Mobile Energy Efficiency Services

Co-promoted Project ID: CENTRO-01-0247-FEDER-047256

E6.4 . White paper on the main results of the project

Authors: R&D Teams from UNIVERSIDADE BEIRA INTERIOR, UNIVERSIDADE DE COIMBRA, and Caixa Mágica Software

Version: 1.0 (September 2022)

PORTUGAL 2020 — UNIÃO EUROPEIA — Fundos Europeus Estruturais e de Investimento

INDEX

1 Introduction

This section contains a brief introduction to the **Greenstamp** project, with its motivation and objectives. At the end of the section, one may find the description and objectives of the task being studied in this document. This general introduction aims to give the context of the project to new team members who participate in only one task.

The rest of the document contains the necessary sections for the research and development of the task being addressed.

1.1 Motivation

The importance that mobile devices have in our lives is such that it is difficult to imagine our daily activities without their use. The use of smartphones, tablets and, more recently, wearables such as smartwatches, has changed and simplified not only the way we communicate, but also the way we have fun, individually and collectively, or the way we work and do business. In fact, the number and scope of mobile apps seem limitless, but users still have increasing expectations about them. The distribution of such applications is highly facilitated by digital markets, which democratise the opportunity to market software to mobile devices. In 2017, the number of mobile applications installed was 178 billion, a figure estimated to grow to 258 billion in 2022.

While much of the app market is targeted at mobile devices, whose autonomy depends on the limited battery life, the fact is that today's markets do not provide any indication of the energy efficiency, absolute or relative, of the applications they offer. With this gap in mind, in the **GreenStamp** project we propose to investigate and develop innovative mechanisms for analysing and cataloguing the energy efficiency of mobile applications integrated into app store processes. Pedagogical recommendation systems for developers will also be studied, on how to improve the efficiency of their applications, and for users, of energy-efficient applications aligned with their profile. The objective is to reduce at least 20% of the energy consumed by applications that follow the technical recommendations proposed and, inherently, of the mobile devices where they are installed, thus contributing to a significant savings of resources consumed in the mobile market, particularly and immediately by the large user base of the company promoting this project (250 million unique users active in 2019).

1.2 Objectives

The **GreenStamp** project aims to investigate and develop techniques and technologies capable of analysing, cataloguing, and informing about the energy efficiency of mobile applications and how to optimise it, thus reducing the energy consumption of the mobile market. This goal is realised in direct impacts on the wide range of citizens who are consumers of apps, mobile application companies and app stores.

As for app consumers, the impact will be first and foremost on your satisfaction. This will come from the certainty that you can have energy-efficient applications that optimise your device's resources, rather than having applications that, in some cases you're usually unaware of at first, drain your

mobile device's limited energy resources with excessive and unnecessary battery consumption. This certainty will be achieved by providing information on the energy profile of applications at the time of their choice and installation, and through a system of recommendations to be investigated. Such knowledge will allow consumers to opt for energy-efficient solutions, as is currently the case in other markets (home appliances, automobiles, real estate, and others), thereby optimising the energy performance of their device and increasing the autonomy time of the device. Thus, a user who chooses efficient applications, will charge your device less often, will have a lower cost in your energy bill and reduce the risk of developing nomophobia, with the certainty that the energy consumption of your applications is great.[1]

1.3 Main contributions

To achieve the strategic objective and impacts mentioned, this project aims to research and develop highly innovative techniques and technologies, unparalleled in the market. They are translated into the following technical-scientific objectives:

- Investigate and conceptualise new systems for the acquisition, processing and analysis of data related to the energy efficiency of mobile applications;
- Investigate and conceptualise innovative machine learning mechanisms and cataloguing energy consumption patterns of mobile applications based on static and dynamic data;
- Investigate and conceptualise ways to information to users about the energy efficiency of apps, and relevant recommendations related to this factor;
- Investigate and conceptualise models and mechanisms of technical and action-oriented recommendation to mobile application promoters, on how to optimise this parameter in their products, in an integrated way in their practice;
- Investigate and conceptualise a new interface to support decision and system management.

[1] https://www.infopedia.pt/dicionarios/lingua-portuguesa/nomofobia

2 Description of Tasks

T1.1: Research on energy efficiency analysis of mobile applications in centralised and decentralised Cloud architectures

Leader: UBI; **Participation**: UC and CMS; **Output**: E1.1

To support and justify technological and scientific choices as well as to motivate innovation in the proposed solutions, we will lead a thorough and exhaustive analysis to the state of the art over 1) the techniques of evaluation (of efficiency) energy; 2) the computer support platforms for such evaluations; 3) a classification of such mechanisms (automatic, semi-automatic, with or without preparation/prior instrumentation of the mobile code by analysing, on source code, on executable code, etc.).

A particular emphasis will be placed on architectural solutions and their impact on the potentiation of such analyses. If a Cloud architecture is consensual, many alternatives within this technological context are possible: how to break down the computing, a file, a data collection? What will be the impact? Centralised? Decentralised? It will be relevant to establish how these issues impact the state of the art. Aspects such as availability, resilience, confidentiality, impact on the personalization/fairness of the analysis (etc.) will be studied.

The whole study will deal particularly with the survey of knowledge in the technological ecosystem of app stores, according to the solution that was specified, or, due to the dissolving of innovation, in other systems where energy efficiency analysis is performed. An exhaustive knowledge base will be created on app energy efficiency analysis, using dynamic or static analysis, in centralised Cloud architectures (centralised processing), decentralised architectures, users' or developers' devices, and hybrids. This work will make it possible to advance the following tasks in an informed manner regarding the existing technical-scientific knowledge.

The scientific skills of UBI team members and employees, in particular the study of centralised vs. decentralised models, lead to the choice of leadership in this task.

T1.2 - Requirements of the data analysis, acquisition, and processing system

Leader: UC; **Participation**: CMS; **Output**: E1.1.

The objective of this task is to identify functional requirements, technical constraints, quality attributes and business objectives that influence the architecture, development, validation, deployment and maintenance of the system of analysis, acquisition and processing of data. This system is anticipated to consist of a developers-side static analysis component, a user-side dynamic analysis component, and a centralised orchestrator component on the app store side. To achieve this goal, the following actions will be carried out:

- Requirements' elicitation: it will be carried out through interviews, by the development team to all stakeholders of the project and by the analysis of the state of the art and current practices in this field. The promoters should promote workshops, with the participation of

stakeholders, dedicated to elicitation, analysis and validation of requirements and quality attributes.

- Requirements' analysis: trade-offs, or conflicts, will be identified between the identified requirements; user stories will be developed to document the identified requirements, as well as quality attribute scenarios; metrics to develop and validate the system will also be identified.
- Documentation: After the previous step, the identified requirements and quality attributes will be recorded and specified in a document that will contain all requirements relevant to the software architecture.
- Validation: The document will be validated by stakeholders.
- Management: Throughout the implementation, this being a research project, it is expected that changes to the requirements will arise and that there will be unidentified restrictions at the outset, both situations may result in requests for changes to the requirements and quality attributes.

The results of this task will be used to formulate in an articulated way the technical-scientific roadmap in T1.5 and will feed the beginning of the research task that will seek to meet the requirements raised here in A2.

UC leads the task based on the scientific skills of its team, and in particular its experience in requirements gathering other systems currently in production in the industrial context.

T1.3 - Requirements of machine learning and profiling mechanisms of application energy consumption patterns, and recommendation systems for stakeholders

Leader: UBI; **Participation**: CMS; **Output**: E1.1.

This task focuses on the mechanisms of machine learning and profiling of patterns of energy consumption of apps, and from recommendation systems to stakeholders. As in the previous task, the objective of this task is also to identify functional requirements, technical constraints, quality attributes and business objectives that have influence on architecture, development, validation, deployment and maintenance of learning, cataloguing and recommendation mechanisms.

The success of this goal will be supported by the elicitation and analysis of requirements, preparation and validation of the documentation, and a dynamic management that reacts to any necessary change in the face of changes in requirements and other restrictions and problems that may occur.

