Sede Amministrativa: Università degli Studi di Padova

Dipartimento di Matematica

SCUOLA DI DOTTORATO DI RICERCA IN: SCIENZE MATEMATICHE

INDIRIZZO: INFORMATICA

CICLO: XXVIII

# SMARTPHONE AS UBIQUITOUS DEVICES FOR BEHAVIOR ANALYSIS AND BETTER LIFESTYLE PROMOTION

**Direttore della Scuola:** Ch.mo Prof. Pierpaolo Soravia

**Coordinatore d'indirizzo:** Ch.mo Prof. Francesca Rossi

**Supervisore**: Ch.mo Prof. Ombretta Gaggi

**Dottorando**: Matteo Ciman

# Smartphones as ubiquitous devices for behavior analysis and better lifestyle promotion

PhD student: Matteo Ciman          Supervisor: Ombretta Gaggi

Doctoral School in Mathematical Sciences
Computer Science area, XXVIII course
University of Padua, Italy

# Contents

V Conclusions and Future Works

# Abstract

*Ubiquitous Computing*, the third computing era, has taken into people' life a huge amount of invisible and pervasive devices, i.e., smartphones, tablets, sensors, etc. Considering their pervasiveness and their presence in people' everyday life, in the last few years research focused its attention on how it is possible to use smartphones to improve people' life.

In particular, nowadays smartphones are equipped with an ample set of different sensors, i.e., accelerometer, barometer, light sensor, etc., that can constantly acquire data from the surrounding environment of the user, or about his/her activity and behavior during the day. This data acquisition can be done in a non-intrusive way, since smartphones are always in people' pocket or bag and are already part of individuals' life. Thanks to these sensors, a smartphone is able to acquire a huge amount of data, which can be used for different purposes.

For example, the use of *ubiquitous computing* for healthcare and medical purposes is extremely interesting. In particular, it can improve people' life, help patients to accept their therapies and reduce the *drop-out-from-therapy* phenomena, and improve medical practice, thus decreasing medical costs both for patients and for governments.

Actually, *ubiquitous computing* for healthcare is facing and addressing several problems that will be discussed in this thesis. First of all, since there is not a single smartphone vendor (which is an advantage from the end-user point of view), when considering smartphone applications for healthcare it is necessary to consider market fragmentation. This kind of applications aims to reach the highest number of possible users, since increasing the number of possible users increases the number of possible patients that could benefit

from the usage of a particular application. Clearly, the objective of reaching the highest number of users has to consider also the problem of containing expenses. For this particular problem, we will analyze cross-platform frameworks for smartphone applications development, also called “write once, run everywhere” framework”. We will highlight pros and cons of them, considering both the developer and the end user point of view and power consumption, providing a deep analysis of their main characteristics and evaluating them with respect to the native mobile application development approach.

Afterward, we will focus our attention to the possibility of using smartphones to infer individuals’ moods and feelings. In particular, since smartphones are always present in everybody life, it is possible to use them to transparently and pervasively acquire data and information about how the smartphone is used by the user to understand his/her behavior. In this way, we explored the possibility to use smartphone data, without the usage of any external sensor, thus reducing the entry cost and increasing user acceptance, to assess their stress state. In particular, we will see how the way people interact with their smartphone changes depending on their stress condition, opening the doors for the possibility to pervasively monitor stress of people all the day long without interfering with their life and providing the possibility to doctors to monitor their patients with objective data.

Finally, since *ubiquitous computing* can be used to improve people’ life, we explore the usage of *serious games* with mobile applications to improve people lifestyle. We applied this idea to the possibility to “teach” people to take stairs instead of elevators or escalators. In this way the user makes more physical activity among the day, and consequently have fewer problems like obesity, heart attacks, etc. Moreover, *serious games* can be used to help doctors with their diagnosis, in particular when dealing with children, since the standard tests designed for adults can be unlikely applied to children, thus affecting the quality of the diagnosis. This thesis shows how using *serious game* and touch interfaces, the best ones for children, we are able to perform diagnosis and rehabilitation of particular diseases that can be treated better if early diagnosed, providing to the doctor good alternatives to standard exercises and tests.

# Riepilogo

*Ubiquitous computing*, è il nome dato a quella che viene definita la terza "era computazionale", che ha introdotto, nella vita delle persone, un grande numero di dispositivi invisibili e pervasivi, come smartphone, tablet, sensori, ecc. Considerata la loro forte presenza nella vita quotidiana, negli ultimi anni la ricerca si è focalizzata su come utilizzarli per migliorare la qualità della vita delle persone.

Al giorno d'oggi gli smartphone sono equipaggiati con uno svariato numero di sensori, come accelerometro, barometro, sensore di luminosità, ecc., che sono in grado di acquisire costantemente informazioni riguardanti l'ambiente circostante, o determinare il comportamento della persona durante la giornata. Questa acquisizione di dati può avvenire in maniera non intrusiva, in quanto gli smartphone sono sempre più presenti nelle tasche o nelle borse delle persone e sono ormai diventati parte attiva e integrante della loro vita. Le informazioni acquisite possono essere utilizzate per diversi scopi.

Per esempio, particolarmente interessante risulta essere l'uso dell'*ubiquitous computing* in ambito medico per migliorare la qualità di vita delle persone, aiutare i pazienti con le loro terapie e ridurre il fenomeno dell'abbandono di una terapia, e migliorare infine la pratica medica, riducendo quindi le spese sostenute sia da parte dei pazienti che dagli Stati per i propri cittadini.

Attualmente, l'uso dell'*ubiquitous computing* in ambito medico sta affrontando diversi problemi che sono stati presi in considerazione ed analizzati in questa tesi. Per prima cosa, poichè non esiste un unico produttore di smartphone (cosa che da un punto di vista dell'utente finale è un aspetto positivo), quando vengono sviluppate applicazioni

in ambito medico risulta essere particolarmente importante tenere in considerazione la frammentazione del mercato. Questo tipo di applicazioni mirano a raggiungere il maggior numero possibile di potenziali utenti in modo da aumentare anche il numero di pazienti che potrebbero beneficiare dall'utilizzo di tale applicazione. Chiaramente, l'obiettivo di raggiungere il maggior numero possibile di utenti, e quindi di supportare il maggior numero di possibili piattaforme e device, si scontra con il problema di contenere i costi di sviluppo. In questa tesi, vengono analizzati i framework cross-platform per lo sviluppo di applicazioni in ambito mobile, chiamati anche framework "*write once, run everywhere*", sottolineando pro e contro, considerando sia il punto di vista dello sviluppatore che quello dell'utente finale (in particolare per quel che riguarda il consumo di energia), fornendo un'analisi approfondita delle maggiori caratteristiche e comparandoli con l'approccio nativo.

Successivamente, verrà posta attenzione sulla possibilità di utilizzare gli smartphone per inferire lo stato emotivo delle persone. In particolare, è possibile utilizzarle gli smartphone per acquisire in maniera trasparente informazioni sull'interazione con lo smartphone stesso, e come questa sia influenzato dalla condizione psicologica dell'utente. I dati provenienti dallo smartphone e dai suoi sensori, senza l'uso di dispositivi esterni (riducendo quindi il costo iniziale ed aumentando la predisposizione dell'utente a tale sistema di monitoraggio), vengono successivamente analizzati per inferire la condizione di stress/no stress dell'utente. In particolare, la tesi dimostra come il modo in cui le persone interagiscono con il loro smartphone è influenzato dal loro livello di stress, ed è quindi possibile monitorare lo stress in maniera pervasiva durante la vita di tutti i giorni, senza interferire con essa e fornendo la possibilità ai medici di monitorare costantemente i loro pazienti con dati oggettivi.

Infine, poichè l'*ubiquitous computing* può essere utilizzato per migliorare la vita delle persone, la tesi analizza l'uso dei *serious games* per migliorare il loro lo stile di vita. In particolare, viene sfruttata questa idea per insegnare alle persone ad utilizzare le scale al posto di ascensori o scale mobili. In questo modo, si aumenta l'attività fisica durante la giornata, adottando quindi uno stile di vita più attivo, riducendo così la probabilità di avere problemi quali obesità, attacchi di cuore, ecc. Inoltre, i *serious games* possono essere utilizzati anche per aiutare i dottori con le loro diagnosi, in particolare quando gli utenti finali sono bambini, in quanto gli esercizi standard attualmente utilizzati sono stati definiti per gli adulti e possono risultare complicati per i bambini, influenzando quindi in maniera negativa la qualità della diagnosi. Questa tesi mostra come utilizzando i *serious games* e le interfacce touch, le migliori per i bambini, siamo in grado di effettuare una diagnosi ed una riabilitazione per particolari patologie che possono essere trattate con maggiore efficacia se diagnosticate precocemente, fornendo al medico curante delle valide alternative agli esercizi utilizzati attualmente.

# Introduction

# 1. Introduction

## 1.1 Ubiquitous computing for health

It was 1988 at Xerox PARC when Mark Weiser coined the term of *ubiquitous computing*. His vision was that technology, with a particular focus on computing technology, could become part of people everyday life, supporting all the possible activities that are part of people routine, e.g., work, home, sport, etc. Weiser's idea can be summarized in the following sentence, proposed in his famous paper "The computer of the 21st century" [139]:

" *The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.* "

Together with the idea of disappearing technology, Weiser introduced even the idea of *calm technology* [140]. The basic idea behind *calm technology* is to not consider the interaction with technology and devices as the principal focus of our attention, but as something that it is only in the person's periphery. In this way, technology and its benefits would become something that actively belong to people' life and do not require a big amount of attention.

