumo energetico. Com isto, o tópico do consumo energético e como melhorar começou ser um dos mais falados atualmente.

Diversos desenvolvimentos foram feitos neste âmbito de maneira a encontrar as soluções mais eficientes para os diversos problemas que os utilizadores e os programadores encontram. Mas isto está longe de ser uma tarefa fácil tanto para um como para o outro, sendo que ainda existe muita pouca informação disponível ou por vezes as soluções não vão de encontro às necessidades de cada um.

Com isto em mente, esta dissertação tem como objetivo verificar qual o browser é mais eficiente no ambiente Android, visto que não existe muita informação nesta área. Para isto, nós selecionamos sete browsers e fizemos quatro cenários de teste, de maneira a forçar os Browsers. De modo a conseguir testar, gravamos um script para cada Browser em cada cenário, tentando imitar a utilização de um utilizador normal. Foi usada a ferramenta RERAN para gravar e repetir cinco vezes cada script e para a sua monitorização é usado a ferramenta Trepn. Os resultados obtidos permitiram concluir um ranking de qual o Browser foi mais eficiente entre os sete selecionados.

Palavras-chave: Green Software, Green Computing, Android, Eficiência Energética, Web Browsers.

Contents

List of Figures

List of Tables

Listings

Acronyms

ANOVA	Analysis of Variance
API	Application Programming Interface
AVC	Android View Client
CLGB	Computer Language Benchmarks Game
CPU	Central Processing Unit
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
ICT	Information and Technology Communication
IT	Information Tecnology
JCF	Java Collection Framework
JPEG	Power Management Integrated Circuit
OS	Operating System
PMIC	Power Management Integrated Circuit
RAPL	Running Average Power Limit
RQ	Research Question
TDP	Thermal Design Power

Chapter 1

INTRODUCTION

Year after year, the number of smartphone users keeps growing, wherein 2021 reached 3,8 billion users (Statista, 2021). In any age group, its use has become a necessity to serve necessary functionalities daily, where its efficiency is one of the essential attributes for our daily, which allows us to perform tasks or connect with other people as we move. Being a fundamental device to our life, one of the criteria of users when buying a smartphone is the duration of its battery (Thorwart and O'Neill, 2017). The excessive consumption of the battery leads to most criticisms in the app stores (Fu et al., 2013), leading to the most concern in the development of applications by developers who rarely have the best solutions to optimize battery consumption.

Despite its importance, optimizing and analyzing energy consumption for mobile devices is by far an easy task for both users and developers. Developers are using different tools according to the specific needs (Hu et al., 2018, Cruz and Abreu, 2017, Di Nucci et al., 2017), obtaining sometimes results that are not systematized and/or not even in the specific context (Li et al., 2016). Monitoring an application's energy consumption thus leads to an extensive number of tests in different test scenarios and environments (Behrouz et al., 2015, Li et al., 2013), consuming a lot of time and also a large initial investment, for example, Android, which is a heterogeneous environment where there are thousands of combinations of producers, devices, operating systems, applications, etc.

Understanding the entire energy consumption system of applications is difficult for users. In addition to not having the necessary tools to monitor the consumption of smartphone applications, their knowledge about the hardware and its behavior is also limited, especially when each application behaves differently in different versions of the operating system.

With all this in mind, the topic of energy consumption and how to optimize it has become an increasing topic for both developers and users (Pinto and Castor, 2017), especially about the mobile device battery (Pang et al., 2016). It is also essential to consider that there are issues for both researchers and programmers in terms of understanding, measurement, and the way that the optimization of energy consumption

must be carried out. So, for instance, many programmers today are genuinely concerned with their software's energy efficiency. However, there is still a barrier between the solution they need and the problem they have, resulting in a lack of knowledge or awareness on this issue, resulting in high consumption. In this way, it can be said shortly that the main problems of efficient software development are then the lack of knowledge or lack of tools to accomplish this goal (Pinto and Castor, 2017).

1.1 Motivation

This thesis aims to provide more knowledge about mobile browsers and their effectiveness. There are other aspects to be explored and investigated, but it was decided to try mobile browsers because everybody now has and uses a browser daily, so there is a need to invest in this topic. It is intended to find the browsers which consume less energy for efficiency, but also without forgetting its performance finding the balance between performance/energy consumption. This thesis will be carried out an empiric study in a mobile environment, more specifically in an Android environment, because this is the most used operating system, with a percentage above 70% (Statcounter, 2020b). With this in mind, is pretended to answer this researches questions:

- **RQ1**: *Which mobile browser is the most energy-efficient for browsing Youtube?* According to Alexa[1], Youtube is the most popular video viewing site on the Internet. Understanding which browser is the most energy-efficient can help heavily reduce the energy consumed during this typical web browsing.
- **RQ2**: *Which mobile browser is the most energy-efficient for browsing Vimeo?* Vimeo has become an alternative to Youtube. Still, it has different characteristics but is getting more known, increasing his community.
- **RQ3**: *Which mobile browser is the most energy-efficient for searching facts or daily stuff in Google?* Nowadays, when a user doesn't know some answer or some truth, try to find it on Google, where there is an answer for almost everything.
- **RQ4**: *Which mobile browser is the most energy-efficient for browsing Facebook?* Social media has become something very common nowadays, and, yet again, knowing which mobile browser can help reduce energy consumption can give users more information for choosing a less energy footprint producing browser.
- **RQ5**: *Which mobile browser is the most energy-efficient overall?* While there may be browsers more suited to specific tasks and mobile applications, understanding overall in a more general sense which is the most energy-efficient one, can further help users choose their used browser.

[1] https://www.alexa.com/topsites

Some browsers should be chosen to answer these researches questions, following certain selection criteria, as it is not possible to test all current browsers since there is a wide range of offers. The selection criteria are the Top 5 of the most popular, that is, those with the highest number of downloads, some focus on the browser that mentions longer battery life characteristics compared to others, and another criterion relate to data privacy, where this issue has been widely discussed and generating a lot of controversies (BATISTA, 2020).

It is intended to try to imitate the behavior of a real user while using the browser after selecting browsers, covering various aspects, such as social networks, searching facts, viewing videos, and other aspects that may become important to the study. To imitate the user, a tool will be used to record all the actions performed and reproduce them whenever necessary. During its execution, all energy consumption will be monitored through a tool with high precision that will be detailed.

1.2 Contributions

The work (de Macedo et al., 2020) was developed in this scope of study but directed to the Desktop environment. In this work, an investigation is made about the energy consumption in Google Chrome and Mozilla Firefox browsers, tested in different test scenarios.

The test scenarios covered different areas, such as live streaming, watching videos, accessing social media, and loading Google Drive sheets. In these scenarios, various aspects were considered, and for their monitoring during execution, the Java-based RAPL(Running Average Power Limit)(David et al., 2010, Pandruvada, 2014) framework jRAPL was used. For recording these scenarios in scripts, the Selenium tool [2] was used, and they were executed ten times each. Additionally, the 20% highest and lowest values were removed to reduce outliers from such issues.

In the obtained results, it was possible to observe that Google Chrome is the most energy-efficient browser compared to different scenarios with Mozilla Firefox. Nevertheless, it was also observed that Google Chrome reached the highest energy consumption peaks, thus showing that Mozilla Firefox has a more consistent consumption, despite consuming more energy.

1.3 Document Structure

This thesis will be divided in this chapters:

- **Chapter 2 - State Of The Art:** Here, we present all the research and documentation necessary to ground better the choices made throughout this dissertation. It contains an introduction to Green Software as well as related work. After that, it is made a study more directed to the Mobile environment and Web Browsers, demonstrating several accomplishments already made in the area.

[2] https://www.selenium.dev/

- **Chapter 3 - Benchmark Architecture and Design:** This chapter details our design and methodology level.
- **Chapter 4 - Results:** Here is the chapter that presents all the results obtained, and they are analyzed in detail. Firstly graphs are shown and explained for better understanding, and then these are analyzed. In the end, some considerations to the project's development are made.
- **Chapter 5 - Conclusion and Future Work:** This final chapter contains the answers to the research questions, final considerations, and future directions that the study can improve.

Chapter 2

STATE OF THE ART

In this chapter, we will look into what has been done thus far on the Green Software. The first section will focus on research that has already been made in some related work of the main theme of this thesis, presenting some details. Afterward, we will look into research done in the mobile environment, specifying some existing tools and works. Finally, we will look into research regarding browsers and specify the paper that mainly contributes to this thesis.