This task will feed the development to be carried out in activity A3, and UBI is the leader of either this task or the said activity.

T1.4 - Requirements for decision support interface and management, and recommendation channels

Leader: CMS; **Participation:** UC; **Output**: E1.1.

This task will unite the knowledge generated in the T1.1 task to implement participatory design with users similar to end users, to meet the requirements of the decision support interface and management, and the channels of recommendations to users and developers regarding the energy efficiency of apps. to define the interface-specific initial requirements. On the other hand, they will be considered members of the team that develops the application distribution system to better understand the requirements of recommendation channels to mobile users and developers. If it is necessary, end users of the app store, users and developers of apps will also be used. from various geographical points that collaborate with the company informing the needs and characteristics of users of various geographies. This strategy of involvement of the main stakeholders from the initial moments will make them feel co-authors of the innovative technology that comes from, which will promote their acceptance at the end of the project.

In conjunction with tasks T1.2 and T1.3, the requirements to be raised in this task are related to three distinct interfaces. On the one hand, it is intended to adjust and detail the solution proposed in this application in terms of the interface of decision support and management of the system. On the other hand, it is intended to define the requirements in terms of recommendations (and recommendation formats) that application developers value in terms of optimising the energy efficiency of their applications. The third strand concerns recommendations to mobile app consumers, and their requirements in terms of what is valued and useful in terms of app energy efficiency on consumer devices.

The stakeholders consulted here will be again in each phase of evaluation of the solution (activity A6), informing R&D with the real needs of the market where the results will apply throughout the project.

The results of this task will inform and launch the a4 activity research roadmap. Because this is a task that will lead to an app store management platform, and because it has the technical resources that by its experience are appropriate for the development of the table, the Magic Box will lead this task and, by inegency, the A4 activity.

Assumptions and dependencies

The research carried out under this task is quite relevant for the tasks Txx, Tyy and Tzz, because (...). The investigation of this task also depends on the results of the tasks of Tuu, Tvv and Tww, because (...).

3 Research on energy efficiency analysis of mobile applications in centralised and decentralised Cloud architectures

(Work in progress)

This section presents the current state of technology regarding green computing applied to mobile devices. The research available is presented and discussed in three different sections, each focusing on a key aspect of this thesis that will later prove itself useful in the development of the solution in which the thesis is focused on.

The Research section presents articles that provide insight into energy consumption analysis and its importance for the user.

The section Existing Energy Profilers will present the energy profilers for mobile devices which are software and hardware tools available to attain predictive models for the energy consumption of mobile devices and an extensive look into which ones may suit the intended use for this project.

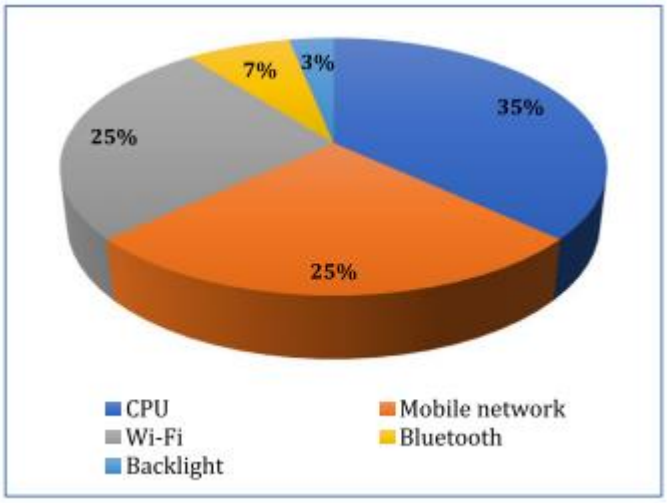
Lastly, the Machine Learning Algorithms will present insights into the available models to solve the problem that is the main focus of this research.

All research presented in this section was developed in the last five years with the exception of two articles presented in the Existing Energy Profilers, were conducted seven years ago however they contain an extensive list of profilers and in depth analysis of said profilers results that was deemed important for a better understanding of this subject matter.

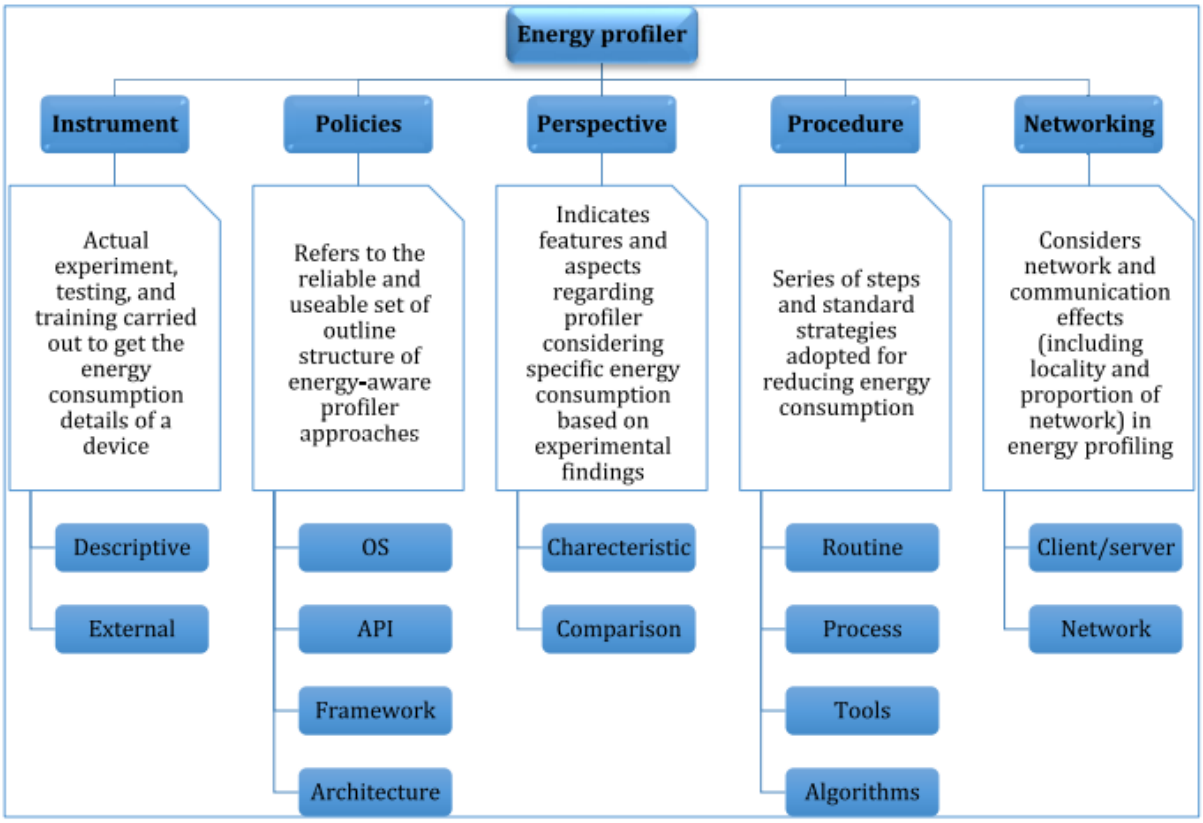
Research

Power Consumption Analysis, Measurement, Management, and Issues: A State-of-the-Art Review of Smartphone Battery and Energy Usage

The article [9] provides an in-depth look at how smartphones and their batteries function and how to measure and reduce the energy consumption through the use of varied techniques. It also analyses the available research into the development of smartphone batteries and hazards associated with these, providing some solutions on how to mitigate them.



Despite the main focus of the article being the smartphone batteries, its contributions to this thesis come mainly from its comprehensive study into the energy consumption of the various components of the smartphone from which the author attains a surface understanding of the energy consumption distribution percentiles between them and of the available profilers and models for measurement and diagnosis of energy consumption providing further information on the development of these software tools.