Modern computing is based on three different eras, as shown in Figure 1.1. Mainframe computers defined the first era, where many people used one single available computer at the same time. The second is the era of PCs, where we have personal computers owned and used by a single person. The third era, the *ubiquitous computing* era, corresponds to the explosion of small portable devices that can be, but not limited to, our *smartphones*.

The result of this diffusion of mobile devices is that people owns and use many computers (even of small dimensions).

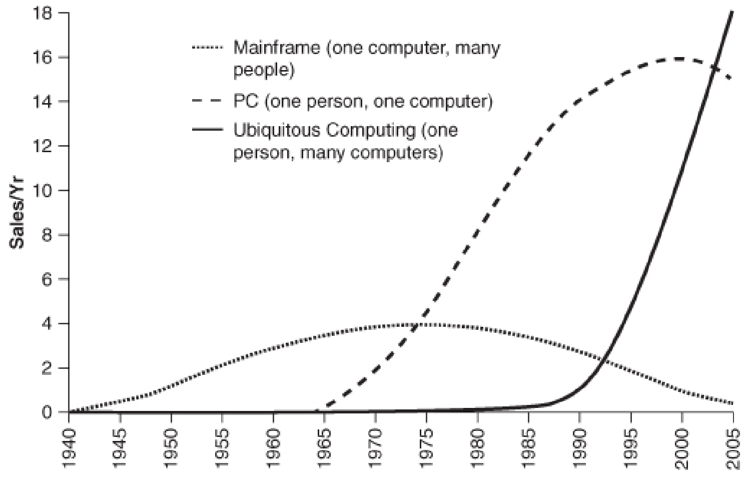
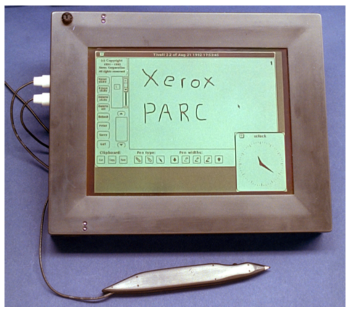


Figure 1.1: The three eras of modern computing.

As we can notice, the evolution of the different eras corresponds to a world where technology is progressively more integrated into everyday life and even less invasive.

When Mark Weiser proposed the idea of *Ubiquitous Computing*, he even made it concrete, in collaboration with Xerox PARC, developing ubiquitous devices [141]. The result of this research were three different devices: the ParcTab (Figure 1.2a), the ParcPad (Figure 1.2b) and the Liveboard (Figure 1.2c). Due to its lower portability properties, we will not discuss in details about Liveboard, which simply provided the functionality of a whiteboard with some more interactivity possibilities.

(a) The Xerox ParcTab.


(b) The Xerox ParcPad.

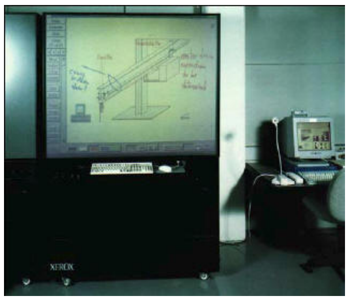

(c) The Xerox Liveboard.

Figure 1.2: Xerox devices

The ParcTab was a palm computer communicating using infrared signals, while the ParcPad was a notebook-sized tablet computer, which used a low bandwidth X-protocol across a radio link, talking with a base station through a short-range near-field radio. The idea behind these devices (and even of Liveboards) was to replace corresponding objects in the workplace with something with similar physical characteristics, but with enhanced technological capabilities, offering something more to the users. For example, the idea of the ParcPad was to replace a single book with the possibility to store several books delivered across the network, i.e., a sort of modern Amazon Kindle, and providing the opportunity to markup this electronic books with a pen interface, insert hyperlinks, word definition and cross-reference with other material. They followed the same approach with the ParcTab, since it served as a simple Personal Information Manager but, thanks to its infrared connection, it supported one of the first pocket email readers, plus the possibility to edit documents or to work as a controller of a room's heating and air-conditioning.

ParcTab and ParcPad can be considered the fathers of the nowadays smartphones and tablets respectively. Thanks to the technological evolution and the presence of sensors on these devices and higher computational capabilities, *ubiquitous computing* has now the possibility to increase its power and influence in individuals' everyday life. Indeed, smartphone sensors let acquire data and information about the environment where the user is, making applications able to understand where they are being used and adapt their operations to provide the best possible user experience.

*Ubiquitous computing* has a wide range of possible context where it could be used, from advertising [87] to entertainment [28], from environment monitoring [23] to healthcare[85].

In this thesis, we want to focus our attention on the application of *ubiquitous computing* to healthcare, with a particular interest in people well-being. As mentioned before, the possibility of using smartphones to capture information about the environment of the user or his/her activity has opened the doors to a lot of interesting applications. For example, it is possible to monitor people' exercise to understand how much physically active they are among the day. More complex is the topic of behavior change, where we use *ubiquitous computing* not only to understand how people behave but even to suggest them some corrections to follow a better lifestyle, in a non-intrusive way with the best persuasive and pervasive approach.

From a doctor perspective, *ubiquitous computing* can powerfully help physicians and patients to move from therapy or rehabilitation made only at the hospital, to home rehabilitation, much more well accepted by patients [13]. In this way, it is possible, for example, to monitor people with a particular disease in their environment, to understand how they behave, or if a particular therapy is effective or not.

As it is possible to see, *ubiquitous computing* has the possibility to revolutionize healthcare, both from a patient and a doctor perspective. It has the power to provide all the technological requirements for systems that are much less intrusive, and that become invisible in individuals' life, and that can help patients and doctors in a strongly practical way.

In this thesis, we will tackle different problems. The first one is related to smartphone applications, and in particular, how it is possible to manage the problem of smartphone market fragmentation without increasing development costs. The second problem we will consider is related to mood analysis and inference using smartphones. In this way, we can pervasively infer people' feelings only using a smartphone and without any external sensors, thus being able to monitor individuals' emotions without interfering with their life. Finally, we will show how *serious games* and *ubiquitous computing* can be used together to improve people' behavior with healthier habits, or to help doctors with therapies and diagnosis, especially when dealing with children.

Section 1.2 and Section 1.3 will explain the current research problems and contributions, divided into tree main research topics.

## 1.2 Problem Statement

### 1.2.1 Cross-platform application development

One of the biggest problem that it is necessary to consider when dealing with mobile applications is market fragmentation, since there exist different smartphone vendors, operating systems, and smartphone models. Let us consider for example an application for behavior change or health monitoring. There are two fundamentals requirements for this kind of applications:

- to reach the highest number of potential users, and
- to reduce as much as possible the entry "cost" for the users.

Therefore, two problems arise when the application is developed only for a single mobile platform. The first one is to impose to the user to buy a particular device if he/she does not already have it, i.e., he/she must spend money, thus increasing the entry "cost" for the user. The second one is that if an application is developed only for a specific platform, it drops out the users that do not have that specific one, drastically decreasing the number of potential users that could change their behavior or follow a particular therapy/monitoring.

One approach that is promising in the last years to reduce development costs and time and to reach the highest number of mobile devices (and so, users) is the use of cross-platform frameworks. The idea of these frameworks is to let the developer write only one single application, in the framework specific language, and deploy this application to the highest number of mobile platforms, depending on the platforms supported. The first research question which this thesis tries to answer is how much these frameworks are efficient.

To the best of our knowledge, a profound and complete analysis of these frameworks has never been provided. First of all, we analyzed which are the main features offered by these frameworks and how much they are easy to use. When we talk about features, we mainly refer to the completeness of the frameworks concerning the APIs that are usually available when developing applications using a native approach. When we talk about the easiness of use, we refer to the learning curve of the framework, that depends on the provided documentation, tutorials, support, etc.

Moreover, a framework can be evaluated even with objective data, in particular for one of the most important aspects when considering applications for smartphone, which is energy consumption. Current analysis available for cross-platform frameworks do not consider this critical issue, and in particular, if, and in which measure, the use of a cross-platform framework influences the energy consumption of the deployed application with respect to the native application developed from scratch.

Thanks to these evaluations, the developer can decide if the cross-platform approach is a good solution for a particular application, or if there are still some situations and specific smartphone applications where the native approach remains the best one. With our analysis, we will show that currently frameworks have several weaknesses that negatively influences both the development of the application and their performances, even if they have some advantages that could let use them in particular circumstances.

### 1.2.2 Ubiquitous smartphone computing for behavior analysis

Nowadays smartphones daily used by people are equipped with an ample set of sensors, i.e., accelerometer, luminosity sensor, barometer, etc., that can constantly acquire data that can become extremely useful and interesting. For example, it is possible to infer user activity using data from the accelerometer, e.g., if he/she is walking, driving, running, etc., or to understand and capture information about the environment surrounding the user.

The possibility to use only smartphone sensors data to infer information about the environment or the user activity clearly reduces the intrusiveness of a system or application that needs this information, e.g., an application for activity recognition or behavior change. Nowadays, some applications already analyze this data to infer user activity among the day. But these solutions has two different problems. The first one, less important but with several drawbacks, is related to the fact that these applications simply acquire data from the smartphone and rely on a server that, offline or at the end of the day, analyzes this data to provides an evaluation of user activity. Clearly, this approach has several drawbacks, since if there is no connection available or the server is overloaded, it is not possible to immediately provide an analysis of the data. Moreover, data transmitted to the server can be high and this could become a problem if the application sends too many data over the Internet. The second problem relates to energy consumption. Indeed, current applications acquire data from a high number of sensors and at high frequency, thus increasing energy consumption and reducing user experience.