2.1 Green Software

Nowadays, we have become aware of reducing our energy consumption and our carbon footprint. This is a real problem that affects at global level and turned out to be one of the most talked-about topics. If this problem is not started, one day, we may not have a healthy environment in which to live. This applies to different sectors, all of which contribute to the growth of this problem, and it is up to each citizen to help in this fight. To fight this, measures as been taken by the organizations, and the leading research and development initiatives have sizeable amounts of funding for projects seeking to accomplish better solutions.

Technological evolution has brought many benefits in this regard, contributing to virtual meetings, dematerialization of activities, improvement in logistics, intelligent transport systems, smart grids, more sustainable management of (smart) cities, etc. But the rapid growth of the information and technology communication (ICT) sector turned to be one of the key reasons why energy consumption has been growing (Gelenbe and Caseau, 2015). The use of information technology (IT), despite the benefits, has a negative impact on the environment where the amount of energy consumed by devices, data centers, services, processes, etc., and the reality is everyone has access to something related to it. Furthermore, as we live in a time of constant technological growth, children's access to these technologies is increasingly beginning at a younger age, when, in addition to their desire to own these devices, they are becoming integrated and

necessary for education (Druin, 2009).

With this in mind and the fact that the use of these devices is increasing rather than decreasing, the word Green Software comes up, referring to the practice based on minimizing energy consumption by optimizing software. This subject is a relatively recent topic (Lo and Qian, 2010). Still, there is already a vast amount of work to prove that this can have a significant effect on energy use when acquiring information about sustainable software alongside sustainable software engineering, being in this way an alternative for reducing global warming.

Every effort made in this way counts, and for this reason, many different aspects have been researched and explored; for example, in terms of software language engineering, they can provide powerful techniques to implement and evolve software languages. The main goal of this technique was to help programmers produce faster programs, and performance was intended for immediate execution. This idea had to change quickly, and many researchers started researching to make a positive impact. A wide range of programming languages differ in several aspects, such as their programming or execution paradigm grammar, which can affect the execution time or memory management. Knowing this, the energy efficiency of 27 well-known software languages from The Computer Language Benchmarks Game's [1] common software repository was analyzed and compared in (Pereira et al., 2017). In each test/language, programs were compiled and executed using the compilers, virtual machines, interpreters, and libraries, all to analyze the performance, considering three variables: execution time, memory consumption, and energy consumption. All results were analyzed considering the execution type and programming paradigm. The energy was collected by monitoring of the tool called RAPL (Running Average Power Limit)(David et al., 2010, Pandruvada, 2014), which allowed understanding how different languages or programming/execution paradigms would affect energy and was also able to show which software languages, execution styles, and paradigms were most energy-efficient across ten different benchmark problems. In the end, languages were ranked according to their results compared with other languages. It was also possible to show how to use these results to support software engineers in deciding which language to use. We can see the 27 languages ranking in Figure 2.1.

Choosing between data structures is another factor that can highly influence energy. The work presented in (Pereira et al., 2016) is a detailed study of energy consumption of the Sets, Lists, and Maps data structures included in Java Collection Framework (JCF). This study shows a quantification of the energy spent by each API method of each of those data structures, also using to measurement the tool called RAPL (Running Average Power Limit)(David et al., 2010, Pandruvada, 2014), more specifically, was used jRAPL (Liu et al., 2015) which is a framework for profiling Java programs using RAPL. It was possible to suggest a transition to reduce energy consumption based on their JCF data structures and energy quantification methods.

A similar approach in (Pinto et al., 2016) where Java collections from the JCF, defined as thread-safe, were tested in a multi-core execution environment to identify the trade-offs between performance and energy efficiency. This study was based on traversal, insertion, and removal operations. They were able

[1] CLGB page: https://benchmarksgame-team.pages.debian.net/benchmarksgame

Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

Figure 2.1: Normalized global results for Energy, Time, and Memory

to improve up to 17% energy savings by switching out collections, showing how such simple changes can considerably reduce energy consumption.

The energy efficiency of the data structure was not just a problem for the Java collections. For Haskell data collections (Lima et al., 2016), another similar study was carried out, where the energy efficiency of Haskell programs was analyzed from the perspective of strictness and concurrency. They also showed how significant impacts could be made by changing which data structures are used, saving up to 60% of energy in one of their settings. In the end, the authors concluded that "strict evaluation for most tasks as the default approach, especially when iterating over elements in Maps."

Several tools were created for various types of tests to obtain the results and carry out the investigations; as already mentioned above, RAPL (Running Average Power Limit)(David et al., 2010, Pandruvada, 2014) can be used in desktops running with Intel processors and reports accurate estimations for CPU and DRAM consumption. RAPL can collect about 100 estimates per second. A C-based interface can be used to obtain the values stated by RAPL. However, it has also been modified to support interfaces for other languages, such as Java (Liu et al., 2015) and Haskell (Lima et al., 2016).

Another example is PowerAPI, which is an API that allows energy consumption profiling at the operating system (OS) process level in real-time(Noureddine et al., 2015, Noureddine et al., 2013). Currently, this tool supports the calculation of CPU and network capacity, which is expressed by power modules. The available implementations provided for this tool are created for distributions of GNU/Linux, but they

are hardware-independent. Thermal Design Power (TDP), which is the overall amount of heat (which is produced by the CPU) that needs to be dissipated by the cooling system, is taken into account to calculate the energy consumption of the CPU. By comparing it to a power meter, the accuracy of calculating the power consumption of software applications with PowerAPI was calculated by Noureddine et al., 2015 showing that the estimated error margin ranged from 0.5% to 3%.

2.1.1 Mobile Energy Efficiency

Always-on connectivity, high-speed wireless networking, high-definition multimedia, and rich user interaction are becoming more common use cases. The development of battery technology has not been able to keep up with the increasing power requirements of the resource demand. The quantity of energy that can be stored in a battery is limited, and it is only increasing at a rate of 5% per year (Robinson, 2009). Larger batteries mean larger devices, which is not an appealing alternative. Furthermore, without active cooling, thermal constraints limit the power budget of small devices to around three watts (Neuvo, 2004).

The increasing number of potent computer devices requires energy-efficient hardware and software, which is being demanded from developers (Pinto et al., 2014). The improvements in energy efficiency can always be exchanged for other benefits such as device size, cost, and R&D efficiency. Indeed, in mobile phone designs, a considerable portion of the hardware technological benefits has been surrendered for programmability (Silven and Jyrkkä, 2007). For that main reason, it's being developed techniques and tools to answer all demands.

For example, the virtual keyboard application is one of the core applications included in every distribution of Android OS. This virtual keyboard is purely a software application, which only depends on the implementation, offering advanced features that overcome regular physical keyboards. Being a software application may affect overall energy consumption. It is possible to know which keyboard out of five widely used in Google Play is more energy efficient in Rua et al., 2020. To realize that, various scenarios were conducted using both human users and automated procedures to simulate real user interaction in order to compare the keyboards. The computerized procedures were performed from the Android View Client (AVC) framework [1], which is a Python-based tool evolved from monkeyrunner ("Monkeyrunner," 2020) that allows multiple devices connected to the workstation to be managed and interacted with. For each of the five keyboards, the automated procedure was executed 25 times in both test modes: default (with only the default characteristics of each keyboard enabled) and minimum (with all features disabled). For measurement of energy consumption, it was used the Trepn energy profiler ("Trepn Profiler," 2013). The results of this experiment show that there are indeed differences in the energy consumed by the keyboards chosen, and in most situations, swapping keyboards or enabling/disabling settings allows energy saving.

Another study (Linares-Vásquez et al., 2014) aimed at the quantity and qualitative investigating energy-greedy API calls and patterns identified from 55 free Android apps. For this, the 55 apps were exercised through scripts recorded with the Monkey Recorder tool ("Monkeyrunner," 2020), where it is tried to

[1] AVC:https://github.com/dtmilano/AndroidViewClient

imitate the actions of real users. These scripts were recorded so that they can be performed 30 times each and are measured using the Monsoon power monitor ("Monsoon power unit," 2020), aligned such measurements with execution traces, and finally identified and traced onto source code the interesting API calls and patterns. With the result, it was possible to define a few usage scenarios for API calls where it was possible to prevent abnormal consumption.

Commits, problems, and pull requests were analyzed in (Cruz and Abreu, 2019) from 1021 Android apps and 756 iOS apps to identify design practices to improve mobile app energy efficiency. As a result, this work provides a catalog of 22 design patterns based on 1563 mobile app changes related to energy. This catalog will help designers and developers of mobile apps make informed decisions, regardless of the target platform, while designing (energy efficient) apps.

2.1.1.1 Tools and Techniques

Despite the previous research efforts, there is a lack of tools that quickly and efficiently measure the energy consumption of mobile applications. Existing tools have three main categories: (i) hardware-based, (ii) model-based and (iii) software-based approaches. Each category has its own advantages and disadvantages.