PORTUGAL 2020
UNIÃO EUROPEIA
Fundos Europeus Estruturais e de Investimento

The thesis [2] was conducted with the intent to develop a mixed strategy between user side preventive power-saving plans with conventional detective strategies and thus come up with an adequate rating system for the apps present in the Google Play Store. Through an in depth look at the current state of green computing, user experience and the information made available by Google in their app store, more specifically the list of permissions pertaining to the app, the author was able to develop a energy rating scheme with the use of stars to represent the quality in terms of energy usage of the application. This thesis utilises Power Tutor in order to develop an energy model of the Samsung I9500 so as to then apply the estimated consumption of the varied components and then associate them to their respective permissions as presented in table below.

From the excerpt of this table we can see that the authors of the study managed to produce a relatively, although simplistic, rating for the energy consumption of each permission through the mobile components they utilise. All of this was possible due to the correlation between the data obtained from Power Tutor and human analysis.

Given the subject matter of this paper, this study is of high relevance, allowing the author to attain an overview of the recent state of green computing with focus on the mobile market and to gather valuable insights on how to approach the problem meant to be solved by this thesis.

Existing Energy Profilers

Modelling, Profiling, and Debugging the Energy Consumption of Mobile Devices

The focus of the article [8] lies in an extensive explanation on how to develop an energy profiler and the analysis of the literature related to several energy profilers with the intent of comparing their performance and ease of use. The authors divide the profilers into three different categories, the categories and respective profilers are the following:

- On-Device profiler with on Device model:
  - Nokia Energy Profiler;
  - Trepn Profiler;
  - Power Booter;
  - Se-same;

- DevScope;
- AppScope;
- V-edge.

- On-Device Profiler with off-Device model:
  - Android Power Profiler;
  - Power Tutor;
  - Power Prof.
- Off-device in Laboratory:
  - PowerScope;
  - Joule Watcher;
  - Fine-grained Profiling with Eprof;
  - Banerjee et al;
  - Shye et al.

The main contribution of this article is the pooling of available profilers, their respective accuracy values and comparison of components between them allowing for an easier choice of which profiler to apply to this thesis. From the result of this survey it is possible to determine that the Trepn profiler has the best accuracy and analyses the consumption of most components that exist in smartphones, however these accuracy values are those reported by the developers of the various profilers and were not tested by the authors of this article and thus, the values may be biassed towards the applications that were used by the developers to test the profilers.

Profiler	Disp	CPU	GPU	GPS	BT	Wi-Fi	3G	4G	Cam	SD Card	Audio	Reported Accuracy
Trepn	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	99%
V-edge	✓	✓	✗	✓	✗	✓	✗	✗	✗	✗	✗	86%
BatteryStats	✓	✓	✗	✓	✓	✓	✗	✗	✗	✗	✗	Not Reported
PowerBooter	✓	✓	✗	✓	✗	✓	✓	✗	✗	✗	✓	96%
PowerTutor	✓	✓	✗	✓	✗	✓	✓	✗	✗	✗	✓	97.5%
DevScope	✓	✓	✗	✓	✗	✓	✓	✗	✗	✗	✗	95%
AppScope	✓	✓	✗	✓	✗	✓	✓	✗	✗	✗	✗	92%
Sesame	✓	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	86%
Eprof	✓	✓	✗	✓	✗	✓	✓	✗	✓	✓	✗	94%
Banerjee et al. [2014]	✓	✓	✗	✓	✗	✓	✓	✗	✗	✗	✓	Not Reported
Shye et al. [2009]	✓	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	93%

The main intent of this study [1] is the development of a concise taxonomy so as to facilitate the understanding and separation of the varied existing methods available for energy profiling of mobile devices, this is then substantiated through an analysis of different methods that suit the two main categories the authors theorised, these categories and profilers are the following:

- Software based:
  - Power-prof;
  - Power Booter;
  - Se-same;
  - Hybrid-feedback;
  - SEMO;
  - Elens;
  - ARO;
  - Wattson;
  - Eprof;
  - P-top.

- Hardware based:
  - DuT;
  - Netw-trace;
  - Power Memo;
  - PowerScope;
  - Network;
  - Web-browser;
  - Multi-core CPU.

For the purposes of this article, the authors focused mainly on the analysis of the software energy profilers as seen in the table below.

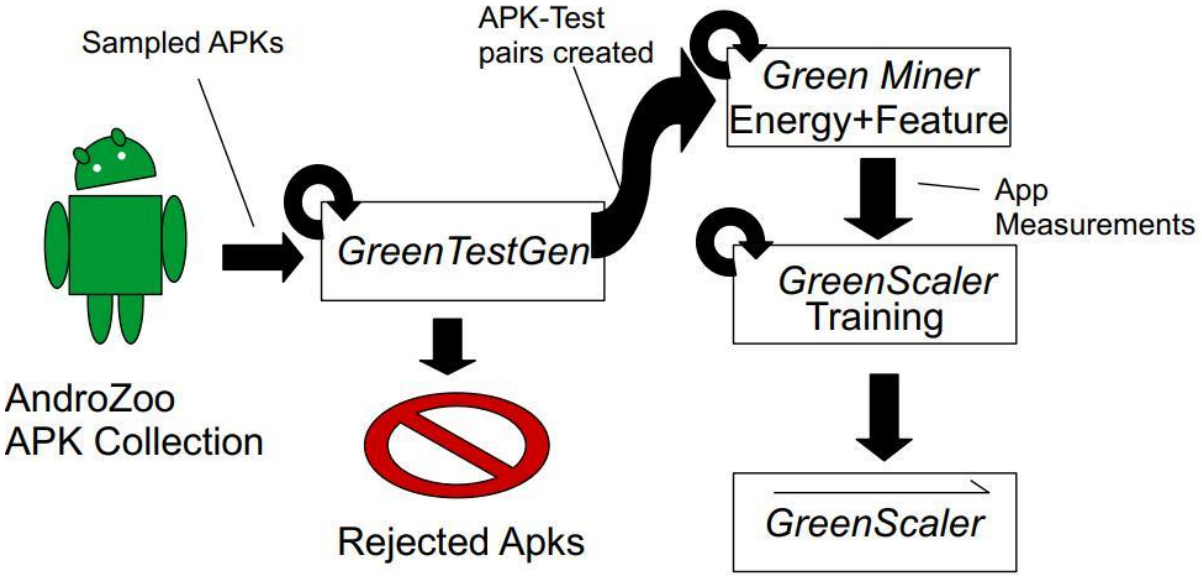
Schemes (Ref.)	Resources analysed								Description
	CPU	LCD	GPS	Wi-Fi	UMTS	GSM	Accelerometer	Compass	
Power Proff (Kjærgaard and Blunck, 2012)	✓		✓	✓		✓	✓	✓	Unsupervised GA based energy profiling
Power Booter (Zhang et al., 2010)	✓	✓	✓	✓	✓				Automated smart battery interface based power model construction
Se-same (Dong and Zhong, 2010)	✓	✓		✓					High rate and accurate self-power modeling
Hybrid-feedback (Gurun and Krintz, 2006)	✓					✓			Hybrid (on-line/off-line) feedback based energy profiling to reduce power model construction time
SEMO (Ding et al., 2011)	✓								Low-rate smart energy monitoring system based profiling to improve accuracy
ELens (Hao et al., 2013)	✓		✓	✓					Program analysis based fine-grained energy profiling
ARO (Qian et al., 2011)	✓				✓				Cross-layer interaction based power optimization
Wattson (Mittal et al., 2012)	✓	✓		✓	✓				Emulation based profiling
Eprof (Pathak et al., 2012)	✓		✓	✓					Fine-grained energy profiling for smart phone applications
P-top (Do et al., 2009)	✓	✓				✓			Process level software power profiling

This article contributes to a deeper and concise knowledge on the available energy profilers and their capabilities. Its contribution for this paper lies on widening the pool of available profilers.

GreenScaler: training software energy models with automatic test generation

GreenScaler [4] is a robust energy model developed with Android developers in mind so as to allow them to estimate the energetic consumption of the applications that they develop. This energy model software makes use of random test generation and a CPU usage focused heuristic in order to select which test cases are best, through this, GreenScaler is able to produce relatively accurate energy models for an application with an upper error bound of 10% when using randomly generated test cases and an even lower error percentage when applied to manually written tests. It is also able to detect energy regression within different versions of an app as long as it's a relevant difference.