In fact, the current challenge of activity recognition with a smartphone is real-time recognition, i.e., to provide immediate feedback to the user, and to analyze data directly on the smartphone, without relying on an external server. But this solution brings one big problem: energy consumption. Current solutions that aim at recognizing activities working in real-time on smartphones do not consider the issue of the limited capacity of the battery available with smartphones. Most of the presented solutions acquire data from the highest possible number of sensors, at high frequency, and try to reach the best accuracy of activity recognition. What is currently missing and not already investigated is how the usage of sensors influences energy consumption. In fact, to collect data from sensors at a particular frequency clearly has a cost for the battery of the smartphone. If data from a sensor is acquired, the energy consumption increases both for data acquisition and even for data analysis, since the smartphone has to analyze more data before it can provide a classification answer. Current research lacks of the absence of a deep analysis of how data acquisition influences activity recognition and energy consumption and how to reach the best trade-off between energy consumption and precision. With our *serious game*, called *ClimbTheWorld*, we will combine these two aspects to find the best solution for activity recognition and user experience.

Moreover, not only smartphone sensors can provide data about the user, but the smartphone itself can provide data, especially about the interaction between the user and the smartphone. Applications usually consider data about location, the number of messages, emails or calls. This data is acquired to understand if there is a change in people behavior depending for example on their mood, but unfortunately, these data are considered very intrusive. Clearly, it is possible to make a step further, that means analyzing how people interact with their smartphone. In this case, when we talk about interaction we consider for example how strong they touch the screen, how they 'scroll' or 'swipe' on the screen, etc. This interaction analysis is clearly less invasive and does not consider privacy related information, and can introduce the possibility to infer individuals' mood simply studying their interaction with the smartphone and, more important, without being too much intrusive and acquiring privacy related information. As we will see later, we will present our solution to assess stress in users building a classifier that uses data about 'tap', 'swipe', 'scroll' gestures and showing that there is a statistical difference in how people write text with smartphones between when they are stressed or not.

### 1.2.3 Power of *serious games* on people life

Another challenge that *ubiquitous computing* is currently facing is not only related to understanding user behavior or moods, but even how to use the information we collect. In particular, in *pervasive health* two are the most important ways of using data from the smartphone. The first one is from a medical perspective, for example, to monitor people during the day in a free environment and not in the laboratory, so improving the quality of life, since they are not forced to stay in a laboratory or an hospital. The second one is from a user point of view, with the possibility to help him/her to change his/her behavior/lifestyle with a better one.

In particular, one of the biggest problem to deal with is which is the best solution and approach for behavior change. It is clear that simply providing to the user statistics about his/her activity among the day is not sufficient to incentivize him/her into behavior change, i.e., make more physical activity among the day. For this reason, *serious games* and *gamification* techniques have been introduced in the last years to try to transform serious activities (or healthier behavior) into something funnier, making it easier and people more likely to follow a new behavior or perform a particular activity. The problem that arises with *serious games* and *gamification* applications is that it is currently not demonstrated that they are effective in every field, so a deep investigation is still undergoing and new evidence of how these approaches work in behavior change are required. We must note here that simply showing that users appreciate the proposed game or task we ask to perform is not sufficient, but is necessary even to demonstrate that the behavior of people effectively changed with respect to the previous one we wanted to correct. With our application *ClimbTheWorld*, we will show how we were able both to engage people in the game we proposed, and even to change their behavior, incentivizing them in taking stairs, and so making more physical activity during the day.

Moreover, the power of *serious games* should be considered and used even for medical purposes, especially when dealing with children, that cannot simply be considered as young adults. Children have different needs and a different approach, even in the medical field, is required. It is clear that, for a medical diagnosis, if we ask to children to perform a game or a funny exercise instead of the standard tests designed for adults, we can gain much more attention from them, and in this way it is possible to have their attention for much more time, giving to the doctor more time to perform a diagnosis. Even in this case, it is not sufficient to simply analyze if the child has or not fun while playing the game, but it is necessary even to show that using *serious games* we can perform (or we give the possibility to the doctor to perform) a medical diagnosis or to get medical data that is the same or event better than the one we would get if, instead of the *serious game*, the child would have done the standard tests. With our *serious game* called *PlayWithEyes* that we will present later in this thesis, we will show how we are able to obtain a medical diagnosis as the one a doctor would get using standard visual acuity tests, but, from a children perspective, in a much more appreciated way.

## 1.3 Research Contributions

The contribution claimed in this thesis are several. For simplicity, we will divide them into the three main parts of the thesis and research issues.

### Cross-platform frameworks

Since we can divide frameworks into different categories depending on the developing process and the final application, we compared these different categories both together and with respect to the native approach. In particular, our research focused on two different aspects. From a developer point of view, we considered which are the main features offered by each framework, with respect to the offered APIs, learning difficulty, platform supported, etc. The second aspect we considered in our analysis is much more related to the user experience, and is energy consumption. It is clear that, when developing applications for mobile devices, energy consumption is crucial. Moreover, for pervasive applications that aim at being as less intrusive as possible, having an application that wastes energy and asks the user to charge his/her smartphone many times among the day, is something not practicable and to avoid. For this reason, we measured the consumption of basic applications that acquire data from smartphone sensors, and we compared their consumption with respect to the same application developed as a native solution.

Our analyses show that:

- First of all, actual cross-platform frameworks are not ready for a strong adoption in smartphone applications, since their energy consumption is higher with respect to the native approach. The higher energy consumption is true both for the iOS and the Android platforms, the two most adopted mobile platforms, and
- We showed how there are differences in energy consumption between frameworks depending on the way they build the final application.

To the best of our knowledge, this is the first time these frameworks are analyzed from an energy consumption point of view and compared with respect to the native approach. In this way, it is easy to show how the current state-of-the-art frameworks are not ready for pervasive applications, and the native solution is often the best choice, especially if these applications have to acquire data from smartphone sensors or to update the User Interface.

### Emotion assessment using smartphones

Since smartphone are present in the everyday life of each person, we decided to analyze human-smartphone interaction to understand user emotions, focusing out attention to stress assessment. In particular, every time we pick up our smartphone to make a call, write a message/email or to navigate through web pages, we interact with the phone through 'taps', 'scrolls', 'swipes', etc. These interactions hold a lot of information, since it is possible to extract data like pressure, timing, touch size, etc.

Our novel approach uses this information to understand if there is a difference between these interactions when people are relaxed and when instead they are stressed. From our results, it is possible to see how some features extracted from this interactions have a statistical difference between relaxed and stressed state. Moreover, this data allows to build a classifier able to understand in real time if the user is currently stressed.

This approach is novel from two points of view. The first one is that it is the first time that human-smartphone interaction is used to assess stress state in users. Secondly, but not less important, it uses no privacy-related information. Several past works, as we will discuss in the following chapters, used information like calls, messages, locations, etc. Considering only the interaction with the smartphone, it is clear that this kind of privacy "violation" is avoided and user acceptance is much higher.

#### *Serious games* for medical purposes

Since simply gaining information about user activity or mood is clearly not sufficient to improve people life, we explored the usage of *serious games* to induce behavior change in people. In particular, we created a *serious games* to incentivize people in taking stairs instead of elevators and escalators. The idea to use only smartphone sensors and not other external sensors aims at reducing the intrusiveness of the system and increasing user acceptance, since smartphones are present in the everyday life of people.

The first step of our work was to understand and gain information about user activity, in particular how to recognize each single stairstep and differentiate them from normal steps, e.g., walking steps, in order to count and give points only when the user takes stairs. This first step is clearly crucial, and we considered together system classification precision and energy consumption, lowering as much as possible the data acquisition frequency and the number of used sensors to avoid to waste energy without losing precision in stairstep recognition. To the best of our knowledge, this is the first time energy consumption is considered when dealing with activity recognition, to find the best trade-off between system precision and energy consumption.

From the user point of view, we developed a *serious game* where we ask users to climb real building using stairsteps made in real life. This *serious game*, that aims at induce behavior change and promote a better lifestyle making people choose the more active solution, e.g., taking stairs, provides two different game modes, one single mode, and three different multiplayer modes, and we tested it with a group of 13 participants. At the end of our experiment, using data from questionnaire and from a logger installed on the application, we were able to show how the game was appreciated by participants and, more important, to induce a behavior change, since we measured an increase in the number of stairsteps made before and after the usage of the game.

Finally, we showed the power of *serious games* with *PlayWithEyes*, a *serious game* for the early diagnosis of amblyopia in children. In particular, we used this *serious game* to perform a screening test for daltonism and visual acuity in kindergarten children, introducing a new way to test and diagnose visual problems, substituting the standard boring exercises used by doctors. The game is more engaging for children and able to keep them more concentrated among all the tests, avoiding poor attention and so not precise answers. The game is based on the usage of an iPad and an iPod that the child uses to provide answers to the questions, which are designed to make a visual acuity test without that the child knows that in reality he/she is currently performing a screening test. From our experiments and results, we show how with this approach we were able to identify children with visual acuity problems, which were already identified with standard doctor tests. In this way, it is clear that it is possible to perform and early diagnosis of this problem and consequently immediately start therapy, that is critical to reduce any possible future problems.

## 1.4 List of Publications

The work presented in this Thesis has been developed via a collaboration with Ombretta Gaggi, Fabio Aiolli, Claudio Enrico Palazzi and Katarzyna Wac. Part of the research developed during the PhD program has been published in international conferences and workshops, and peer reviewed journals, as described below.