Hardware-based tools can delineate the exact energy profile of a mobile app, but they require hardware components that are expensive and difficult to set up. Monsoon power unit ("Monsoon power unit," 2020) is a hardware-based tool directly linked to a typical lithium-ion battery-powered device's battery power connectors (e.g. smartphones). Since it acts as a battery replacement, supplying the device with power, it knows precisely the amount required at any given time. It can capture up to 5000 measurements per second depending on the model.

Model-based techniques aim to explain mathematical functions that can estimate the energy consumption of mobile apps on a given hardware unit. However, such methods require careful calibration of the parameters to calculate power consumption correctly. PETrA (Di Nucci et al., 2017) offers mechanisms for testing Android applications automatically and reporting the energy consumed by each method (on average). Their built-in energy profiler uses these power profiles as an energy model. The key drawback is the fact that manufacturers do not always have power profiles, and there is no awareness of the accuracy of the current ones.

Finally, software-based approaches estimate a mobile application's power profile solely by depending on a device's machine capabilities, such as the CPU frequency. Thus, since they rely on measurements obtained by physical hardware components (e.g., CPU, battery, etc.), but without any specialized hardware tools, these tools can be considered hardware-assisted. By nature, they are easier to use and cheaper than pure hardware-based solutions, but they are believed to be less precise (Hao et al., 2013). Trepn energy profiler ("Trepn Profiler," 2013) is suited for Android devices that use Snapdragon CPUs, and its estimations are obtained through a PMIC (Power Management Integrated Circuit). Hence it considers the whole consumption of the device. It collects a new estimation every 100 milliseconds. The accuracy of this

tool is already validate in several works (Rua et al., 2020, Linares-Vásquez et al., 2014, Di Nucci et al., 2017).

2.1.2 Web Browsers

With the advancement of time, technology has become more integrated into our lives. With that, the use of the internet has become a tool in our everyday habits that we use at any moment, either to obtain some information or for simple leisure. The truth is, browsers have become almost unattachable from an internet experience, and where we depend upon them for everything web-related. We use web browsers for social life, personal research, watching videos, playing games, working, etc. Herewith, we can say that web browsers are one of the most important and used internet tools (Borgolte and Feamster, 2020; Nejati and Balasubramanian, 2016). Knowing what was said above regarding the battery life of smartphones and the work that already exists to improve energy efficiency in different aspects, it is also necessary to research and explore the topic of web browsers. Besides, there are already different types of work, and they have already researched into various aspects, there are still answers to give.

In 1993, Mosaic (STAFF, 2021), the first graphical user-friendly Web browser, was released, making the World Wide Web available to everyone and contributing to the beginning of an information boom that continues to this day. Marc Andreesen founded Netscape a year later, and Navigator became the company's core product. The following year, Microsoft entered the battle by introducing Internet Explorer, its Web browser. These incidents sparked the "browser wars,"as they are now known.

There's a great browser competition because there's a great variety in that offer where every user can choose which one best suits their needs and demands. As a result of this competition, researchers have started to test in order to give factual information to the community. Therefore, investigations begun in this direction, like in Thiagarajan et al., 2012, where infrastructure is presented for measuring the precise energy consumption used by a mobile browser to load and render websites as well as the energy needed to render individual web elements, such as cascade style sheets (CSS), Javascript, images, and plug-in objects. This was performed in different types of areas like finance, e-commerce, email, blogging, news, and social networking sites, at popular websites such as Gmail, Amazon, and many others. With the data collected, it was possible to demonstrate that downloading and parsing cascade style sheets and Javascript consumes a significant portion of the total energy required to render a page for popular sites, and rendering JPEG images are considerably cheaper than the other formats. It was also possible to provide concrete recommendations on how to design web pages without affecting the user experience in order to reduce the amount of energy needed to render the page.

Other research was carried out on current browsers to inform users about which is the most energetically effective, taking into account the performance of the device because the time the user spends waiting for a response by the browser on the smartphone increases energy consumption, knowing that energy consumption is defined by power consumption over time so the extra waiting time might affect the energy consumption. It is compared three commonly used web browsers, Google Chrome, Mozilla Firefox,

and Opera, in terms of page loading, comparing 150 random websites considering two different Internet frequencies, checking whether there is a better page loading browser (Hsu et al., 2017). This approach permitted users to have an insight into the energy consumption and performance of these web browsers but only when loading the websites and not exploited at the level of regular use by a user.

A recent approach (Bouaffar et al., 2021) developed a simple command-line tool, *PowDroid*, to measure the energy consumed in Joules by any applications run without requiring access to source code like Energy Profiler [2] integrated within Android Studio IDE does. It can be used to perform benchmarks and analyze which components are draining more battery. Three test scenarios were made to prove its efficiency: web browser, camera, and weather applications. The web browsers Mozilla Firefox, Google Chrome, Microsoft Edge, Opera, Samsung Browser, and Brave, were tested and the test was consisted of searching for a keyword and opening three websites. Each test took 4 minutes, with the battery level at 50% at the start (recharging it to 50% before starting the next experiment), screen brightness at 50%, and audio at 50%. Their experiments found that Brave is the most energy-efficient, while Firefox was the worst with a 33,8% increase in energy consumption. They allegate this happens due to the default built-in AdBlock in Brave. We can see the energy consumption of each browser in:

Application	Energy (Joule)
Mozilla Firefox	427,78
Google Chrome	404,27
Microsoft Edge	386,18
Opera	368,33
Samsung Browser	345,93
Brave	319,66

Figure 2.2: Energy consumption - Web browsers

Although our tests that will be described below are not the same as those applied in this work, we can say that Opera is the browser that consumes the most and the Brave is not in all cases the one that consumes the least, but this will be demonstrated and explained in the sections that follow.

The main contribution and motivation to this study is the research conducted on the desktop between Google Chrome and Mozilla Firefox (de Macedo et al., 2020), where energy efficiency is analyzed using Intel RAPL (Running Average Power Limit)(David et al., 2010, Pandruvada, 2014) to monitor energy consumption during the execution of each script under the same specific test conditions. These scripts were generated with Selenium [3], where multiple actions were performed to force the browsers, such as watching videos, scrolling down, etc., imitating a user, depending on the website. This research was a test-bed and preliminary study to serve with advancements and inspiration to explore this further, observing the results obtained.

[2] https://developer.android.com/studio/profile/energy-profiler
[3] https://www.selenium.dev/

Besides focusing on energy consumption and more on performance, there is a different point of concern regarding the security of browsers as they process the user's private data for certain operations. In addition, social life is often done through browsers, which can lead to improper access to information relating to the personal life of each user or even a bank operation. Thus, more intensive work has been done to improve conditions and prevent these cases from happening. Performance analysis of the algorithms used to avoid computer threats was performed in (Ramesh and Umarani, 2012). For this, a Web programming language was created, which would be compared against five Web browsers in terms of their ability to handle the encryption of the programming language's script with the browsers. After that, a test simulation is run to determine the optimal encryption algorithm vs Web browser. It was possible to determine that different algorithms perform differently in different Web browsers and which algorithm performs best and is most compatible with which browser.

Chapter 3

BENCHMARK ARCHITECTURE AND DESIGN

This Chapter will present the methodology used to obtain the results needed to answer our RQ.

3.1 Benchmark Design

The growing use of smartphones leads to the increasing use of mobile apps, including mobile browsers. When choosing their favored browser, users consider many factors, such as speed, resource management, themes, plugin compatibility, etc. Due to the lack of information, users do not currently consider the energy efficiency of the browsers. While many might assume speed directly equates to energy efficiency, several research works are showing this is, in fact, not always direct Abdulsalam et al., 2015; Couto et al., 2014; Pereira et al., 2020.

Nowadays, it exists more than 100 mobile browsers, and it would be pretty impossible to have measurements of them all for that reason. Each of them has its different specifications and suits differently to every user. So, to answer RQ, some browsers need to be chosen, following specific selection criteria.

The selection criteria, first of all, are the Top 5 of the most popular and consequently the most downloaded, Figure 3.1. According to the Statcounter, 2020a, Safari is the second browser most used worldwide, but it won't be tested due to our environment is in Android, and it doesn't exist for it, then it will be Google Chrome, Samsung Internet Browser, UC Browser, Opera Mini, and Mozilla Firefox.

Browsers	Number of Downloads
Chrome	5mM+
Samsung browser	1mM+
UC Browser	500M+
Opera Mini	500M+
Mozilla	100M+

Table 3.1: Number of Downloads from browsers.