This energy model software is able to achieve its purpose, through the use of GreenMonkey, an adaptation of Android Monkey the UI/Application exercisers. This adaptation was developed since the original was somewhat limited, mainly because it didn't allow for the user to define an event distribution and it included irrelevant events as well that aren't related to the app itself. Later in the research the authors compared their version's performance to the original's and found that their version produced much better results when the CPU-utilisation heuristics were applied than the original and thus opted with GreenMonkey for automatic test generation. Below we can see a simple image depicting the process flow of GreenScaler.

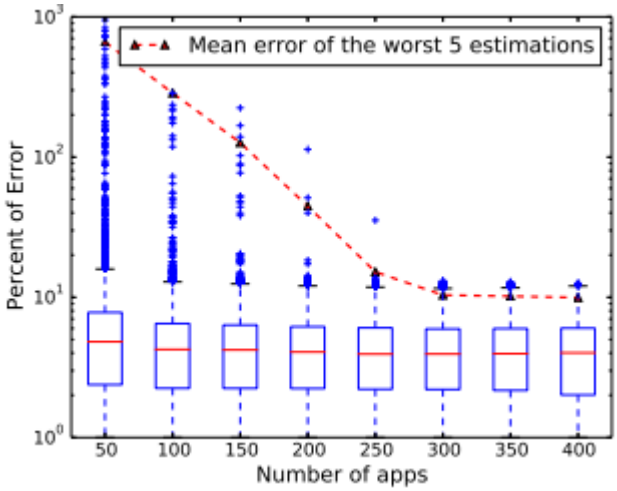


The authors evaluation of both resource usage heuristics, these being a CPU-utilisation heuristic and an energy estimation heuristic, was done by using the same feature table but using the different datasets obtained from implementing the different heuristics to the test selection process. The leave-one-out method was applied so as to test the accuracy of the model, the Anderson-darling normality test was also utilised and it determined that both error distributions are not normally distributed and thus the Kruskal-Wallis test was used which yielded that the error distributions from the models obtained out of the different heuristics were statistically different.

Due to this, the Wilcoxon-rank-sum test was applied to attain the 99% confidence interval of mean percent error in joules as well as the Cliff's delta to measure the effect size between them.

After the statistical analysis was complete, the chosen model for GreenScaler was the Model based off of CPU-utilisation, due to it having a better upper error bound and a difference of 3% in the mean error for the 5% worst estimations. The CPU heuristics model also benefits from its simplicity since it only requires to capture the CPU jiffy (period of an alternating current power cycle) information whereas the energy model heuristic requires far more data and to trace every single system call by an app.

The research detailed in this report is highly relevant due to important insights gathered from it that will benefit the projects future development, e.g. when implementing a test selection, a simple CPU-utilisation heuristic seemed to be the best performing one, being preferred over the use of a more complex energy estimation or a code coverage heuristic. In terms of training the model it was found that with 400 different apps in the training set, it's able to reach the upper error-bound of 10%, however the error rate decay slows down considerably after using 300 apps in the training set as seen in the graphic below..



Software-Based Energy Profiling of Android Apps: Simple, Efficient and Reliable?

PETrA [6] or Power Estimation Tool for Android is a software-based tool developed with the intent of understanding if it was possible to build such a tool and attain accurate energy consumption measurements without the use of hardware-based tools that are expensive to acquire but provide the most accurate measurements.

This tool functions in the following manner, firstly it starts by handling app preprocessing in which PETrA is given an app, it's directory and is then tasked with installing and preparing the app for analysis, then it proceeds to compute the energy profile of the app by receiving a test case, this test can be a manually written one or one generated automatically.

After the test is finished then, through the use of Project Volta's Android tools, PETrA is capable of creating a power profile of the app and from it, together with the use of certain formulas, pinpoint how much energy the app consumes when performing certain tasks, finally the tool generates an output, a csv file with the energy estimations for each of the method calls the app executes.

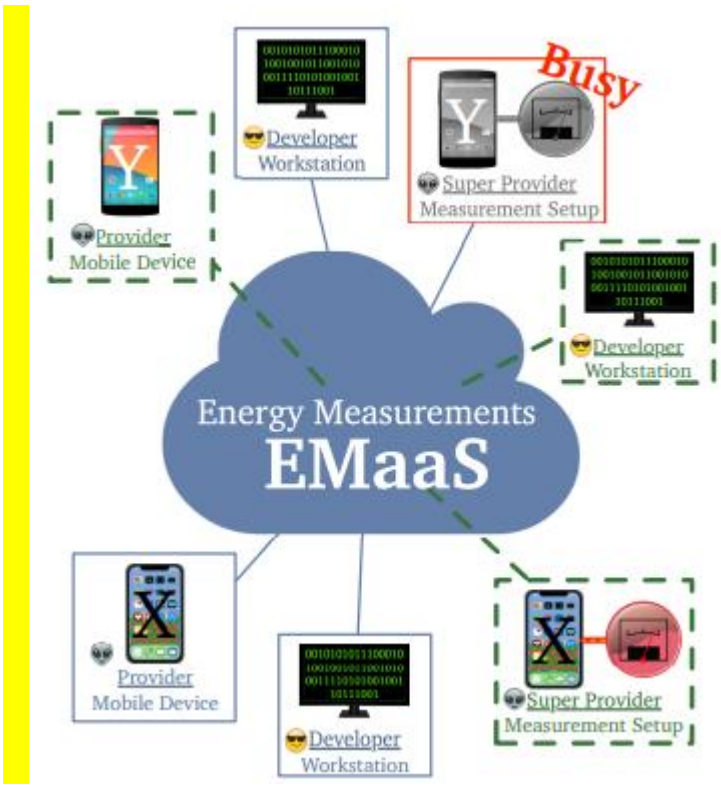
EMaaS: Energy Measurements as a Service for Mobile Applications

The paper [5] details the development of a peer-to-peer cloud based system with the intent to supply mobile developers with a platform in which they can have access to various energy models and hardware analysis methods so they can assess the energy consumption levels of the applications they are developing.

This is an incredibly useful feature since it allows developers to analyse the energy consumption of their application without having to acquire different sets of tools and knowledge which they may not have and may be expensive to acquire by themselves.

In order to accomplish this feat, the authors developed the platform with three different types of users in mind, being these the Developers, users with the intent of using the platform for the assessment of the energy consumption of the various apps they develop, the Providers, users who supply the network with devices which will allow the applications from developers to run on said devices in order to estimate the energy consumption resorting to adequate energy models to achieve this end, and finally the Super-Providers, who will provide hardware based energy measurements in order to create a trust value for comparison with the energy model acquired measurements.

Despite the three different types of user it is possible for any one user to meet requirements for all three types and fulfil all tasks simultaneously.



The process of measuring said consumption occurs has follows:

1. Developer requests assessment for his application by providing an APK to the platform along with an instrumentation build and test cases for said app;
2. APK is sent to a Provider and a Super-Provider, for this to be properly processed it is required that both these parties have the same device model as the Developer;
3. Provider returns energy consumption measurements based off of a energy model adequately arranged for the device in question whilst the Super-Provider will provide

measurements based off of readings from the monitoring of hardware readings with the use of proper equipment for the task;

4. Results are returned to the Developer and the process is concluded.

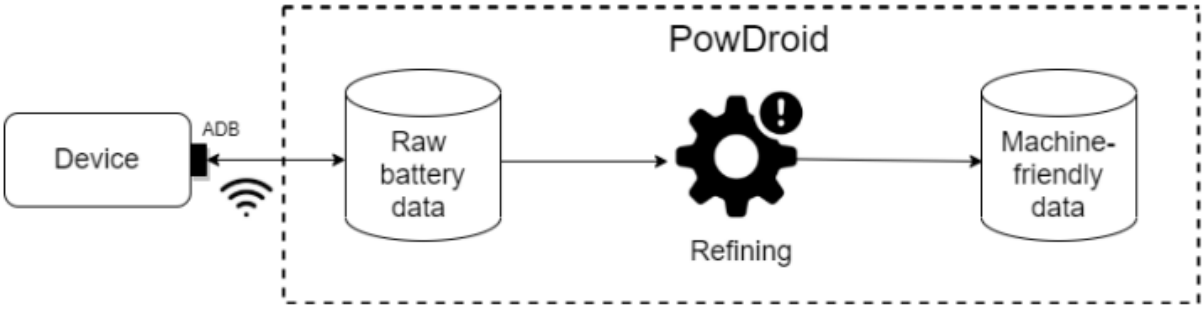
This process sustains itself on two major implemented mechanisms to assess which results can be delivered to the Developer so he can attain accurate data, these are the Reliability Consultant, the Energy Model available, the Hardware-based power Monitor is also crucial however it's only used has a trust base system for the previous two instead of being an implemented algorithm.