### Journals

**[J2]** O. Gaggi, M. Ciman, “The use of games to help children eyes testing”, In *Multimedia Tools and Applications*, *To Appear* (Accepted December 28th, 2014)

**[J1]** O. Gaggi, C.E. Palazzi, M. Ciman, G. Galiazzo, S. Franceschini, M. Ruffino, S. Gori, A. Facoetti, “Serious games for Early Detection of Developmental Dyslexia”, In *Computers in Entertainment*, *To Appear* (Accepted March 13rd, 2014)

### Conferences & Workshops

**[C12]** A. Bujari, M. Ciman, O. Gaggi, C.E. Palazzi, “Can a Game improve People’s Lives? The case of *Serious Games*”, In *Proceedings of the EAI International Conference on Smart Objects and Technologies for Social Good (GOODTECHS 2015)*, Rome, Italy, October 26, 2015.

**[C11]** K. Wac, M. Gustarini, J. Marchanoff, M.A. Fanourakis, C. Tsiourti, M. Ciman, J. Hausmann, G. Pinar Loriente, “mQoL: Experiences of the ‘Mobile Communications and Computing for Quality of Life’ Living Lab”, In *Proceedings of the 17th IEEE International Conference on e-Health Networking, Applications and Services (HealthCom 15)*, Boston, USA, October 14-17, 2015.

**[C10]** A. Bujari, M. Ciman, O. Gaggi, G. Marfia, C.E. Palazzi: “PathS: Enhancing Geographical Maps with Environmental Sensed Data”, In *Proceedings of the 2015 Workshop on Pervasive Wireless Healthcare (MobileHealth’15)*, pp. 13-16, Hangzhou, China, June 22, 2015.

**[C9]** M. Ciman, Y. Formaggio, O. Gaggi, M. Regazzo, “May SmartPhones Help to maintain Audience Attention during Presentations?”, In *Proceedings of the 11th International Conference on Web Information Systems and Technologies (WEBIST 2015)*, pp. 55-63, Lisbon, Portugal, May 20-22, 2015.

**[C8]** M. Ciman, K. Wac, O. Gaggi, “iSenseStress: Assessing stress through human-smartphone interaction analysis”, In *Proceedings of the 9th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth 2015)*, Istanbul, Turkey, May 20-23, 2015 **[Acceptance rate: 30%]**.

**[C7]** F. Aiolli, M. Ciman, M. Donini, O. Gaggi, “*ClimbTheWorld*: Real-time stairstep counting to increase physical activity”, In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (Mobiquitous 14)*, pp. 218-227, London, Great Britain, December 2-5, 2014 **[Acceptance rate: 19,3%]**.

**[C6]** F. Aiolli, M. Ciman, M. Donini, O. Gaggi, “A Serious Game to Persuade People to Use Stairs”, In *Proceedings of the 9th International Conference on Persuasive Technology (Persuasive 2014)*, Padua, Italy, May 21-23, 2014.

**[C5]** M. Ciman, O. Gaggi, “Evaluating impact of cross-platform frameworks in energy consumption of mobile applications”, In *Proceedings of the 10th International Conference on Web Information Systems and Technologies (WEBIST 2014)*, pp. 423-431, Barcelona, Spain, April 3-5, 2014 **[Best Student Paper Candidate]**.

**[C4]** M. Ciman, O. Gaggi, N. Gonzo, “Cross-Platform Mobile Development: A Study on Apps with Animations”, In *Proceedings of the ACM Symposium on Applied Computing (SAC14)*, pp. 757-759, Gyeongju, South Korea, March 24-28, 2014.

**[C3]** M. Ciman, O. Gaggi, “From a Physical System to a Pervasive Solution to Increase People Physical Activity: Is It Possible?”. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME 2013) - 9th International Workshop on Networking Issues in Multimedia Entertainment (NIME 2013)*, pp. 1-6, San Josè, California, USA, July 15-19, 2013 **[Best Paper Award]**.

**[C2]** M. Ciman, O. Gaggi, L. Nota L. Pinello, N. Riparelli, T. M. Sgaramella, “HelpMe!: a Serious Game for Rehabilitation of Children affected by CVI”, In *Proceedings of the 9th International Conference on Web Systems and Technologies (WEBIST 2013)*, pp. 257-262, Aachen, Germany, May 8-10, 2013.

**[C1]** M. Ciman, O. Gaggi, M. Sbrignadello, “Toward the creation of a Green Content

Management System", In *Proceedings of the 7th International Conference on Web Systems and Technologies (WEBIST 2011)*, pp. 408-412, Noordwijkerhout, The Netherlands, May 6-9, 2011.

### Submitted

**[J3]** M. Ciman, O. Gaggi, "Energy Consumption of Cross-platform Frameworks for Mobile Development", submitted to *IEEE Transactions on Mobile Computing* (First Submission January 27th, 2015, currently under second revision)

**[J4]** F. Aiolli, M. Ciman, M. Donini, O. Gaggi, "Increasing People Physical Activity using a Mobile Application" submitted to *Elsevier Pervasive and Mobile Computing* (First Submission August 24th, 2015)

## 1.5 Structure of the Thesis

This section describes as this thesis is structured.

### Cross-platform frameworks

We start by considering cross-platform frameworks. In particular, in Chapter 2, we provide a description of why these frameworks could become extremely important in the future and we report a classification of them depending on the approach they use to produce the final application. Afterward, in Chapter 3, we analyze these frameworks from a developer point of view, considering developing aspects like the programming language, the IDE, the entry level difficulty, the documentation, the available APIs, etc. In Chapter 4, cross-platform frameworks are analyzed from an energy consumption point of view. In particular, we consider data acquisition from sensors available on the smartphone, Android and Apple devices, and we measure differences with respect to the native approach (where a native application is developed using the native development language and tools) highlighting how, at the time of writing, cross-platform frameworks have negative impact on battery and energy consumption.

### Emotions assessment using smartphones

In this part of the thesis we analyze the possibility to use smartphones to sense, without the usage of external devices attached to the body of the user, user's emotions. In particular, in Chapter 5, a novel solution for stress sensing using human-smartphone interaction is presented and described. The idea of this solution is to analyze user interaction with the smartphone, i.e., 'scroll', 'swipe', 'tap', etc., to understand if he/she is stressed while using it among the day. In this way, a background process can be installed on the smartphone to monitor this interaction and understand when the user is stressed. This makes possible to gain information about users' mood without any external device, without acquiring privacy-related information. This approach clearly increases user acceptance in case of a constant monitoring of stress levels of a patient.

### *Serious games* for medical purposes

Since gaining information about user's mood or physical activity is not sufficient to help him/her changing his/her behavior for a better lifestyle, the third part of the thesis explores how it is possible to incentivize people to change their behavior and with which results, especially considering the *serious games* paradigm. In particular, after an initial discussion about *serious games* for behavior change presented in Chapter 6, Chapter 7 presents the design and implementation of a Android smartphone application called *ClimbTheWorld*, with the goal of incentivizing people in taking stairs instead of escalators or elevators. In particular, simply showing the number of stairsteps made by the user is not sufficient for behavior change, we use the *serious games* paradigm introducing single player and multiplayer games to engage users more and have more “power” in their behavior change. The result of a test with real users shows how our design and the following implementation was effective in behavior change and life quality improvement. Afterwards, in Chapter 8 we present a *serious game* for the early diagnosis of amblyopia and daltonism. The idea of the *serious game* is to transform the static doctor’s tests made to diagnosis this problem into a funny moment for children. In this way, if children have fun, tests can last more and even be used in kindergarten to acquire data that after will be analyzed by a doctor. Clearly, the longer a test lasts, the more accurate will be the diagnosis. Tests with 65 children of a kindergarten of different ages showed how, with this approach, it is possible to obtain results equals to the ones obtained by doctors with standard tests, but in a much easier and funny way for children.

# II Cross-platform frameworks

# 2. Cross-platform frameworks overview

One of the problems of mobile applications development, strongly used by *ubiquitous computing* due to the natural pervasiveness of mobile devices, for example for continuous patient monitoring or to support behavior change, is related to market fragmentation. It is clear that excluding a set of users depending only on their mobile operating system must be avoided, but on the other side, it is even true that economic and time costs drastically increase if it is necessary to develop one different application for each mobile operating system. For this reason, the idea of cross-platform frameworks is to let the developer develop only one single application in the framework specific language, and to deploy it to all the supported mobile systems. In this way, it is possible to avoid to discard possible users depending on their mobile operating system, and even time and economic costs can be limited. Unfortunately, all these frameworks have clearly some limitations since, depending on how they are developed, they can limit the number of native features available or have limited support to mobile systems.

In the following chapters, we will start with an overview of the different approaches used by cross-platform frameworks (see Section 2.1). In particular, we will see how these frameworks can be categorized into four different categories, depending on the technologies used to develop the mobile application and on how much the architecture of the final deployed application is similar (or not) to the one of a native application. This classification will be useful to understand the results that will follow, especially in Chapter 4. The next two chapters, Chapter 3 and Chapter 4, will analyze these frameworks, considering in particular two different point of views. The first one (Chapter 3) considers the developer point of view, e.g., how it is easy to develop an application using a particular framework or following a particular approach. The second one (Chapter 4) contains a more objective analysis, that will consider energy consumption issues. In particular, we will discuss how the adoption of a particular cross-platform framework, that follow a particular approach, as explained in Section 2.1, influences, and in which measure, energy consumption.

To the best of our knowledge, this is the first tentative for such a deep analysis of the current state-of-the-art frameworks and evaluation, in particular from a final user perspective, evaluating energy consumption issues that strongly influence user acceptance of smartphone applications.

Part of this chapter has been published in [30] [31].

## 2.1 Cross-platform frameworks classification

Following the classification provided in [120], cross-platform frameworks can be divided into four different classes: the *Web Approach*, the *Hybrid Approach*, the *Interpreted Approach* and the *Cross-Compiled Approach*. This classification is very useful especially to understand results of energy consumption experiments, since belonging to a particular class deeply influences the performances of the deployed application.