Another criterion was a specific specification of Brave that affirms to be 2x to 4x speed boost on Android, saving data and battery. Due to that, users could count up to 2.5 extra hours of browsing time per battery charge. The last browser chosen was DuckDuckgo because privacy has been widely discussed and generating a lot of controversies BATISTA, 2020.

It is intended to try to imitate the behavior of a real user while using the browser after selecting browsers, covering various aspects, such as social networks, viewing videos, and other elements that may become important to the study. It will be used a tool called Reran [1] Gomez et al., 2013 for the creation of scripts, which consists of recreating the specific action on the smartphone trying to imitate real user actions and recording them to be possible to replicate the desired times. It was made one script for each browser, but the steps in each script are equal in all, and the execution times have insignificant differences between them to ensure each browser is tested in equal terms.

3.1.1 Scripts

In order to answer RQ1, it was made three scripts were each one consisted of browsing *youtube.com*, then searching one specific video by typing in the search bar and playing it in 720p. The videos chosen were Despacito from Luis Fonsi and the short cartoon Masha and the Bear-Recipe for Disaster from Get Movies. These videos have been selected from the list of most viewed videos on Youtube Wikipedia, 2021. The other video was random and is another music, Paradise from Coldplay. This test was not conducted on Opera Mini and DuckDuck browsers because they did not allow the video to be placed in fullscreen, leaving two black bars on the screen, reducing the video size. Since this did not happen for other browsers, it would influence the results.

The strategy to answer RQ2 was the same as Youtube, but only the video Paradise from Coldplay was the same. The other videos were *A Mind Sang* and *Shadows In The Sky*, and they were chosen randomly. Contrary to what happened with Youtube, where two browsers did not allow putting the video in fullscreen, on Vimeo, it turns out that only the Firefox and UC browsers allow putting it in fullscreen and therefore were not tested.

The strategy used to answer RQ3 consisted of three scripts. They consist of browsing *google.com* at the start, then two of them are searching true's, and one is searching one recipe. One of the searches is searching *"who was the first man in the moon"*, then click on the Nasa website [2] staying in this website 30 seconds and after that goes back and goes to Apolo 11 on Wikipedia [3] waiting for 60 seconds, 30 seconds on the main text, and scrolling down through details. The other search is *world war two* and clicks on Wikipedia site [4]. There will be 30 seconds for the first text, scrolling down to the following text for 50 seconds and reaching the text is 25 seconds for the first paragraph and 15 seconds for the second. The last script searches "recipes with chicken"and clicks on the delish website [5]. Then scrolls down to the

[1] https://github.com/RRua/RERAN
[2] https://www.nasa.gov/audience/forstudents/k-4/stories/first-person-on-moon.html
[3] https://en.Wikipedia.org/wiki/Apollo_11
[4] https://en.Wikipedia.org/wiki/World_War_II
[5] https://www.delish.com/cooking/recipe-ideas/g2972/chicken-weeknight-dinners/

second's recipe. Then consists of reading ingredients and the five steps for the recipe, where each has 20 seconds of a wait but lasts only 10 seconds. After this, it scrolls up to a video of the recipe and stay's till the end. This test was not carried out in the DuckDuck browser due to its privacy policy, which always erases the browse history and cookies. For that reason, it did not allow the development of a script with the RERAN tool because it was impossible to guarantee that the clicks were in the same place throughout the different executions.

To answer RQ4, the script starts to browse *facebook.com* where an account is already logged in to be more accessible. Then is searched *Jornal de Noticías*[6] by clicking on history search, which is one news page in Portugal. This page was selected to do the same test in each script to every browser. After that, goes to the photos section, where 15 photos of the front page of a newspaper are loaded one by one. It goes back to the landing page of *Jornal de Noticias* and scrolls down through the feed where news is posted. After that, it goes to account saves, where is a video from the page and it takes 60 seconds to finish. This test wasn't conducted in Opera Mini and DuckDuck for the same reason on Youtube.

To sum up, all these different aspects of the test will answer RQ5 by analyzing the results of every result obtained.

3.2 Execution

To obtain the energy consumption of each test, it was used the tool called Trepn energy profiler "Trepn Profiler," 2013 which is suited for Android devices that use Snapdragon CPUs. Its estimations are obtained through a PMIC (Power Management Integrated Circuit). They can be used to profile hardware usage, resources usage, and power consumption of both the system or standalone Android applications. Hence it considers the whole consumption of the device, and it collects a new estimation every 100 milliseconds. The accuracy of this tool is already validate in several works Rua et al., 2020, Linares-Vásquez et al., 2014, Di Nucci et al., 2017. In order to obtain structured data and do better analysis, the output of Trepn is in CSV files.

Different precautions were taken to have consistent data and reduce effects from cold starts, warm-ups, and cache effects. Every script was executed five times, and the script from Facebook was performed in the middle of the night to ensure there wasn't added news and the feed was the same for every browser.

All data collected in CSV files are worked on to be more accessible. The data is in microwatts is changed to Joules and milliseconds to seconds. For this, a script made in python is used, and the sum of the total energy consumption is made by multiplying the energy consumption obtained at the moment for the time elapsed in this interval.

[6] https://www.facebook.com/jornalnoticias

Listing 3.1: Formula used to calculate total consumption.

```
...
 while(i<len(df)-8):
      p=float(df['EnergyConsumption'].loc[i+1])*0.000001
      a=float(df['Time'].loc[i+1])*0.001-float(df['App'].loc[i])*0.001
      total.append((p*a))
      i=i+1

   while(i-3<len(df)):
      total.append(np.nan)
      i=i+1

   df[”TotalN”] = total
  ...
}
...
```

All measurements were performed in the same LG Nexus 5 devices in version 6.0.1. The brightness level was at a minimum, and all applications were turned off, only Trepn open in the background.

Chapter 4

ANALYSIS AND DISCUSSION

This chapter will present and analyze the results of a benchmark in which all scripts were performed using the RERAN tool and the energy consumption result obtained using Trepn.

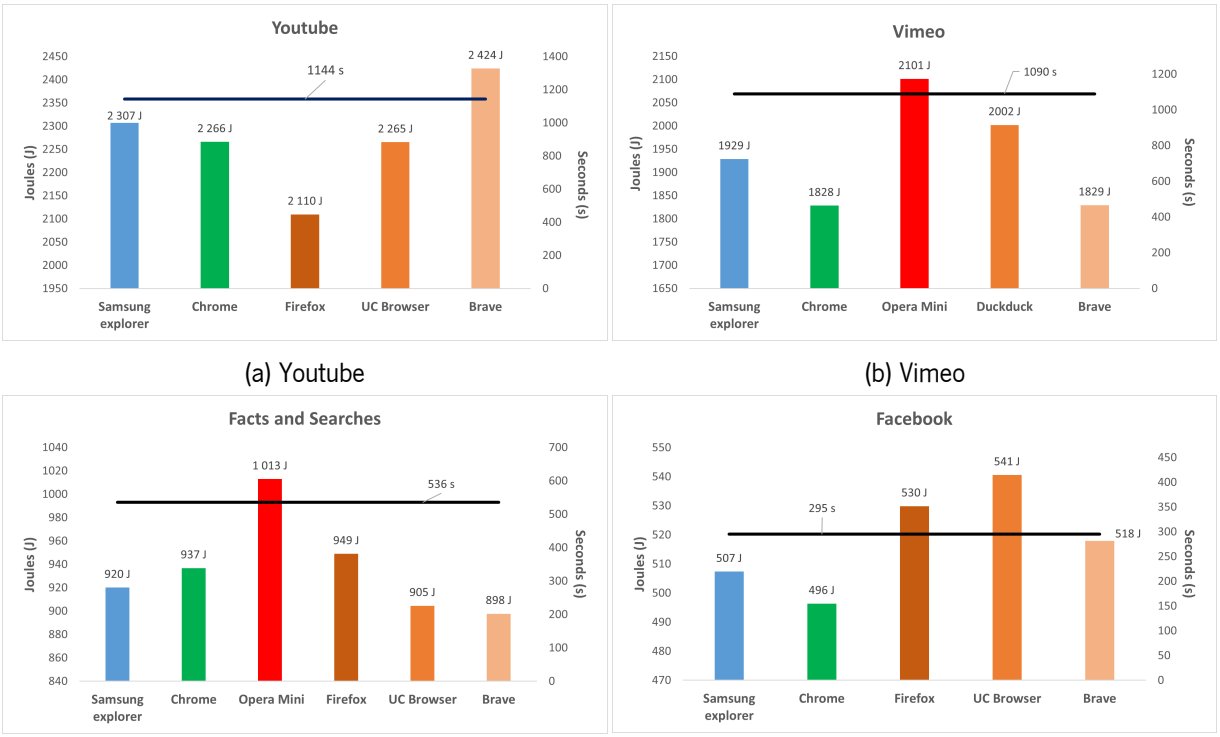
4.1 Benchmark results

This section will present and analyze the results obtained by each browser in detail. The overall energy consumption of each script executed by RERAN and monitored by Trepn will be analyzed and concluded with the help of two statistical techniques by comparing all of them according to the specified group of tests.