When a Developer requests an energy measurement this request will first go through the reliability consultant in order to determine if the available energy model for said device model is enough to accurately determine the measurement on its own. It does this through applying the values obtained from both the Provider and the Super-Provider in a mathematical formula that will evaluate said energy model's reliability, the closer to zero the result is the more reliable the model will be. In case the reliability of the model exceeds a certain threshold above or below zero, it will be considered as unreliable for the respective case and the values obtained from the Super-Provider will be returned to the developer as well as they will also be used to update the existing energy model and the reliability consultant so that these are better prepared for future requests, otherwise the energy model obtained value is used since it's assessed to be reliable.

PowDroid: Energy Profiling of Android Applications

Powdroid [3] is a fusion between utilisation-based and event-based energy profiler that checks the battery status at the moment of app initialization and compares it with its final value at the end of its execution. This is accomplished through the use of a Wi-Fi connection and a command-line tool with the phone and four specific tools, these are Batterystats, a tool in the Android framework that collects raw data from the battery, Bugreport, also from the Android framework and produces a zip file containing a report of the app's usage, Battery Historian, a tool from Google used to create an easy to read web-based visualisation of the report provided by Bugreport and lastly, the authors, run a few scripts in order to split, process and unite the metrics in order to return a CSV file containing a list of all events and corresponding usage data.

This profiler was tested with three different types of apps, these were web browsers, camera and weather applications, the authors state that their results are consistent with the results of recent comparative studies.

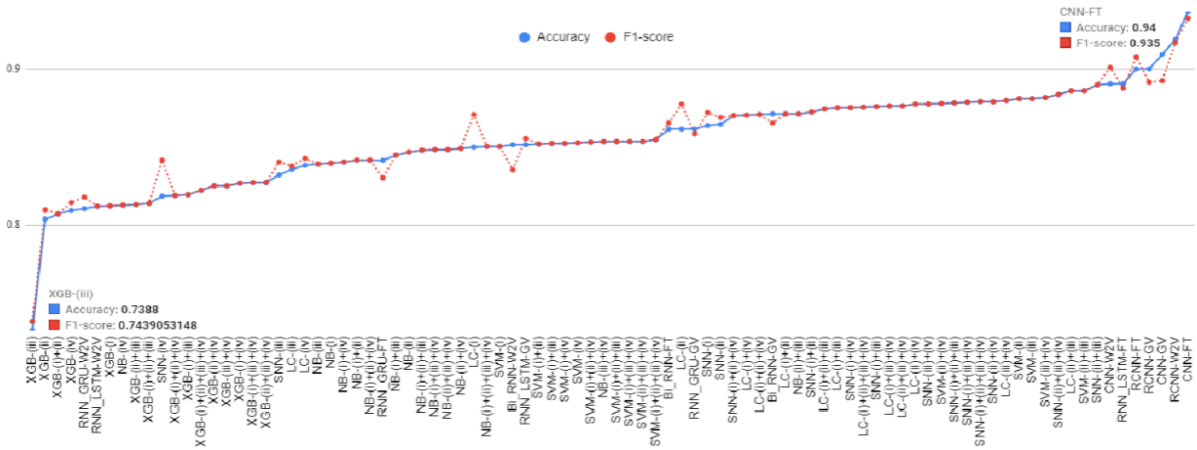


The limitations of this profiler are the following: Monitors the phone battery consumption as a whole, relies on battery and hardware information provided by Batterystats hence, it is not in real time and not fine grained. Also, the values provided are estimations based on the metrics they applied which means that battery estimations are based on battery drain and the energy consumption of the individual components are not mapped.

ReviewViz: Assisting Developers Perform Empirical Study on Energy Consumption Related Reviews for Mobile Applications

The study [7] deals with the development of a visualisation tool to help developers better sift through the user reviews of their apps. This tool specifically deals with the topic of energy consumption and tests various machine learning models with the intent of choosing the best one for natural language assessment of the topic at hand.

The results from the study show that the CNN-FT, a type of Convolutional Neural Network with the added capability of fault tolerance, is the model that attains the best accuracy and F1-score values (approximately 0.94 for accuracy and 0.935 for F1-score as it is shown below) relinquishing however, the run-time, taking approximately 35 to 40 minutes to complete.



Although the purpose of this study is the development of a data visualisation tool, it shows the possibility of using machine learning and user reviews to gather information about energy consumption, proving itself useful for the project.

Conclusion

The research performed found no machine learning solution to study the correlation between energy consumption and the data readily available about an application in its appstore page. Despite this, all the tools and knowledge deemed necessary for the development of said solution exist and are publicly available. In respect to energy efficiency analysis of mobile applications in centralised and decentralised cloud architectures however, several studies were found that apply some form of energy analysis in order to improve their efficiency, both in terms of resource usage and time spent as well as energy consumption, for either the mobile device or the cloud network with a preference for decentralized cloud structures in order to improve energy efficiency as seen with the articles [10],[11],[12] and [13].

[1] Raja Wasim Ahmad et al. “A Review on mobile application energy profiling: Taxonomy, state-of-the-art, and open research issues”. In: Journal of Network and Computer Applications 58 (2015), pp. 42–59.

[2] Abdullah Mahmoud Almasri. “Google Play Apps ERM: (Energy Rating Model) Multi-Criteria Evaluation Model to Generate Tentative Energy Ratings for Google Play Store Apps”. PhD thesis. 2021. url: http://hdl.handle.net/10284/9671.

[3] Fares Bouaffar, Olivier Le Goaer, and Adel Noureddine. “PowDroid: Energy Profiling of Android Applications”. In: 2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW). IEEE, 2021, pp. 251–254.

[4] Borle S. Romansky S. Chowdhury S. and A. Hindle. “GreenScaler: training software energy models with automatic test generation”. In: Empirical Software Engineering 24 (2018), pp. 1649–1692.

[5] Luis Cruz and Rui Abreu. “EMaaS: Energy Measurements as a Service for Mobile Applications”. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER). IEEE, 2019, pp. 101–104.

[6] Dario Di Nucci et al. “Software-based energy profiling of Android apps: Simple, efficient and reliable?” In: 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, 2017, pp. 103–114.

[7] Mohammad Abdul Hadi and Fatemeh Hendijani Fard. “ReviewViz: Assisting Developers Perform Empirical Study on Energy Consumption Related Reviews for Mobile Applications”. In: Proceedings of the IEEE/ACM 7th International Conference on Mobile Software Engineering and Systems. ACM, 2020, pp. 27–30.

[8] Mohammad Ashraful Hoque et al. “Modeling, Profiling, and Debugging the Energy Consumption of Mobile Devices”. In: ACM Comput. Surv. 48.3 (2015), pp. 1–40.

[9] Pijush Kanti Dutta Pramanik et al. “Power Consumption Analysis, Mea-

surement, Management, and Issues: A State-of-the-Art Review of Smart-

phone Battery and Energy Usage". In IEEE Access 7 (2019), pp. 182113–

182172.