The *Web Approach (WA)* consists of developing a web application, using HTML, CSS and Javascript, which can be accessed through a browser installed on the smartphone. In fact, in this case, the final result is not an application installed on the smartphone, but a web application available on the Internet.

The new emerging features of HTML5 for mobile devices and CSS3 allow the creation of rich and complex applications, therefore this choice cannot be considered a limitation. For example, web applications can access device sensors, i.e., accelerometer, gyroscope, etc., or user and device specific information, i.e., contacts.

We must note here that device support for advanced features depends on the browser maturity, i.e., the adoption of recent features must be carefully considered since it could reduce the set of possible users. Another aspect to consider is what is the preferred browser of the user since different browsers have different implementations of the W3C Recommendations, e.g., accelerometer frequency update. The developer has to take into account these differences when developing web applications.

Moreover, it is almost impossible to provide a look and feel of the final web application equal to the native application (whatever it means, according to any possible device), and this could be a problem for the final user. Figure 2.1 shows this approach. *jQuery Mobile*[1]

[1]jQuery Mobile: http://jquerymobile.com

and *Sencha Touch*[2] are examples of frameworks that follow this approach.

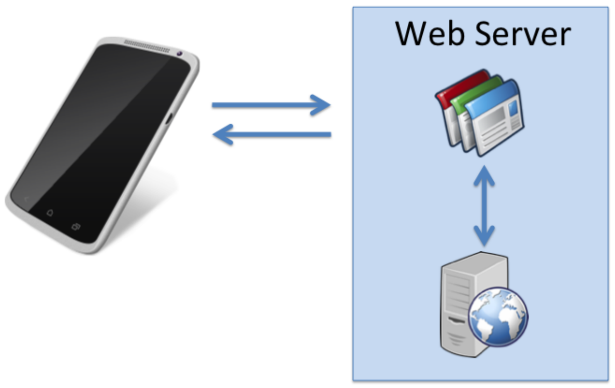


Figure 2.1: Architecture of an application using the WA.

The *Hybrid Approach (HA)* allows developers to create applications which use the webkit rendering engine to display controls, buttons and animations. The webkit engine is also responsible to draw and manage User Interface objects. The framework itself instead provides APIs and access to device hardware and features.

In this case, the application can be distributed and installed and appears on the user device as native mobile application, but the performances are often lower than the native solution, since its execution requires to run the browser rendering engine. Even in this case, to create applications with a native look and feel, even if more easier with respect to the *Web Approach*, still remains not trivial. Figure 2.2 shows the architecture of the deployed application using this approach. An example of this class of frameworks is PhoneGap, also known as Apache Cordova[3].

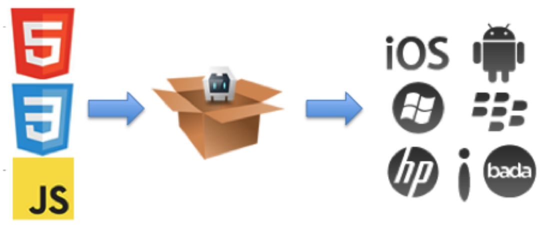


Figure 2.2: *Hybrid approach* application architecture.

The third class, called the *Interpreted Approach (InA)*, gives developers the possibility to write the code of the application using a language which is different from languages natively supported by the different platforms, e.g., Javascript. This approach allows developers to use languages they already know and to learn how to use the APIs provided by each framework. The application installed on the device contains also a dedicated

[2]Sencha Touch: http://www.sencha.com/products/touch/
[3]Apache PhoneGap: http://phonegap.com/

interpreter which is used to execute the non-native code. Even this approach provides access to device features (how many and which ones depend on the chosen framework) through an abstraction layer, and this negatively influences performances.

This approach allows the developer to design a final User Interface that is identical to the native User Interface without any additional line of code. This kind of frameworks allows an high level of reusable code. Titanium[4] is an example of this class of frameworks and Figure 2.3 shows the architecture of the framework.

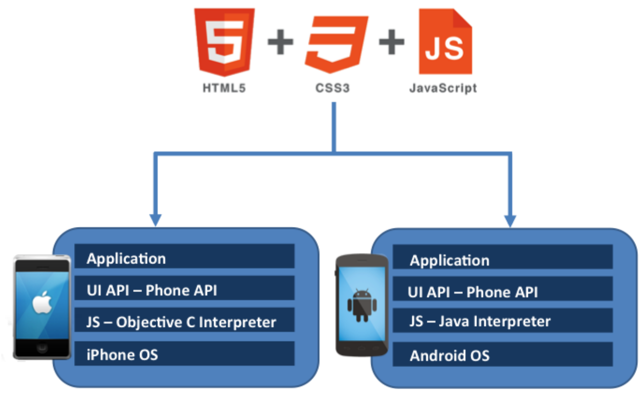


Figure 2.3: *Interpreted Approach* architecture.

Finally, the *Cross-Compiled Approach (CCA)* is similar to the previous one since it lets the developer write only one application using only a specific programming language, e.g., C#, but the final application does not contain an interpreter, but the framework generates, after compilation, different native applications for each different mobile platform. The final application uses native language, therefore can be considered a native mobile application to all intents and purposes. Examples of this framework are Mono[5] or MoSync[6], and Figure 2.4 depicts how this class of frameworks works.

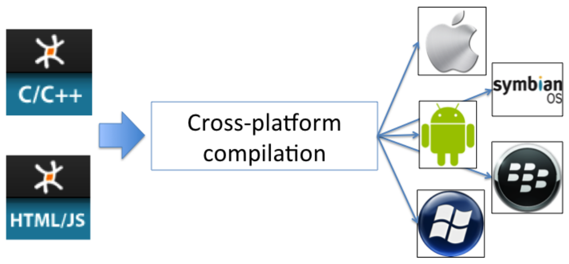


Figure 2.4: *Cross-Compiled Approach.*

Table 2.1 provides a final resume of this classification, highlighting *pros* and *cons* of each approach.

[4]Titanium Appcelerator: http://www.appcelerator.com/titanium/
[5]Mono: http://www.mono-project.com/
[6]MoSync: http://www.mosync.com/

| Approach | Programming Language | Supported Platforms | Pros | Cons | Example |
|---|---|---|---|---|---|
| *Web* | HTML, CSS, Javascript | Android, iOS, Windows, BlackBerry (*) | - Easy to update<br>- No installation<br>- Same UI over different devices | - No access to Application Store<br>- Network delays<br>- Expensive testing<br>- No native UI | jQuery mobile, Sencha Touch |
| *Hybrid* | HTML, CSS, Javascript | Android, iOS, Windows, Blackberry, | - Access to Application Store (**)<br>- Support to most smartphone hardware | - No native Look and Feel | PhoneGap |
| *Interpreted* | Javascript | Android, iOS, Blackberry | - Access to Application Store<br>- Native Look and Feel | - Platform branching<br>- Interpretation step | Titanium |
| *Cross-Compiled* | C#, C++, Javascript | Android, iOS, Symbian | - Native UI<br>- Real native application | - UI non reusable<br>- High code conversion complexity for complex applications | Mono, MoSync |

Table 2.1: Resume of different cross platform approaches. (*): Supports depends on the browser chosen by the user. (**) Apple Store usually tends to refuse applications developed with this approach.

Considering the final application that each framework produces, and how much this application is similar to a real native one, we can note some issues. Starting from the *Web Approach*, in this case we do not have a native application at all, but a (set of) web page(s). The *Hybrid Approach* provides a final application that can be installed on the device, but does not use native UI. The *Interpreted Approach* starts to introduce some native components, i.e., the User Interface, while using the interpretation step for the application code. Finally, the *Cross-Compiled Approach* produces a real native application, where no extraneous components are used and only native code is executed.

# 3. Development Tools analysis

Given the classification of the different frameworks depending on their approach and how the final application is developed, our first analysis aims at providing an evaluation of these frameworks considering the developer point of view. What we want to rank and consider in this analysis are the main aspects that can influence the choice of the best framework for a particular application for a developer that has to start to develop an application from scratch with the new framework.

This means considering, for example, the developing language, the availability of documentation or tutorials, or how frequently the framework is updated to support new features provided by new versions of the operating systems. We will see that it is possible to give a rank to these frameworks considering the various aspects of our analysis, but it is currently not possible to provide a final absolute rank since all frameworks have both strengths and weaknesses.

It is clear that this kind of evaluation is far to be an absolute one since, as we will see later, to provide an objective evaluation of all considered aspects, we should ask rankings from a huge amount of developers, which is out of the scope of this thesis. Therefore, our analysis can be seen a simple indication of the main characteristics and the all pros and cons of the frameworks.

## 3.1 Related works

Other works in literature try to compare frameworks for cross-platform development of mobile apps. Heitkötter et al. [71], compared jQuery Mobile[1], Sencha Touch[2], The-M-Project[3] and Google Web Toolkit combined with mgwt[4] according to a particular set of criteria, which included license and costs, documentation and support, learning success, user interface elements, etc. They concluded that jQuery Mobile is a good solution for simple applications or as the first attempt in developing mobile apps, while Sencha Touch is suited for more complex applications.

Palmieri et al. [109], evaluated Rhodes[5], PhoneGap[6] , dragonRAD and MoSync[7] with particular attention to the programming environment and the APIs they provide. The authors provided an analysis of the architecture of each framework and they concluded highlighting Rhodes over other frameworks, since this is the only one that supports both MVC framework and web-based services.