4.1.1 Results

To better understand the results obtained by our tests, we display some graphical visualizations of the energy consumed by browsers.

Figure 4.1.a) to 4.1.d) shows the total energy consumption in Joules of each browser in each scenario. Each browser has one bar representing the three tests performed and executed five times each. For example, we have the energy consumption from all videos, and the five results of each video are summed in the blue bar for Samsung Explorer. This column is associated with the y-axis on the left side. We can also observe a black bar, which indicates the time consumed by the script's execution in each test scenario. This bar is associated with the y-axis on the right side.



(a) Youtube

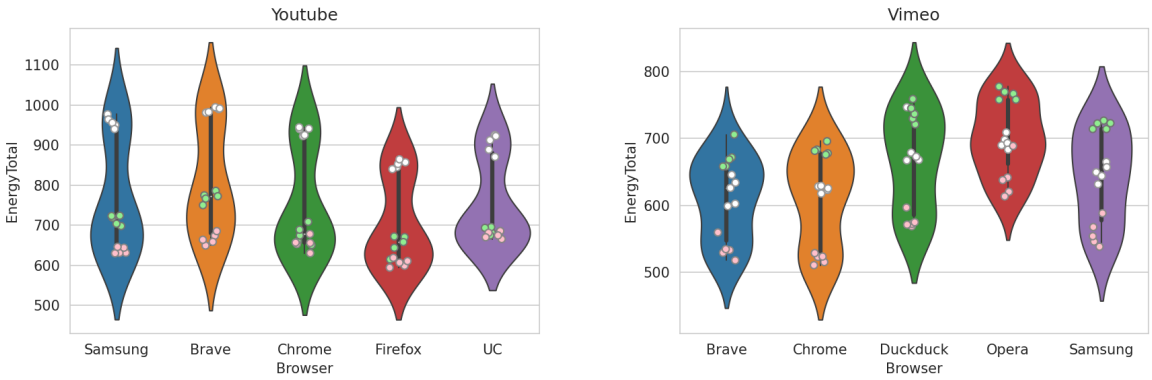
(b) Vimeo

(c) Facts and Searches

(d) Facebook

Figure 4.1: Total energy consumption in Joules considering the 3 test - Column

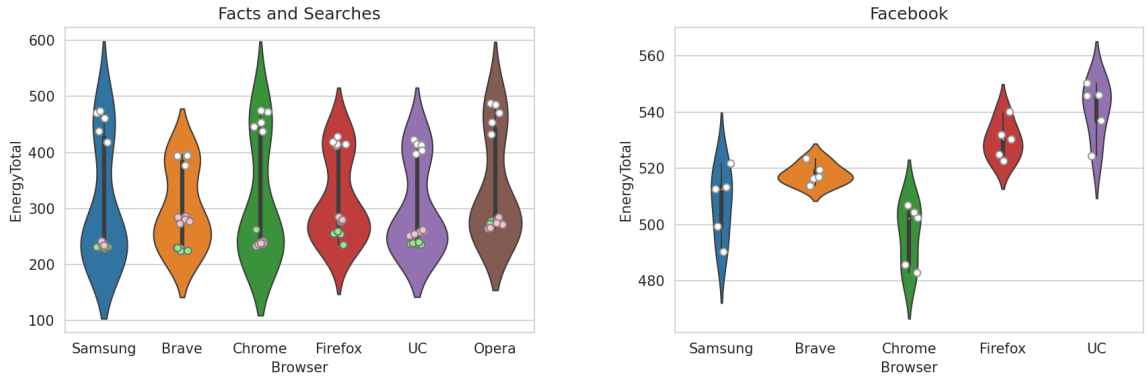
Figures 4.2.a) to 4.3.b) are presented violin plots for each test group and each Browser. There are also shown all five results of the executions of each case of study in dots separated by colors. With these plots, it´s possible to display the data collected for energy consumption density. Each plot allows understanding which Browser is consistent or inconsistent through the different scenarios. The value of dots corresponds to the y-axis(Total Energy(Joules)). The graph allows us to see that the results of the tests performed are very consistent because the dots of the same color are quite close to each other, and therefore, despite having been done five times each test, these are proven to be enough to prove due to their consistency.



(a) Youtube

(b) Vimeo

Figure 4.2: Total energy consumption (Joules) - Violin Plots



(a) Facts and Searches

(b) Facebook

Figure 4.3: Total energy consumption (Joules) - Violin Plots

4.2 Discussion

Our main focus in this work is to be able to answer RQ 5, and for that, we need to validate all the remaining RQs. Looking at Figure 4.1.a), we can see a discrepancy between the results obtained. We can see that all had an execution time of 1144 seconds, so the duration of use of the smartphone is no longer a factor that influenced the results obtained, as said before. Firefox was clearly the browser that consumed the least energy, 2110 Joules, while Brave consumed 2424Joules. The difference between Chrome and UC browser is almost nil, 2266 and 2265 Joules respectively, with Samsung Explorer consuming slightly more than these, 2307 Joules. Even though Chrome and Samsung sometimes show an ad, where it is passed after five seconds of the ad but is still loaded and can negatively influence consumption, they managed to get better results than Brave that has a built-in AdBlock, which prevents any kind of ad, thus helping to get less energy consumption and more battery life. In the tests performed for Firefox, no ads appeared when loading the videos, which may have influenced the lower consumption. The UC browser, on the other hand, although no ads were shown, has other default add-ons being loaded during video loading, which consumes more power than Firefox.

The results obtained in the Vimeo test group, Figure 4.1.b), show that for the duration of the 1090 second test execution, there is a significant difference between the browser that consumed the most energy and the one that consumed the least. Opera Mini consumed in total 2101 Joules while Chrome consumed only 1828 Joules. Opera Mini's consumption is due to the browser's goal, which is to ensure high performance but save on the space it takes up, and it also has AdBlock-like Brave built-in. To get this increased performance, the browser consumes more energy. Brave managed to show this time, unlike Youtube, a very low consumption almost equaling Chrome, 1829 Joules, not being the one that consumed less energy for a minimal difference. DuckDuck, on the other hand, is another one that consumed the most energy, being a little behind Opera Mini at 2002 Joules. This browser, which is known for its privacy policies, is one of the factors why it has a higher power consumption than the others. The processes that run in the background to ensure the safety of the user's navigation end up leading to higher consumption. Samsung Explorer consumed 100 Joules more than Chrome, that is 1929 Joules, showing as well as

Youtube consistency in its consumption, in which it is not one of the least consuming but also not one of the most consuming. All tests in this group had no ads loading, which allowed better comparison of results.

In Figure 4.1.c), we can see that the browser that consumed the most energy was Opera Mini, consuming 1013 Joules for 536 seconds. Brave is again the least consuming group browser with 898 Joules, thus differing from the UC browser by 7 Joules, 905 Joules. The remaining browsers have a similar consumption, with Samsung Explorer having 920 Joules, Chrome 937 Joules, and Firefox 949 Joules. Since this group is about loading pages with information and videos specified above, these pages may contain ads and therefore may influence the energy consumption results. Although Opera Mini has a built-in Adblock, this was the browser that consumed the most, which compared to Brave, which is another one with Adblock, consumed the least. In this respect, Brave makes its claim of minimizing energy consumption true. Another aspect that we can see is the loading of web pages. As we have seen in other work already done, they often keep their focus on performance, i.e., opening the web page in the shortest time possible, thus forgetting to reconcile energy consumption. One of the cases that we can observe is the case of Firefox, which in terms of loading videos, we saw in the Youtube group that it is the most efficient browser. Still, here in terms of loading pages, it has already become the one that consumed more, other than Opera Mini.

Finally, in Figure 4.1.d) shows the results for the Facebook group where it can be seen that Chrome is the browser that consumed the least energy, with an energy consumption of 496 Joules in 295 seconds. UC Browser has a consumption of 541 Joules, being the browser with the highest consumption in the group. Together with UC Browser is Firefox which has little difference regarding consumption, with 530 Joules, making it almost the browser with the highest consumption. Brave and Samsung Explorer have similar consumption, consuming 518 and 507 Joules, respectively. Even though Brave has an ad blocker that sometimes appears on the Facebook web page, it can't get better results than Chrome and Samsung Explorer. Once again, there is a test regarding page loading in this group, and Firefox gets high results compared to the other browsers. Besides page loading, there is the loading of images. UC Browser and Brave show a higher consumption derived from that, considering that the different test strands specified in Design have already been dealt with in the other test groups.