[10] Y. Zhang, J. He and S. Guo, "Energy-Efficient Dynamic Task Offloading for Energy Harvesting Mobile Cloud Computing," *2018 IEEE International Conference on Networking, Architecture and Storage (NAS)*, 2018, pp. 1-4.

[11] E. Ahvar, A. -C. Orgerie and A. Lebre, "Estimating Energy Consumption of Cloud, Fog, and Edge Computing Infrastructures," in *IEEE Transactions on Sustainable Computing*, vol. 7, no. 2, 1 April-June 2022, pp. 277-288.

[12] C. Anuradha, M. Ponnavaikko, "A RNN based offloading scheme to reduce latency and preserve energy using RNNBOS", in Measurement: Sensors , Vol. 24 , 2022, 100429.

[13] Khadija Akherfi, Micheal Gerndt, Hamid Harroud, "Mobile cloud computing for computation offloading: Issues and challenges", in Applied Computing and Informatics, Volume 14, Issue 1, 2018, Pages 1-16.

4 Requirements of the data analysis, acquisition, and processing system

Develop a system for classifying the energy consumption of applications submitted to the Aptoide app store.

According to the methodology proposed in this project, this task is divided into five parts: requirements elicitation, functional requirements, business objectives, technical restrictions and quality restrictions, detailed right away.

4.1 Requirements Elicitation

- Requirements elicitation was carried out through a meeting between Caixa Mágica Software teams and members of the University of Coimbra on August 2, 2022.

4.2 Functional Requirements (FR)

4.2.1 FR01 - Provide the dispatcher with a way to invoke the energy certification service and obtain the result of the certifications carried out;

4.2.2 FR02 - Allow adding and removing certification techniques;

4.2.3 FR03 - Configure the output format and select the algorithm responsible for certification;

4.2.4 FR04 - Run functional assessment and results in correction tests;

4.2.5 FR05 - Allow parameterization of algorithms;

4.2.6 FR06 - Allow the Judge to obtain information about the certification;

4.2.7 FR07 - Delete previous results of energy assessments;

4.2.8 FR08 - Activate and deactivate the energy evaluation;
4.2.9 FR09 - Suggest improvements in the application code to allow improving the performance of energy consumption;
4.2.10 FR10 - Search and obtain information about certifications already carried out.

4.3 Business Objective

Allow the app store to make an option that competitors do not have available to its customers.

4.4 Technical Restrictions (TR)

4.4.1 TR01 - Be deployable with resources to containers in cloud environments.
4.4.2 TR02 - Snap to the current pipeline.
4.4.3 TR03 - Interact through web services with the components: dispatcher and Judge.

4.5 Quality Attributes (QA)

4.5.1 QA01 - When the system receives a new application, check if the apk has the requirements for analysing energy consumption.
4.5.2 QA02 - Allow deploying or updating without downtime.
4.5.3 QA03 - Dispatcher receive apk. When a new application is loaded in the app store, the apk and application information are passed to the energy analysis tool.
4.5.4 QA04 - Analyse apk energy consumption during the pipeline without interfering with critical pipeline flow.
4.5.5 QA05 - Prevent unauthenticated access to the system.
4.5.6 QA06 - The data considered for energy certifications must be saved and later made available.
4.5.7 QA07 - Availability?
4.5.8 QA08 - Recovery in case of error?
4.5.9 QA09 - Duration/historical amount to keep?
4.5.10 QA10 - Performance?

5 Requirements for machine learning and profiling mechanisms for app energy consumption patterns, and recommendation systems for stakeholders

For designing an ML-based recommender systems (MLRS), the team has provided a set of requirements to design a proper MLRS to decrease energy consumption. In the following, we listed the main requirements.

- Providing a method to consume the energy consumption for each function in each application. It should be automatic since, to gather an adequate volume of data, manual methods are insufficient.
- Extracting the affecting factors in energy consumption for a specific task. By doing this, the factors can be seen as a proxy of energy consumption, and as a result, they can be used interchangeably for energy consumption.
- An MLRS should be designed for a specific task. Therefore, it is important to identify the Energy-intensive components. Therefore, the focus should be on one of them.
- A large volume of data should be available to design an MLRS. Data should be extracted for a specific MLRS.
- There are several approaches to designing and implementing MLRS from deep learning methods to evolutionary computation. It is taken into consideration as one of the main requirements since the selection is based on data availability and their type.
- The amount of data should be enough to extract the proper models.
- To provide a specific MLRS, there must be appropriate data for it, including some features to describe the task as well as energy consumption (or a proxy or/and affecting factors).

In the following, we have reviewed the profiling mechanisms for energy consumption as well as recommendation systems for stakeholders.

From the literature, there is no AI-based-recommender system for stakeholders with the goal of energy consumption. The main problem in this area seems to be that there is no available proper data for this purpose. However, several studies have recommended hints and suggestions to diminish energy in smartphones and tablets. All of these studies are based on some quantitative and statistical analysis. To this end, they took two different conditions and tried to assess their energy consumption. In the following, the main contributions in this area to provide some recommendations are highlighted.

One of the early research studies on the scope of energy consumption is [1]. This paper is one of the most important research on practice coding for Android Apps to decrease energy consumption. It divides energy-saving best practices into three different categories, including, energy of networks, save-memory energy, and the programming tips to improve runtime performance. Among all best practices, they selected three as the representatives as follows:

1) **HTTP request:**

they measured the energy consumption of downloading different sized data files using HTTP request. Figure 1 shows the trend for energy consumption in terms of the different sized data files. The x-axis shows the number of bytes, downloaded by the request, while y-axes is the energy consumption in terms of mAh.

From the figure, the energy consumption of an HTTP request stays in the range of 0.5-0.6 mAh when the downloaded data is less than 1,024 bytes (1,000 on the x-axis in Figure 1). When the data size becomes larger, the energy consumption of the HTTP GET request has a roughly linear relationship with the size of downloaded data.

This observed trend is caused by the fixed overhead introduced by the packet header and control information in the HTTP protocol and its lower level network protocols, TCP and IP. Regardless of the size of the sent data, the packet header's size remains fixed. Instead of the packet size, the amount of the control information is mostly determined by the quantity of packets. Therefore, when the packet size is small, the majority of energy is used to send packet headers and control information. The scenario changes as the data size takes control as the volume of data rises. When examining network throughput, this inefficiency of transmitting little packets is also shown.

Therefore, the authors suggested some recommendations as follows:

Developers should refrain from making short HTTP queries and should consider whether bundling small HTTP requests may save energy. Developers should refrain from requesting small amounts of data from distant databases, especially for some RESTful apps. If numerous short HTTP requests must be made, developers should aim to optimize their apps or protocols to combine these into a single, more energy-efficient request.

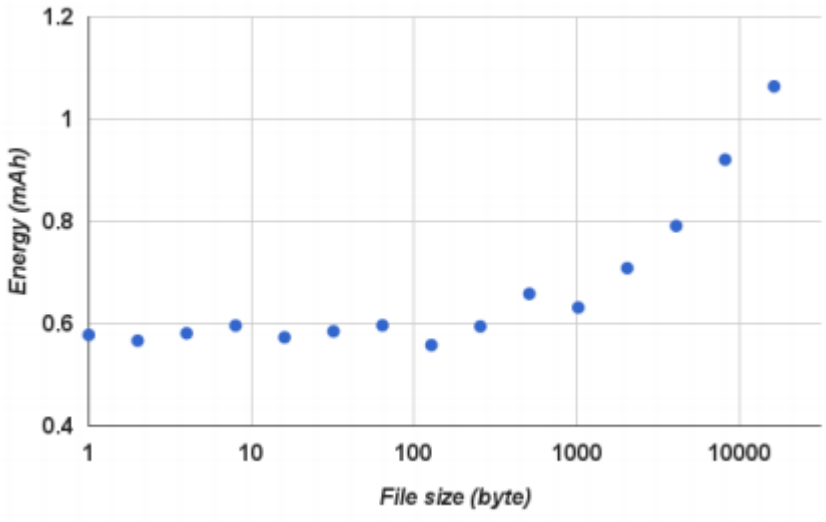


Figure 1. Energy consumption of downloading the different sized data files [1].