Issues about performances are discussed in [26] by Charland and Leroux. They stated frameworks based on web technologies experience a decrease in performances when the applications implement transition effects, effects during scrolling, animations, etc., but this problem affects essentially games, while the loss in performances is unnoticeable in business application, i.e., applications which support business tasks.

We must note here, that all the addressed works make a critical analysis of the chosen frameworks according to criteria that can be addressed studying the framework and its documentation, but they do not always require to use it, developing even a simple application. They studied the frameworks, the supported operating systems and sensors, the provided APIs, etc., but they do not try to implement a case study application to understand the real performances of the frameworks.

In some cases, the authors interview some developers about their experiences to understand their point of view. As an example, Majchrzak et al. [98], interviewed a set of local regional companies to find out what kind of apps they developed, what technologies they used and so on. As a result, they highlighted the best practices in use in the companies.

This approach is successful to describe the status quo of the cross-platform framework for apps development but the experiences of the developers can be very different, and can influence their judgment. Our idea is to ground this theoretical analysis by asking programmers to develop the same case study app with different framework to better

[1]jQuery Mobile: http://jquerymobile.com
[2]Sencha Touch: http://www.sencha.com/products/touch/
[3]M-Project: http://www.the-m-project.org/
[4]GWT: http://www.gwtproject.org/
[5]Rhodes: http://rhomobile.com/
[6]PhoneGap: http://phonegap.com/
[7]MoSync: http://www.mosync.com/

compare their performances.

Moreover, almost all the cited works consider only *business apps* in their analysis and not more dynamic applications like games. Therefore, a complete analysis of frameworks against features like animations and transitions is still missing.

## 3.2 Case Study

To compare performances of different cross-platform frameworks we choose a *serious game* which aims to help children to learn letters. The fishes Nemo and Dory race along a path and the child has to move the Nemo fish to avoid obstacles. Nemo moves between four positions denoted by four letters (see Figure 3.1), and the obstacles move horizontally. Sometimes a little octopus comes in, suggesting, with an audio message, the right position. Each time Nemo hits an obstacle, its speed is decreased, each time it avoids an obstacle or captures the right letter, its speed is increased. The blue bar at the top of Figure 3.1 shows the position of the two fishes, and who is winning.

This game requires implementing a lot of animations to move the two fishes on the blue bar, Nemo between the four positions of the letters, to move the obstacles and the octopus. Moreover, to realized the illusion of the horizontal movement of Nemo, the background is moved during all the play, changing its speed depending on the result of the choices of the player.

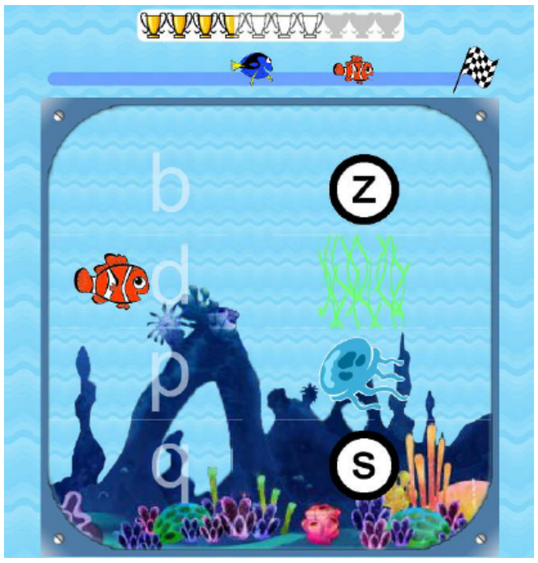


Figure 3.1: Screenshot of the test game

## 3.3 Analysis guidelines

An impartial comparison between different frameworks for cross-platform development is a challenging goal since programmers' background skills often bias evaluations, framework learning time, specific application, etc. For this reason, our evaluation assumes that developers that were asked to realize the application described in Section 3.2, already know the programming language used by each framework and have to learn only how to use the framework tools (SDK, APIs, etc.).

In the following, we provide a description of a set of criteria that we considered fundamental when analyzing cross-platform frameworks.

### 3.3.1 Licenses and costs

For each framework, we analyze how much expensive is to adopt a particular framework and how our final application can be distributed to the users or other developers.

Distributing licenses can be open source or closed source, with both positive or negative aspects, that can be used depending on the final purposes of the application. For example, let us consider an open source license like GPL[8]. This license allows to see the source code of the library of the framework, giving the possibility to understand better the behavior of several APIs, in particular when the documentation is very poor. One of the positive aspects of this type of license is that users can contribute to this library, providing new API or helping the developers in bugs correction.

For our analysis and development, we considered only frameworks that provide at least a free-of-charge usage plan. In the final grade, we also consider expensive solutions, but we give more weight to support and services provided for free.

### 3.3.2 API

The set of APIs provided by a framework allows to access to particular native system features and sensors that would not be available otherwise. To compare all the different APIs is almost impossible and clearly out of our purposes. However, not all the applications require the same set of APIs, but some APIs can be considered more important and relevant than others, i.e., the accelerometer, Bluetooth, etc. Table 3.1 provides the list of APIs that we consider as fundamental when developing dynamic applications.

Moreover, to understand how much the developers of a framework are active, we consider the list of open (and closed) bugs of the APIs. Clearly, this measure has to be carefully evaluated because a less used framework can have less open bugs, simply because of the small number of users who can find them.

Finally, two other key aspects are the possibility for the developer to create a third

[8]GNU General Public License: http://www.gnu.org/licenses/gpl-3.0.en.html

| | | | |
|---|---|---|---|
| NFC | Compass | FileSystem | In App Billing |
| Advertise | Calendar | Bluetooth | Accelerometer |
| Audio | Contacts | Geolocation | Notification |
| Camera | Database | Gyroscope | Networking |

Table 3.1: API evaluated over different frameworks

party API, to increment features that are not natively provided by the framework, and the APIs documentation.

### 3.3.3 Tutorials and Community

Tutorials are the first entry point for a programmer that wants to adopt a new framework. They are extremely important because they are provided by the developers of the framework with the scope of reducing the learning time necessary to develop an application.

In our evaluation, we considered the presence/absence of tutorials, the amount and the quality of the comments provided to code examples and the completeness with respect to the features provided by the framework.

When the documentation and the tutorials are not enough, the developers usually find a good help consulting the community. The more the framework is widespread and the community big, the more easy will be for the programmer to get answers to questions related to the development of the application.

For this reason, we consider the community as a separate evaluation issue. In particular, we consider:

1. *Official forum of the framework*: it is the most important because usually the developers of the framework are involved in the discussions (with high quality answers);
2. *Stackoverflow*[9]: to determine how much a framework is widespread, it is possible to filter the questions about each particular framework;
3. *Github*[10]: the evaluation is based on the number of followers, developers, and recent activities;
4. *LinkedIn Skill & Expertise*[11]: LinkedIn gives the possibility to researchers to understand how much a particular skill is widespread through the different users and which is the growing trend, even comparing it with similar ones;
5. *Google Trends*[12]: this tool gives the possibility to understand which are the most common queries made by users in a particular period.

[9]Stack Overflow, http://stackoverflow.com/
[10]GitHub, Inc.: http://github.com
[11]LinkedIn Skill & Expertise, LinkedIn: http://data.linkedin.com/projects/skills-and-expertise
[12]Google Trends, Google: https://www.google.com/trends/

From a developer point of view, the first three points are the most significant.

### 3.3.4 Programming complexity

To evaluate programming complexity, we try to measure the complexity of the code needed to develop the case study app with each different framework. However, it is extremely difficult to provide an absolute evaluation for this particular aspect, due to background skills of each developer, programming style, etc.; these are all aspects which influence the written code. As a general statement, we assume that a framework that provides a high number of APIs gives the possibility to write fewer lines of code.

We used CLOC [5], to count the number of code lines, and Gleeo Time Tracker [66] to keep track of the total amount of time needed. This measurement cannot be considered an exact measure of the code complexity, but it only gives an idea of how much code is required for that particular application with each framework. The use of a higher number of applications and developers could improve this evaluation, but it would be extremely time-consuming and is out of the scope of this evaluation.

### 3.3.5 IDE

An important factor, which greatly improves the development of applications, is the availability of good Integrated Development Environment (IDE), e.g., a Graphical User Interface (GUI) tool that gives the possibility to build the UI of the application with drag and drop operations, or a complete debugger. During this analysis, we evaluated the availability of these tools (considering Windows, Apple and Linux operating systems), and their usability.

### 3.3.6 Supported mobile devices

The number and type of supported devices are one of the most important aspects to evaluate. Clearly, the positiveness (or negativeness) of supporting (or not) a particular mobile platform has to be weighted with its diffusion. So, a wider diffuse platform will have more weight on the final grade.

We positively evaluated even the possibility to develop both a mobile and a web application.

### 3.3.7 Native user interface

An application provides a *native* User Interface (UI) if it uses the same style of the operating system for interaction widgets, e.g., buttons, check boxes, etc. This feature is usually very appreciated by the final users.

A framework that supports native UI allows the development of applications that have different UIs depending on the target platform. Usually, this is made with different CSS files for each supported device. In our case, even if the test application does not need a native interface, we evaluate its support very positively.

### 3.3.8 Basic knowledge

The last criterium considers the programming skills and technologies required by each framework. We preferred frameworks that require to know and use a low number of different programming languages/technologies and we evaluated the possibility to choose between several languages positively.

Another aspect that we have to consider is where to store information. Almost all the mobile operating systems usually provide the possibility to use an SQLite database, therefore we expect that even the frameworks provide this tool. The possibility to use XML files to store information can reduce the complexity of this operation since the management of an XML file is much more easier and less expensive in terms of computational costs, so will be positively evaluated.