4.2.1 Statistical techniques

Software engineering is becoming a scientific field rather than relying on marketing in the last few decades. This is happening because if we want to maintain control over the program, we must evaluate new approaches, techniques, languages, and tools before implementing them and this help to build software engineering as a scientific field. Science is discussed in Dr Simon Singh's "Fermat's Last Theorem"[160], and the following summarises the key points presented during the conversation. Physical phenomena are studied in science by presenting hypotheses. The phenomena are observed, and if the observations support the hypothesis, the hypothesis becomes evidence. So, two different statistical techniques will be

used to test the hypothesis.

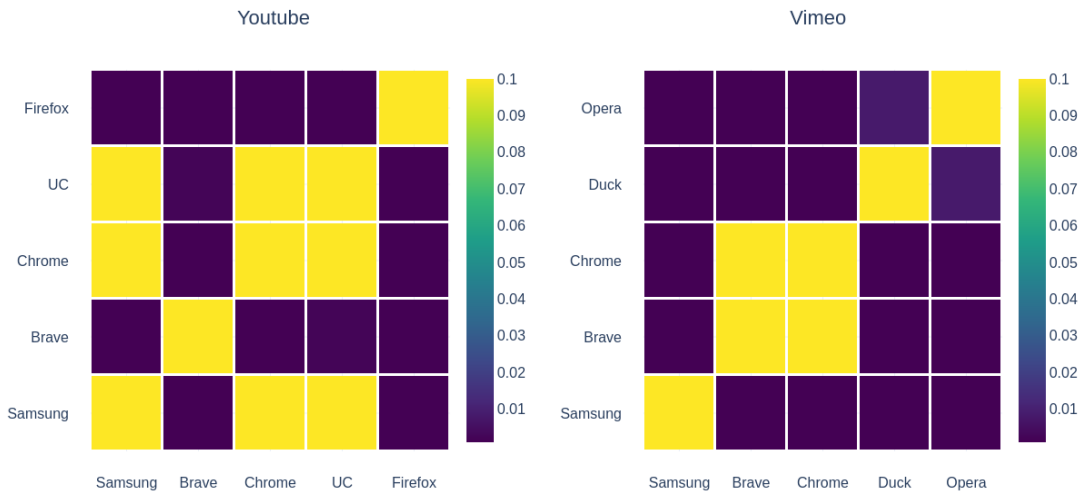
Hypothesis testing aims to determine if a sample from a statistical distribution can be used to reject a specific null hypothesis, H0. The null hypothesis, in other words, defines some properties of the distribution from which the sample was selected, and the experimenter intends to reject these properties with a given significance. The dependence of the distribution on a single parameter is a common example. In order to set up H0, you must first formulate the distribution and assign a value to the parameter that will be tested.

In order to validate whether the results obtained overall have statistically significant relevance or not, we performed a statistical analysis on the collected data. As such, we tested the following hypotheses:

$$H_0 : P(A > B) = 0.5$$
$$H_1 : P(A > B) \neq 0.5$$

This represents that when it is drawn randomly from A and B, the probability of being A is greater than the probability of being B is 50% in the case of our null hypothesis. It is different from 50% in the alternative hypothesis.

For all the data collected, pairs of browsers were made in the appropriate test groups in order to understand if there is significant relevance between the pair difference, i.e., in the Youtube group, we will get ten pairs (Samsung, Brave), (Samsung, Chrome), (Samsung, UC), (Samsung, Firefox), (Brave, Chrome), (Brave, UC), (Brave, Firefox), (Chrome, UC), (Chrome, Firefox), (UC, Firefox). This same logic applies to the Vimeo and Fact and Searches group, with Facts and Searches having 15 pairs for six browsers in the test group. We considered the samples independent, non-normal distributed and ran the Wilcoxon signed-rank test with a two-tail P value with α =0.05. With this in mind, the following heatmaps are presented with the appropriate comparison results for each pair:



(a) Youtube (b) Vimeo

Figure 4.4: HeatMap - Wilcoxon signed-rank

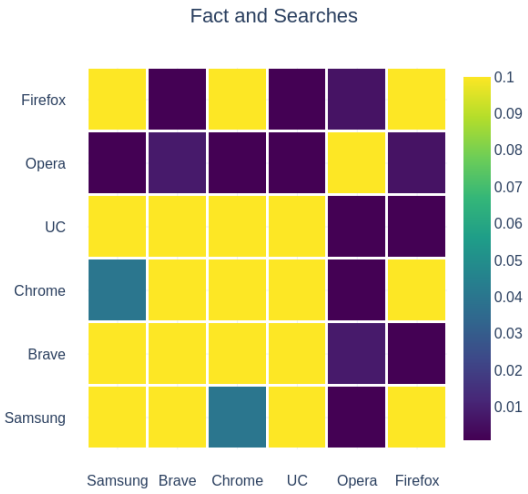


Figure 4.5: HeatMap - Wilcoxon signed-rank - Facts and Searches

In these HeatMaps it can be seen which pairs of Browsers have a significant difference. Looking at the scale to the right of each HeatMap, the color of the p-values < 0.05 is indicated, which is the shades of blue towards purple. In yellow we have the pairs that have no significant difference. In HeatMap Figure 4.4.a) you can see that all pairs have a p-value < 0.01, except the pairs (Samsung, Chrome), (Samsung, UC) and (Chrome, UC). In Figure 4.4.b), which refers to Vimeo, the pair (Brave, Chrome) has p-value > 0.05 and no significant difference. All other pairs show a high significant difference. In Figure 4.5 the pairs of Opera, (Firefox, Brave) and (Firefox, UC) in the Fact and Searches group, have p-values < 0.01, while the others have p-value > 0.05 and so the difference between them is not significant. This group, the differences are not very significant, only when it is compared with Opera, due to the difference in energy consumption.

To calculate a nonparametric effect size, Field (Field, 2009) suggests using Rosenthal's formula (Rosenthal, 1991; Rosenthal et al., 1994) to compute a correlation, and compare the correlation values against Cohen's (Cohen, 1988) suggested thresholds of 0.1, 0.3, and 0.5 for small, medium, and large magnitudes respectively. All pairs that showed a significant difference, with a p-value < 0.05, the effect size was calculated and all results obtained are greater than 0.5. We can then conclude that all pairs with a significant difference have a large effect.

In the case of the Facebook group, the data follows a normal distribution and so it was necessary to use another statistical method, Analysis of Variance (ANOVA).

The ANOVA statistical analysis method compares the means of more than 2 test groups, that is, in this case, it will serve to compare the means of the various pairs within the Facebook group. It is thus possible to analyze whether there is a significant variance between them. ANOVA One-Way was the type used for this case because it has only one independent variable. The **ANOVA hypotheses** are:

- **Null hypothesis:** Groups means are equal (no variation in means of groups)
- **Alternative hypothesis:** At least, one group mean is different from other groups

After calculating the p-value by using the ANOVA analysis, a p-value of 0.00003 was obtained. The p-value obtained from the ANOVA analysis is significant ($p < 0.05$), and therefore, we conclude that there are significant differences among treatments. To know the pairs of significantly different treatments, we will perform multiple pairwise comparison (post hoc comparison) analyses for all unplanned comparisons using Tukey's honestly significantly differenced (HSD) test. With this, the p-values of each pair were calculated, and these are represented in Figure 4.6.

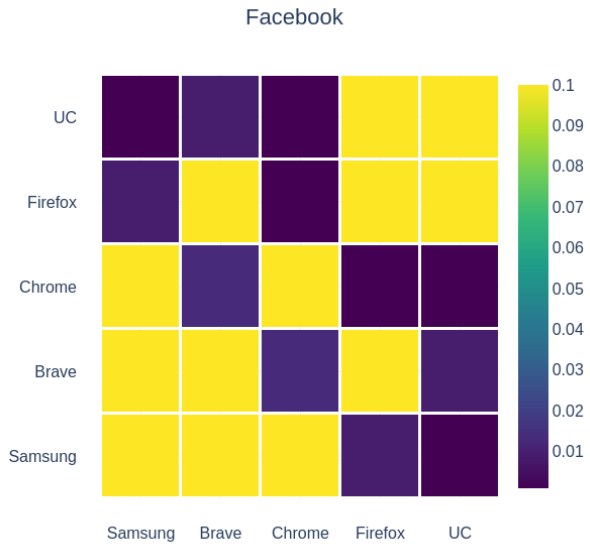


Figure 4.6: HeatMap - Analysis of Variance (ANOVA) - Facebook

As we can see, the p-value calculated through ANOVA analysis, 0.0003, is due to most of the compared pairs having a p-value < 0.05, except the pairs (Samsung, Brave), (Samsung, Chrome), (Brave, Firefox) and (Firefox, UC). To understand the effect size of this analysis, Cohen's F (Cohen, 2013) formula was used, where the basic rules of thumb for Cohen's F are that 0.10 indicates a small effect, 0.25 indicates a medium effect, and 0.40 indicates a large effect. The result of the formula was 0.77, which means that all pairs with a significant difference have a large effect size.