PORTUGAL 2020 UNIÃO EUROPEIA Fundos Europeus Estruturais e de Investimento

2) **Use of memory:**

The average energy consumption of accessing different heap-based objects that represent different levels of memory usage is measured to create other recommendations basd on the use of memory. Figure 2 shows the result. X-axis indicates array length. From the figure, we can conclude that the average energy consumption per access to an array cell is raised by increasing the size of memory usage. However, this increase is modest.

The findings of the experiment suggest that although memory is not free and programmers should refrain from allocating superfluous memory, they are not as expensive as previously believed.

The average energy used for each access is very slightly increased as memory utilization grows. This outcome can influence development practices by motivating programmers to allocate additional memory if doing so could help other components use less energy. For instance, programmers may assign more cache space to limit network access.

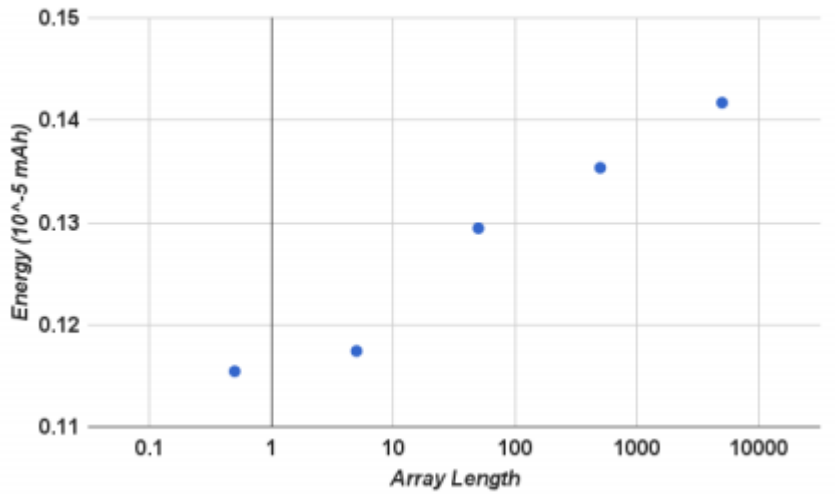


Figure 2. Energy consumption at different levels of memory usage [1].

3) **Performance tips:**

They compared the energy consumption of code that was and was not implemented using best practices oriented towards runtime performance. In particular, they considered three practices as : (1) avoiding access to the length property of an array in the loop's body, (2) going straight to the field instead of utilizing getters and setters, and (3) using static rather than virtual invocations. For each of these practices, they compared the energy usage of the programs after implementing code that either followed or did not follow the recommendation.

From the results, they suggested three recommendations for energy saving as follows.

1) This implies to developers that decreasing the energy consumption of loops may be accomplished by initializing a variable with the length only once and utilizing that as a constant.
2) If developers want to conserve energy, they should access object fields directly rather than through a method.
3) This finding suggests that static invocation is more effective with Android. This is most likely caused by the additional search costs associated with invoking virtual methods. Even while utilizing static methods exclusively is not a smart programming practice, developers should take this into account when trying to conserve energy in code parts that regularly use other

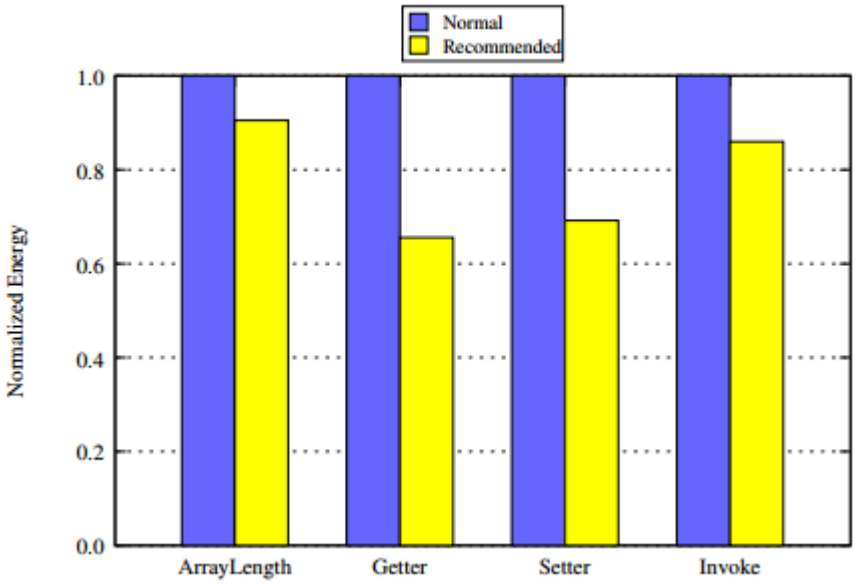


Figure 3. Energy consumption of the performance oriented best practices. [1].

In other research[2], the authors consider the impact of two best practices in terms of energy consumption, including, the use of appropriate syntax and avoiding getters/setters. To analyse the impact of each best practice, the code without the practice is executed to obtain its evaluation results. After that, this code is modified by applying the practice , and its evaluation is performed.

1) In the first experiment, they calculated the energy consumption of different variants of a loop. To have a better understanding, the source codes for the same operation are given in Figure 4. It can be seen that the authors tried to write a function in 3 ways and consume the energy consumption of each

code, separately. The results can be seen in Figure 5. It clearly shows that the way of writing a loop in Android can be effective in energy consumption.

2) In the next experiment, the impact of avoiding getters/setters methods is evaluated. From the experiments, provide in Figure 6, they conclude that, as a recommendation, developing a method without a getter can save energy.

```
public void zero(Foo m[])
 {
  int sum=0;
   for(int i=0;i<m.length;i++)
   {
    sum+=m[i].a;

   }

 }

 public void one(Foo m[])
 {
  int sum=0;
  Foo m1[]=m;
  int len=m1.length;
   for(int i=0;i<len;i++)
   {
    sum+=m1[i].a;

   }

 }

 public void two(Foo m[])
 {
  int sum=0;
   for(Foo m2:m)
   {
    sum+=m2.a;

   }

 }
```

Figure 4-experimental codes for the *for* practice [2]

Method	Result	
	Performance Inc.CPU Time(ms)	Energy(J)
For without length	53.87327	0.00607
For with length	50.16987	0.00549
For-each	47.16917	0.00514

Figure 5-Energy consumption or the results of the *for* practice [2]

Method	Result	
	Performance Inc.CPU Time(ms)	Energy(J)
withGetter	921.4926	0.128
wihoutGetter	309.9727	0.0422

Figure 6- results of avoiding getter and setter methods [2]

[3] shows the best practices for energy consumption in several categories. The patterns the authors listed is independent of the development environment. In other words, they can be used on both iOS and Android. These patterns can be used as recommendations to developers with the goal of decreasing the energy consumption.

- **Dark UI Colors**

A dark UI color theme can save battery life on devices[4].

- **Dynamic Retry Delay**

Some functionalities of the mobile application necessitate gathering information from external sources (e.g., update information from a server). However, under the same situations, if the resource is not accessible, the app will pointlessly attempt to connect to the resource many times, wasting power. Therefore, the developers should increasee the time elapsed between attempts to access the same resource after each unsuccessful one to save energy [3].

- **Avoid Extraneous Work**

Mobile applications must handle several tasks at once. There are situations where the outcome of those activities is not immediately apparent (for instance, when the user interface is displaying other pieces of information) or where the outcome is not always pertinent to the user. This is especially important when using background-running programs. The phone is wasting resources since the data is far out of date. Therefore, the developers should avoid doing activities that are not obvious to the user, provide little value, or will soon become outdated [3].

- **Race-to-idle**

Numerous resources used by mobile apps can be explicitly closed after use. These resources demand additional power consumption while they are active since they are prepared to respond to requests from the app. Therefore, developers, as quickly as they can, should make resources or services available (e.g., wakelocks, screen)[5].