## 3.4 Frameworks

In this section, we describe our evaluation based on the set of criteria discussed in Section 3.3. A comparison between frameworks based on the complete list of grades is reported in Table 3.2. All frameworks are graded with a number ranging from 0 to 5.

### 3.4.1 MoSync (*Cross-Compiled Approach*)

MoSync is a framework developed by MoSync Inc. since 2004[13]. It supports several mobile platforms, and it is particularly interesting since it produces native code and real native applications as final product.

It gives the possibility to choose between C, C++ and HTML+JavaScript as the programming language, though each language has a different set of APIs and supported features.

#### Licenses and costs

If used for commercial purposes, the MoSync framework requires to include a license inside the project, with the possibility to distribute it to the users. The are essentially two different licenses: GPL or the commercial MoSync license.

With the GPL license, the application will be freely distributed as what established by the GPL v2 license[14].

[13]MoSync: http://www.mosync.com/
[14]GPL v2 license: https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html

With the MoSync license, the developer has the possibility to distribute the application with close source. This commercial license has two versions, with different prices. For example, it is possible to get a free commercial license with support limited to the forum, while the most expensive one let the developer get much more support.

Interestingly, MoSync sells a license that gives the possibility to the developer to ask for several tests of his application from the tester unit of the company, using different devices.

Due to the high number of different possibilities and licenses our final evaluation is **5**.

### API

Although the total amount of supported APIs is high, they are not developed for recent versions of the platforms. iOS is supported till version 6.1, Android till version 4, Blackberry is in beta till version 6 and Windows Phone till version 7.8.

Despite this declared support, it has to be clear that even if a version is supported, not all the functionalities provided by that platform version are available through the APIs. Moreover, the programming language chosen from the development affects the set of available APIs. When a particular version of a platform is not supported, only backward compatibility is guaranteed.

Analyzing the APIs in detail, it is possible to see the lack of some important features: e.g., Bluetooth is not supported in the last version of iOS, while it is not possible to use Google Maps with Android.

As mentioned before, the main purpose of our analysis is to evaluate different frameworks for the development of a game. The biggest problem is that MoSync does not provide any API for animations. For this reason, it is necessary to build them manually, redrawing the objects every frame. Considering the other APIs, the main lacks are Bluetooth for JavaScript and Calendar and Contacts for both JavaScript and C++.

The documentation is complete only for the JavaScript implementation, while for C++ is extremely elementary and sometimes with big misses, e.g., the developer needs to look into the code of the library to understand the class that provides image scalability.

Finally, it is easy to include other APIs for the JavaScript implementation while, using C++, it is essentially impossible to add new libraries.

For all these reasons, the final grade is **3/5**.

### Community

The MoSync project is not extremely diffuse. The forum support is complete and provides an answer to the developer usually in a couple of days, but probably this time would decrease buying a commercial license. Not many questions are labeled with MoSync on *StackOverflow*, i.e., only 200 questions against 11000 of PhoneGap, and even on *LinkedIn* it is not much diffuse. MoSync has several followers on *GitHub*, and the developers share the code daily.

In the end, the community is relatively small, but the forum support is essentially enough to get answers for different problems during development; therefore, the final grade is **3/5**.

#### Tutorial

Several tutorials are provided to the users (almost 50), with examples from the usage of sensors to the *InApp Billing*, but no examples are provided for animations. The final grade is **4/5**.

#### Programming complexity

The test application developed with MoSync was written in C++, to reach the best performances. The lines of code necessary to implement the application using MoSync were 2194, and about 150 working hours that, as we will see later comparing these values with the ones of the other frameworks, show a high complexity of the framework. In particular, the most time-consuming task was to develop animations that are not natively provided by the framework. For this reason, the final grade is **2/5**.

#### IDE

The IDE provided for MoSync has both positive and negative aspects. The positive aspects are the possibility to test the application with the integrated simulator in all the supported platforms, to easily build the final application file and to select which are the necessary requirements, i.e., sensors, of the application. Unfortunately, the IDE suffers from a long start-up time and several lacks, like the absence of the logger for the Android platform.

The debugger is really poor for the C++ implementation, while it gives much more support and functionality using the JavaScript and web-view development, e.g., step-by-step analysis.

The IDE provides some interesting features but lacks of a complete integration with other tools that are usually provided for the development of native applications, therefore the final grade is **4/5**.

#### Supported mobile device

MoSync supports nine mobile operating systems: Android, Blackberry, iOS, J2ME, Moblin, MeeGo, Symbian, Windows Phone and Windows Mobile. But, as already mentioned in Section 3.4.1, the supported versions of these platforms are quite old, and the support is not complete. Due to this weakness in supporting recent versions of mobile operating systems, the final grade is **2/5**.

#### Native user interface

MoSync gives the possibility to choose between three supported languages: JavaScript, C++ and C. For the web-app development, using JavaScript, HTML and CSS is extremely easy to deploy "native like" applications using several JavaScript functions that build, at run-time, the right UI. It is even possible to use third party APIs and features that improve support to native interfaces.

For C and C++ applications, it is possible to build the UI using NativeUI or Maui. The first one allows to create native User Interfaces only for iOS, Android and Windows Phone. MAUI allows to develop a single UI that will be the same for every platform and device. Due to all these available possibilities, the final grade: **4.5/5**.

#### Basic knowledge

The developer can choose between a web-app, using HTML, CSS and JavaScript, or to use C++ or C. Clearly, these three possibilities allow the developer to choose the preferred one, drastically reducing the basic knowledge necessary to start with this framework. Therefore, the final grade: **5/5**.

#### Conclusions

MoSync has shown to be a wide and powerful framework. The possibility to develop an application combining either JavaScript and C++ (or using only one of them) is clearly a positive aspect. Even the performances of the application are great. Unfortunately, the available APIs are not complete, have a low level of support and are sometimes not well documented, even if the forum and the tutorials are very useful during the application development.

### 3.4.2 Titanium (*Interpreted Approach*)

Titanium is a cross platform framework developed since 2006 by Appcelerator[15]. It provides the possibility to develop mobile applications (in particular for iOS and Android) using HTML, JavaScript and CSS.

Differently from other frameworks, Titanium is not what we could call a "write once, run everywhere" framework, but it allows to write both code that can run in all platform and code that allows to exploit features of a specific platform, e.g., iCloud for the iOS system. However, the goal is to maximize the amount of reusable code for each application. In this way, Titanium allows to exploit the positive aspects of the cross-platform development, without losing the possibility to use some feature specific to a particular environment.

Titanium gives as final output a native application, so performances are excellent, even if the deployed application contains essentially a bridge between JavaScript calls and native

[15]Appcelerator Titanium: http://www.appcelerator.com/

APIs of the target device.

### Licenses and costs

Titanium provides either a free version of the framework and some versions with fees. The difference between them is that with the the versions with fee the developer gets the possibility to analyze the source code of the apps, a less strict limitation for the usage of cloud services and much more support. The negative aspect of these kinds of license is their cost, really high (the cheapest one is 999$ per month for at least 12 months). For these reasons, the final grade is **4/5**.

### API

Titanium provides a high number of APIs. They are well documented, containing a description of every method, its input and output parameters and, sometimes, an usage example.

Titanium provides several APIs for transactions, rotations and scaling, with the possibility to combine them together. The biggest limitation is that it is not possible to know the exact position of a moving object. In this case, it is necessary to implement a timer that changes the position of the object. Unfortunately, this very simple solution decreases the fluidity of the system.

All the most important APIs mentioned in Table 3.1 are supported for most mobile systems. We must note here, that Titanium lacks support for Bluetooth. Moreover, there are a lot of APIs for Android and iOS, but for other platforms, the set of provided APIs and features is very poor. Therefore, the final grade is **4.5/5**.

### Community

The community behind Titanium is extremely wide. There are 470k registered developers, with a lot of conversation on the forum. Even *StackOverflow* counts a lot of questions and answers, showing that the framework is widely known. So, the final grade is **5/5**.

### Tutorials

There are several online resources where it is possible to get information to start developing with this framework. The official website provides several examples, tutorials and even videos dedicated to who wants to start to develop with this framework. It is necessary to mention even *KitchenSink*, a free application that shows the majority of the feature provided by the framework and how to use them. For this reason, the final grade is **5/5**.

### Programming complexity

To develop our application using the Titanium framework, the total amount of lines necessary were 1392, while the working time was of about of 58 hours. With respect to the MoSync framework, the total lines of code were 37% less, really similar to the amount of required time. This result clearly shows a simplicity of the framework that decreases the amount of time necessary to develop an application. Therefore, the final grade is **5/5**.

#### IDE

The framework provides the Titanium Studio IDE, which contains several tools for developing and debugging. However, this IDE requires several native tools, e.g., the Android DDMS (to access to the log of the application), to fully support the developer. For this reason, the final grade is **4/5**.

#### Mobile devices support

The platforms supported by this framework are iOS, Android, Blackberry, Tizen and Web Mobile, but the APIs almost cover Android and iOS. The versions supported for Android start from version 2.3.3 (only less than 4.9% do not reach this minimum version), while for iOS from version 5.0 (only 1.4% of the devices do not have at least this version).

Although the great variety of supported platforms, we must note here that Titanium provides a good set of APIs only for iOS and Android.

Moreover, it is possible to develop even a web application, that could work on every device due to its integrated browser. The application is based on HTML5 and not all the APIs are supported, but only the ones that can be used with a browser. Therefore, the final grade: **4/5**.