4.3 Threats to Validity

This thesis aimed to measure the energy consumption of browsers in Android environment and compare the results to know which browser is more efficient. We present in this section some threats to the validity of our study, separated into four categories (Campbell, 1979).

Conclusion Validity. In this category are the threats which may influence the capacity to draw correct conclusions. We can conclude that Chrome is one of the most energy-efficient browsers, along with Brave. Although on Youtube, Brave shows high results, it proves to be a good alternative. Opera Mini spent more energy in all the groups it was tested. Our results are consistent, but as it was impossible to compare the

selected browsers in all groups, there may be cases where a browser not tested in one group shows better results.

Internal Validity. This category concerns itself with what factors may interfere with the results of our study. While testing, we encountered an initial problem when executing the scripts. Not all browsers have the same response time in terms of performance, and therefore, when making the scripts they had different times for the same steps. This could influence the results by some having more execution time, and so we made individual scripts in which the execution time is the same for all. Another problem encountered was that it was impossible to compare all the browsers in all the groups. This was not possible due to the specific characteristics of the browsers, as mentioned above.

Construct Validity. Here are threats that involve generalizing the results to the concept or theory behind the experiment. Seven browsers were analyzed, although it was never possible in the four existing test groups to test them all together. A test script was made for each browser in each test group, giving a total of 52 scripts, each of which was run 5 times. There were thus 260 runs. We use a viable tool, Trepn, for monitoring the execution of scripts to get reliable results. The tool used was RERAN which was reused from Google, making it a valuable and reliable tool for script execution and recording.

External Validity. This category is concerned with the generalization of the results to industrial practice. The obtained results were the best performing ones when we set up this thesis. Measurements in different systems, might produce slightly different resulting values if replicated. We believe these results can be further generalized, and other researchers can replicate our methodology for future work.

Chapter 5

CONCLUSION AND FUTURE WORK

This thesis presents a study regarding energy consumption in an Android environment to determine which Browser is the most energy-efficient. For this, we selected seven Browsers from the 1000+ that exist to restrict the field of study. A benchmark architecture was developed to test and obtain the necessary results to make comparisons between them. Four test groups were created, Youtube, Vimeo, Facts and Searches, and Facebook, to try to cover different areas and thus mimic the use of a real user. The tool used for recording and repeating the scripts was RERAN, developed by Google. To monitor the execution of the scripts, we used Trepn, which has proven to be very viable and is also very practical and accessible.

In table 5.1, we can see the summary of Browsers' rankings in each test scenario.

	Chrome	Samsung	Brave	Firefox	UC Browser	Opera Mini	DuckDuck
Youtube	3	4	5	1	2		
Vimeo	1	3	2			5	4
Facts and Searches	4	3	1	5	2	6	
Facebook	1	2	3	4	5		
Mean	2.25	3	2.75	3.3333	3	5.5	4

Table 5.1: Classification of each Browser in each Scenario

After analyzing these results, it is possible to answer the five research questions presented in Section 1.1:

- **RQ1**: *Which mobile browser is the most energy-efficient for browsing Youtube?* As we can see, in the Youtube test group, the Firefox Browser was the most energy-efficient. On YouTube, ads often appear in some browsers, such as Chrome and Samsung, influencing their consumption. Nevertheless, they had a little different consumption from UC Browser, which doesn't have ads in its executions. Even so, Firefox presents a result of reduced energy consumption compared with

the others. Brave ends up in the last position, even though it has Ad-Block embedded, which may have affected it.

- **RQ2**: *Which mobile browser is the most energy-efficient for browsing Vimeo?* On this question, it is possible to affirm that Chrome's most energy-efficient browser. As Vimeo never shows ads, Chrome, unlike Youtube, shows the best consumption results. Brave and Samsung Explorer are also viable options since they differ little from Chrome. Opera Mini has a high consumption, occupying the last place in the ranking.

- **RQ3**: *Which mobile browser is the most energy-efficient for searching facts or daily stuff in Google?* In this case, we can say that Brave is the most energy-efficient browser. In this test group, Brave does justice to one of the characteristics that it claims to have, consuming less energy than the other browsers. Even though it's a test scenario where several things like video, loading pages, etc., Brave seems to have a development prepared for diversified use. UC Browser is also a viable option since it presents little difference from Brave. Once again, Opera Mini has the highest consumption, not a viable option.

- **RQ4**: *Which mobile browser is the most energy-efficient for browsing Facebook?* In this case, it is possible to state that the most energy-efficient browser is once again Chrome. In this test scenario, there is again the loading of a video and, in addition, the loading of images and the news feed. Chrome presents a better capacity than the others to manage its energy consumption, being the most viable option. Samsung Explorer is also a viable option.

- **RQ5**: *Which mobile browser is the most energy-efficient overall?* To answer this question, one needs to make an overall assessment of the four test scenarios. In Table 5.1, each Browser's rankings were averaged, considering which test scenarios they entered, to understand the rankings in total better. Although the Browsers are not tested in all the test scenarios, it is possible to state that Chrome presents the best results throughout the four test scenarios. Right after we have Brave, that doesn't get a first place by the results obtained on Youtube. We can conclude that Chrome is the most energy-efficient Browser, but Brave is also a viable option, presenting interesting results in different scenarios.

For future work, it would be to try to introduce more browsers to the test environment and to be able to test all browsers in all test scenarios to be more reliable. That said, it would be interesting to cover more scenarios such as email, changing a Word document, etc., and explore more different aspects in the existing scenarios.

We hope that with this research and the results obtained, it will be possible to help promote the information and growth of Green Software for more sustainable use. This study may also serve as an incentive or as a basis for other related studies.

Bibliography

Abdulsalam, S., Zong, Z., Gu, Q., & Qiu, M. (2015). Using the greenup, powerup, and speedup metrics to evaluate software energy efficiency. *Proc. of the 6th Int. Green and Sustainable Computing Conference*, 1–8.

BATISTA, M. M. (2020). Tensões entre privacidade e publicidade direcionada [Em linha]. http://hdl.handle.net/10071/21072

Behrouz, R. J., Sadeghi, A., Garcia, J., Malek, S., & Ammann, P. (2015). Ecodroid: An approach for energy-based ranking of android apps. *2015 IEEE/ACM 4th International Workshop on Green and Sustainable Software*, 8–14. https://doi.org/10.1109/GREENS.2015.9

Borgolte, K., & Feamster, N. (2020). Understanding the performance costs and benefits of privacy-focused browser extensions. *Proceedings of The Web Conference 2020*, 2275–2286. https://doi.org/10.1145/3366423.3380292

Bouaffar, F., Le Goaer, O., & Noureddine, A. (2021). PowDroid: Energy Profiling of Android Applications. *2nd International Workshop on Sustainable Software Engineering (SUSTAINSE)*. https://hal.archives-ouvertes.fr/hal-03380605

Campbell, D. T. (1979). *Quasi-experimentation: Design & analysis issues for field settings*. Boston.

Cohen, J. (2013). *Statistical power analysis for the behavioral sciences*. Academic press.

Couto, M., Car cão, T., Cunha, J., Fernandes, J. P., & Saraiva, J. (2014). Detecting anomalous energy consumption in android applications. In F. Quintão Pereira (Ed.), *Programming languages* (pp. 77–91). Springer Int. Publishing. https://link.springer.com/chapter/10.1007/978-3-319-11863-5_6

Cruz, L., & Abreu, R. (2017). Performance-based guidelines for energy efficient mobile applications. *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, 46–57. https://doi.org/10.1109/MOBILESoft.2017.19

Cruz, L., & Abreu, R. (2019). Catalog of energy patterns for mobile applications. *Empirical Software Engineering*, *24*(4), 2209–2235. https://doi.org/10.1007/s10664-019-09682-0

David, H., Gorbatov, E., Hanebutte, U. R., Khanna, R., & Le, C. (2010). Rapl: Memory power estimation and capping. *2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*, 189–194. https://doi.org/10.1145/1840845.1840883

de Macedo, J., Aloísio, J., Gonçalves, N., Pereira, R., & Saraiva, J. (2020). Energy wars - chrome vs. firefox: Which browser is more energy efficient? *2020 35th IEEE/ACM International Conference*

on Automated Software Engineering Workshops (ASEW), 159–165. https://doi.org/10.1145/3417113.3423000

Di Nucci, D., Palomba, F., Prota, A., Panichella, A., Zaidman, A., & De Lucia, A. (2017). Petra: A software-based tool for estimating the energy profile of android applications. *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, 3–6. https://doi.org/10.1109/ICSE-C.2017.18

Druin, A. (2009). *Mobile technology for children: Designing for interaction and learning*. Morgan Kaufmann.