- **Open Only When Necessary**

Some resources must first be opened in order to be used. It could be tempting to access the essential tools as soon as a work is started (like when an activity is created). Resources, however, will be actively awaiting requests that use energy. Therefore, developers should only launch resources or services when absolutely essential[6].

- **Push Over Poll**

Resources must provide updates for mobile applications (e.g., from a server). Regularly querying such resources is one approach to look for updates. However, this will result in several queries that won't return any updates, wasting energy. The recommendation here is, instead of actively searching for resources, to use push notifications to get updates from those resources (i.e., polling)[3].

- **Power saving mode**

Users want to prevent losing data if the device's battery is low. Before they arrive at a power station, they have connectivity. Users who allow the gadget to shut off risk missing crucial calls or being unable to complete crucial tasks. However, in this crucial situation, apps can do meaningless functions that drain the battery. To this end, the recommendation is that the program offers a power saving mode in which it utilizes fewer resources while still offering the user's essential minimal functionality. It may be turned on manually or when certain power events occur (e.g., when the battery reaches a given level)[3].

- **Power Awareness**

There are several features that, while they enhance the user experience, are not necessarily required for users (e.g., UI animations). Additionally, certain tasks do not require rapid execution and may not have a high priority, such as backing up data to the cloud. The suggestion is to, depending to the power state, enable or deactivate certain functions or tasks. It's possible that the battery is going low even while the device is connected to power, thus it could be best to wait until a certain battery level is achieved (or the power save mode is deactivated)[7].

- **Reduce Size**

In mobile apps, data transfer is a typical process. Such activities, however, are energy hogs, and it is best to cut the transmission time as much as feasible. Sending superfluous data should be avoided, only exchange what is absolutely essential. When it is possible, use data compression[8].

- **WiFi over Cellular**

Cellular network data connections typically consume more battery power than WiFi ones. Low priority tasks that involve exchanging large quantities of data through a data connection should be postponed until a WiFi connection is available [9].

- **Suppress Logs**

Developer should avoid intensive logging since it has been revealed that logging activities at rates above energy efficiency are considerably reduced by one message per second [10].

- **Batch Operations**

Tail energy consumption associated with beginning and stopping resources is frequently caused by performing an activity. Therefore, the recommendation is to batch multiple operations, instead of putting the device into an active state many times [1].

- **Decrease rate**

Mobile applications must run operations on a regular basis. The program will execute actions more frequently if there is little delay between two executions. In some circumstances, even if procedures are carried out more often, consumers' perceptions will not change. Therefore, developers should find the shortest time between operations that does not degrade the user experience by increasing the delay between operations. Developers can manually adjust this delay, or consumers can specify it [3].

- **User Knows Best**

Solutions for energy efficiency frequently offer a trade-off between features and power usage. The tradeoff varies depending on the user; some may be OK with less functionality but greater energy efficiency, and vice versa.

Therefore, it is recommended to allow people to personalize their selections for features that save energy. Mobile apps should offer the best options by default for ordinary users as this may be more natural for power users [3].

- **Enough resolution**

High resolutions are alluring while gathering or showing data. The issue with employing high-resolution data is that doing so requires more resources for gathering and manipulation (e.g., memory, processing capacity, etc.). As a result, unnecessary energy use rises[3].

- **Sensor fusion**

Mobile apps offer features that need reading data or executing operations on variou sensors . Such procedures may be energy hogs that use a lot of electricity. Therefore, they ought to be named as seldom as possible. It is recommended to utilize supplemental information from low-power sensors to determine if a specific energy-hungry operation has to be carried out [11].

- **Kill Abnormal Tasks**

There may be activities in mobile apps that are surprisingly energy hogs (e.g., taking a long time to execute). The developers should give energy-hungry tasks or wake locks an appropriate timeout [3].

- **No screen intraction**

There are certain apps that need constant screen use. There are applications, nevertheless, where a less energy-intensive screen can be used instead. To this end, developers should allow users to utilize alternate interfaces to engage with the app (e.g., audio) [3].

- **Avoid using unnecessary animations and graphics**

  Mobile apps frequently include eye-catching animations and visuals. To avoid draining the users' devices' batteries, they must be appropriately calibrated. Therefpre, the developers should look into how important graphics and animations are to the user experience, and they avoid using graphics animations or high-quality graphics. If possible, resort to low frame rates for animations [3].

[1] Li, Ding and Halfond, William GJ, An investigation into energy-saving programming practices for android smartphone app development, Proceedings of the 3rd International Workshop on Green and Sustainable Software, P. 46-53, 2014

[2] S. Mundody and K. Sudarshan, “Evaluating the impact of android best practices on energy consumption,” in IJCA Proceedings on International Conference on Information and Communication Technologies, vol. 8, pp. 1–4, 2014.

[3] L. Cruz and R. Abreu, “Catalog of energy patterns for mobile applications,” Empirical Software Engineering, vol. 24, no. 4, pp. 2209–2235, 2019.

[4] Agolli T, Pollock L, Clause J, Investigating decreasing energy usage in mobile apps via indistinguishable color changes. In: 2017 IEEE/ACM 4th international conference on mobile software engineering and systems (MOBILESoft). IEEE, pp 30–34, 2017

[5] Liu Y, Xu C, Cheung S-C, Terragni V , Understanding and detecting wake lock misuses for android applications. In: Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundations of software engineering (FSE). ACM, pp 396–409, 2016

[6] Banerjee A, Roychoudhury A , Automated re-factoring of android apps to enhance energy-efficiency. In: Proceedings of the international workshop on mobile software engineering and systems. ACM, pp 139–150, 2016

[7] Bao L, Lo D, Xia X, Wang X, Tian C, How android app developers manage power consumption?: an empirical study by mining power management commits. In: Proceedings of the 13th international conference on mining software repositories. ACM, pp 37–48, 2016

[8] Boonkrong S, Dinh PC, Reducing battery consumption of data polling and pushing techniques on android using gzip. In: 2015 7th international conference on information technology and electrical engineering (ICITEE). IEEE, pp 565–570, 2015

[9] Metri G, Agrawal A, Peri R, Shi W , What is eating up battery life on my smartphone: a case study. In: 2012 international conference on energy aware computing. IEEE, pp 1–6, 2012

[10] Chowdhury S, Di Nardo S, Hindle A, Jiang ZMJ , An exploratory study on assessing the energy impact of logging on android applications. Empir Softw Eng 23(3):1422–1456, 2018

[11] Shafer I, Chang ML, Movement detection for power-efficient smartphone wlan localization. In: Proceedings of the 13th ACM international conference on modeling, analysis, and simulation of wireless and mobile systems. ACM, pp 81–90, 2010

[12] Kim D, Jung N, Chon Y, Cha H, Content-centric energy management of mobile displays. IEEE Trans Mob Comput 15(8):1925–1938, 2016

6 Requirements for decision support and management interface, and recommendation channels

Based on analysis and brainstorming sessions and interviews with relevant stakeholders (notably Aptoide), the team has eased a set of requirements for the **Greenstamp** system interface, which takes into account various perspectives and views:

From the users' point of view:

- Promote the energetic labelling of apps when they are developed;
- Calculation of consumption estimates throughout use;
- Confirmation of energy labelling (user feedback in app store);
- Warning (notification) when an application consumes a lot of power;
- Differentiate between "families" of apps (a social network has a different cost than an alarm)
- Analysis by functionality and not by consumption measurement (have consumption measurements tabled by functionality, GPS type, etc.);

From the app store's point of view:

- Show information about the type of applications the user has installed;
- Taking into account the list of installed apps, make an estimate of the total consumption;
- Focus application analytics on main business: Games

From the developers' point of view:

- Classification of application development frameworks ;
- Merit classification of software houses;
- Historical Performance Developer: https://developer.android.com/topic/performance/power/setup-battery-historian
- Make available in the IDE a tool that makes an analysis of inefficient code patterns, suggesting solutions;
- Gamification of the provision of developers;
- Datasets with app characteristics and permissions ==> App Consumption Estimation

From the market's point of view:

- Opening a potential new business area: Energy certification of apps.

7 References

All to fill