Fu, B., Lin, J., Li, L., Faloutsos, C., Hong, J., & Sadeh, N. (2013). Why people hate your app: Making sense of user feedback in a mobile app store. https://doi.org/10.1145/2487575.2488202

Gelenbe, E., & Caseau, Y. (2015). The impact of information technology on energy consumption and carbon emissions. *2015*(June). https://doi.org/10.1145/2755977

Gomez, L., Neamtiu, I., Azim, T., & Millstein, T. (2013). Reran: Timing- and touch-sensitive record and replay for android. *2013 35th International Conference on Software Engineering (ICSE)*, 72–81. https://doi.org/10.1109/ICSE.2013.6606553

Hao, S., Li, D., Halfond, W. G. J., & Govindan, R. (2013). Estimating mobile application energy consumption using program analysis. *2013 35th International Conference on Software Engineering (ICSE)*, 92–101. https://doi.org/10.1109/ICSE.2013.6606555

Hsu, D., Kanj, I., & Trieu, M. (2017). Chrome VS Firefox VS Opera on Android : Which Browser Consumes Less Energy in Android ? (2614921), 1–35.

Hu, Y., Yan, J., Yan, D., Lu, Q., & Yan, J. (2018). Lightweight energy consumption analysis and prediction for android applications. *Science of Computer Programming*, *162*, 132–147. https://doi.org/https://doi.org/10.1016/j.scico.2017.05.002

Li, D., Lyu, Y., Gui, J., & Halfond, W. G. J. (2016). Automated energy optimization of http requests for mobile applications. *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, 249–260. https://doi.org/10.1145/2884781.2884867

Li, D., Hao, S., Halfond, W. G. J., & Govindan, R. (2013). Calculating source line level energy information for android applications. https://doi.org/10.1145/2483760.2483780

Lima, L. G., Soares-Neto, F., Lieuthier, P., Castor, F., Melfe, G., & Fernandes, J. P. (2016). Haskell in green land: Analyzing the energy behavior of a purely functional language. *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 517–528. https://doi.org/10.1109/SANER.2016.85

Linares-Vásquez, M., Bavota, G., Bernal-Cárdenas, C., Oliveto, R., Di Penta, M., & Poshyvanyk, D. (2014). Mining energy-greedy API usage patterns in android apps: An empirical study. *11th Working Conference on Mining Software Repositories, MSR 2014 - Proceedings*, 2–11. https://doi.org/10.1145/2597073.2597085

Liu, K., Pinto, G., & Liu, Y. D. (2015). Data-oriented characterization of application-level energy optimization. In A. Egyed & I. Schaefer (Eds.), *Fundamental approaches to software engineering* (pp. 316–331). Springer Berlin Heidelberg. https://link.springer.com/chapter/10.1007/978-3-662-46675-9_21

Lo, C. D., & Qian, K. (2010). Green computing methodology for next generation computing scientists. *2010 IEEE 34th Annual Computer Software and Applications Conference*, 250–251. https://doi.org/10.1109/COMPSAC.2010.31

Monkeyrunner. (2020). https://developer.android.com/studio/test/monkeyrunner

Monsoon power unit. (2020). https://www.msoon.com/powermonitor-support

Nejati, J., & Balasubramanian, A. (2016). An in-depth study of mobile browser performance. *Proceedings of the 25th International Conference on World Wide Web*, 1305–1315. https://doi.org/10.1145/2872427.2883014

Neuvo, Y. (2004). Cellular phones as embedded systems. *2004 IEEE International Solid-State Circuits Conference (IEEE Cat. No. 04CH37519)*, 32–37.

Noureddine, A., Rouvoy, R., & Seinturier, L. (2013). A review of energy measurement approaches. *SIGOPS Oper. Syst. Rev.*, 42–49. https://doi.org/10.1145/2553070.2553077

Noureddine, A., Rouvoy, R., & Seinturier, L. (2015). Monitoring energy hotspots in software. https://doi.org/10.1007/s10515-014-0171-1

Pandruvada, S. (2014). *Intel® power governor*. Retrieved November 26, 2020, from https://01.org/blogs/2014/running-average-power-limit-%E2%80%93-rapl

Pang, C., Hindle, A., Adams, B., & Hassan, A. E. (2016). What do programmers know about software energy consumption? *IEEE Software*, *33*(3), 83–89. https://doi.org/10.1109/MS.2015.83

Pereira, R., Couto, M., Cunha, J., Fernandes, J. P., & Saraiva, J. (2016). The influence of the java collection framework on overall energy consumption. *2016 IEEE/ACM 5th International Workshop on Green and Sustainable Software (GREENS)*, 15–21. https://doi.org/10.1109/GREENS.2016.011

Pereira, R., Carção, T., Couto, M., Cunha, J., Fernandes, J. P., & Saraiva, J. (2020). Spelling out energy leaks: Aiding developers locate energy inefficient code. *Journal of Systems and Software*, *161*, 110463. https://doi.org/10.1016/j.jss.2019.110463

Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J. P., & Saraiva, J. (2017). Energy efficiency across programming languages: How do energy, time, and memory relate? 256–267. https://doi.org/10.1145/3136014.3136031

Pinto, G., Liu, K., Castor, F., & Liu, Y. D. (2016). A comprehensive study on the energy efficiency of java's thread-safe collections. *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 20–31. https://doi.org/10.1109/ICSME.2016.34

Pinto, G., & Castor, F. (2017). Energy efficiency: A new concern for application software developers. *60*, 68–75. https://doi.org/10.1145/3154384

Pinto, G., Castor, F., & Liu, Y. D. (2014). Mining questions about software energy consumption. https://doi.org/10.1145/2597073.2597110

Ramesh, G., & Umarani, R. (2012). Performance analysis of most common encryption algorithms on different web browsers. *International Journal of Information Technology and Computer Science*, *4*(12), 60–66. https://www.mecs-press.org/ijitcs/ijitcs-v4-n12/IJITCS-V4-N12-6.pdf

Robinson, S. (2009). Cellphone energy gap: Desperately seeking solutions. *Strategy Analytics*.

Rua, R., Fraga, T., Couto, M., & Saraiva, J. (2020). Greenspecting Android virtual keyboards. *Proceedings - 2020 IEEE/ACM 7th International Conference on Mobile Software Engineering and Systems, MOBILESoft 2020*, 98–108. https://doi.org/10.1145/3387905.3388600

Silven, O., & Jyrkkä, K. (2007). Observations on power-efficiency trends in mobile communication devices. *EURASIP Journal on embedded systems*, *2007*, 1–10. https://link.springer.com/content/pdf/10.1155/2007/56976.pdf

STAFF, H. C. (2021). *Ncsa mosaic internet web browser: The complete history*. Retrieved August 10, 2021, from https://history-computer.com/history-of-the-ncsa-mosaic-internet-web-browser/

Statcounter. (2020a). *Mobile browser market share worldwide*. Retrieved November 10, 2020, from https://gs.statcounter.com/browser-market-share/mobile/worldwide/#monthly-201909-202009-bar

Statcounter. (2020b). *Mobile operating system market share worldwide*. Retrieved November 26, 2020, from https://gs.statcounter.com/os-market-share/mobile-tablet/worldwide/#monthly-201909-202009-bar

Statista. (2021). *Number of smartphone users worldwide from 2016 to 2021*. Retrieved November 26, 2020, from https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

Thiagarajan, N., Aggarwal, G., Nicoara, A., Boneh, D., & Singh, J. P. (2012). Who killed my battery? analyzing mobile browser energy consumption. https://doi.org/10.1145/2187836.2187843

Thorwart, A., & O'Neill, D. (2017). *Camera and battery features continue to drive consumer satisfaction of smartphones in us*. Retrieved November 29, 2020, from https://www.prnewswire.com/news-releases/camera-and-battery-features-continue-to-drive-consumer-satisfaction-of-smartphones-in-us-300466220.html

Trepn profiler. (2013). https://developer.qualcomm.com/forum/qdn-forums/software/trepn-power-profiler/29775

Wikipedia. (2021). *List of most-viewed youtube videos*. Retrieved August 10, 2021, from https://en.wikipedia.org/wiki/List_of_most-viewed_YouTube_videos