#### Native user interface

Considering the main platforms supported by Titanium, iOS and Android, all the graphics elements of the application are coherent to the target platform. This means that the same application will have completely different User Interface if deployed on an iOS or Android device.

To increase the performances of the graphic rendering, the base technique used by the framework is *lazy instantiation*. In this way, objects are created only when needed and immediately deleted when no more necessary. For this reason, the final grade is **5/5**.

#### Basic knowledge

To develop applications using Titanium, a developer simply needs to know JavaScript. All the APIs of this framework can be called simply using this language, that is a mandatory prerequisite, therefore, the final grade: **5/5**.

#### Conclusions

From our experience, Titanium has shown several good aspects, e.g., the high number of APIs, good documentation and it simplifies the development of an application. Therefore, Titanium must be considered every time is necessary to develop a cross-platform application, even for the frequent updates of the platform and APIs.

The negative aspect of this framework relies on the support mainly limited on iOS and Android.

### 3.4.3 jQuery & jQuery mobile (*Web Approach*)

jQuery[16] is a JavaScript library that provides several facilities to developers, like DOM traversal and manipulation, event handling, animations and AJAX calls in a cross-browser and easy way. jQuery mobile[17] is a framework built on jQuery that provides several graphics elements designed for mobile devices.

We must note here that, unlike other frameworks, jQuery and jQuery Mobile allow the developer to built web applications that will work either on a desktop and mobile web browsers, but not native mobile applications. Therefore, the user must access the application through the web browser.

#### API

jQuery provides a wide set of different APIs for every purpose, e.g., adding or deleting a new element, handling click events, object's style manipulation, etc.

jQuery also provides a set of APIs for animations, like fade in/out, hide, show, animate, etc. The biggest problem of these APIs is that even if they work really good with the desktop browsers, several problems arise in the mobile environment. In particular, what we experienced is a degradation of the performances, i.e., the application is not fluid anymore.

The biggest problem is the support for specific features of the mobile devices, like Bluetooth, accelerometer, gyroscope, etc., since jQuery (and jQuery mobile) support is limited to what is accessible through the web browser. This is clearly a big issue because most of the games relies on these features to improve the user experience and so the adoption of this framework would drastically decrease the possibilities offered to the developer.

Several plugins are available thanks to the community, which are a significant advantage in reducing the necessary developing time.

To conclude, jQuery & jQuery mobile provides several APIs, even for animations, but the degradation of the performances with mobile devices and the impossibility to use sensors are clearly a strong limitation. For these reasons, the final grade is **2/5**.

#### Programming complexity

The development of the application with jQuery and jQuery Mobile has been fast enough, thanks to the big number of used APIs. Most time has been dedicated to cross-browser testing since there are no automatic tools able to provide information about compatibility.

[16] jQuery: https://jquery.com/
[17] jQuery Mobile: http://jquerymobile.com/

In particular, the number of lines of code necessary was 1249, while the amount of hours were 90. Therefore, the final grade is **4/5**.

#### IDE

No IDE is provided by the jQuery team to develop applications with this framework.

#### Supported mobile devices

jQuery supports the majority of the mobile devices like iOS (from version 3.2), Android (from version 2.1), Windows Phone (version 7.5-8), Blackberry, etc., and even several mobile browsers like Firefox Mobile, Chrome for Android, Opera Mobile.

As mentioned before, even if all these mobile systems and mobile browsers are supported, it is necessary to remember that support for complex web applications is only theoretical due to performance degradation. Therefore, despite performances degradation, the final grade is **5/5**.

#### Native user interface

jQuery mobile offers, by default, a CSS theme that is similar to the iOS native interface. The community provides several CSS templates that can be included inside a web application to provide a similar, but not identical, to the native one, look and feel, either for Android and iOS. What it is necessary to consider is that the main purpose of jQuery Mobile is to provide a look and feel that will remain the same over the different platforms from which the web application will be accessed. For this reason, the final grade is **3/5**.

#### Basic knowledge

As mentioned before, jQuery and jQuery Mobile require HTML, CSS and JavaScript as mandatory programming languages. No other languages are required, and, due to the enormous diffusion of these languages, the requirements are low. So, the final grade is **5/5**.

#### Conclusions

jQuery and jQuery Mobile provide the possibility to develop a web application either for desktop and mobile browsers. Several APIs exist but their performances, in particular for animations, decrease with mobile browsers and it is not possible to use sensors that are not accessible through the browsers.

### 3.4.4 Phonegap (*Hybrid Approach*)

PhoneGap[18] is a frameworks that allows to wraps a web application, developed with HTML, CSS and JavaScript, inside a WebKit engine that will run on the selected device looking like a native application. It even provides access to different sensors of the devices through dedicated APIs.

[18]PhoneGap: http://phonegap.com/

### API

PhoneGap defines several APIs that provide the possibility to access the most important sensors and tools of mobile devices. In particular, it provides access to the accelerometer, the camera, geolocalization services, contacts, calendar, etc.

Phonegap does not provide any particular API for animations, that are the core of our case study game, but, due to its nature of web application, it can be extended with other frameworks, i.e., jQuery. The result that we got from our experience is that even in this case, as in the case of jQuery Mobile, the application is fluid and very usable, if accessed through a web browser, while it encounters performance problems if used in a mobile device. Our tests show that the animations are not fluid, and the reasons are too many images used, dimensions of the images and memory usage, image scaling or particular CSS properties like *background-repeat*.

For this reason, even PhoneGap does not provide satisfactory performances for mobile game application and its final grade is **2/5**.

### Programming complexity

The total amount of time necessary to develop an application with PhoneGap is almost the same of a web-application using jQuery or jQuery Mobile (90 hours and about 1250 lines of code). In fact, PhoneGap simply requires to create a new project with the web application for the target platform and to change (or to add) very few JavaScript functions (in particular the ones related to smartphones sensors and features). Therefore, we assume to use the same grade of other web framework for mobile (**4/5**).

### IDE and Native user interface

PhoneGap does not provide any IDE to develop applications. The developer has to rely on IDE for native applications, i.e., XCode for iOS or Eclipse for Android.

Moreover, it does not provide the possibility to have Native User Interface. To reach this goal, it is necessary to use PhoneGap with other frameworks like jQuery Mobile or Sencha Touch that can give to the application a Native User Interface. Therefore, these two criteria are not applicable.

### Supported mobile devices

PhoneGap supports the most common mobile operating system: Android, iOS, Blackberry and Windows Phone. The most complete APIs support is provided for Android and iOS, while Blackberry and Windows Phone have some APIs provided by Phonegap that are not already implemented yet, i.e., compass and media. For this reason, the final grade is **4/5**.

| | MoSync | Titanium | jQuery | Phonegap |
|---|---|---|---|---|
| **Licenses** | 5 | 4 | 5 | 5 |
| **API** | 3 | 4.5 | 2 | 2 |
| **Community** | 3 | 5 | 5 | 5 |
| **Tutorials** | 4 | 5 | 4 | 5 |
| **Complexity** | 2 | 5 | 4 | 4 |
| **IDE** | 4 | 5 | - | - |
| **Devices** | 2 | 4 | 5 | 4 |
| **GUI** | 4.5 | 5 | 3 | - |
| **Knowledge** | 5 | 5 | 5 | 5 |

Table 3.2: Final remarks comparison between frameworks

#### Basic knowledge

As mentioned before, Phonegap requires HTML, CSS and JavaScript as mandatory programming languages. No other languages are required, and, due to the enormous diffusion and knowledge of them, the requirements to be able to start developing web applications are low. Therefore, the final grade is **5/5**.

#### Conclusions

Phonegap is a powerful framework that provides the possibility to convert a web application to a native application, where this native application is a WebKit engine wrapper that wraps the web application. It even provides access to the most common sensors and tools of the different smartphones. The most important issue regards the performances of the final application, which are far from the performances of the web or a native application.

### 3.4.5 Final remarks

To conclude and provide a final recap of all these evaluations, what we can see is that at the time of writing the cross-platform frameworks considered in our analysis has good potential, but are still not sufficiently advanced to provide a complete support to developers in applications development. For example, several frameworks still not provide a sufficient amount of APIs to give access to native sensors or information, i.e., contacts or media, or lacks on the number of supported mobile devices, both because they do not support them at all or because they do not provide support for the latest version of the mobile operating system. This clearly constitutes a big limitation for developers, since if a framework does not support all the possible native APIs, it clearly reduces the number of features that the application can provide.

# 4. Energy consumption issues analysis

The analysis described in the previous chapter focused its attention on a developer point of view. One of the most important aspect when dealing with mobile devices and applications is energy consumption [18] [119]. As we already discussed in Section 2.1, different framework approaches bring to completely different final applications, which can be more or less close to the native solution, especially regarding application architecture.

In this chapter, we analyze how these different approaches to deploy the final application influence battery consumption. This analysis allows to provide objective data and shows that there are differences between each framework and the native solution, but also between the different approaches for cross-platform development. The experiments considered the two most diffuse mobile operating systems, Android and iOS, testing for both of them two different devices. Data about energy consumption allows to decide which framework is the best solution depending on the application we want to develop when energy consumption is a critical issue. For example, updating or not the User Interface, strongly influence energy consumption, and different applications can lead to different choices of the cross-platform framework to use.

It is clear that considering energy consumption includes other measures and data, e.g., execution speed, memory usage, etc. Indeed, it is clear that if an application consumes more than another one, probably this comes from a higher number of FLOPS, higher memory usage and consumption, etc. The value of the energy consumption puts together all this factors, and it is probably the most expressive and significative value, especially from the end-user point of view.

## 4.1 Related Works

An initial survey of several write once run everywhere tools, i.e., PhoneGap, Titanium and Sencha Touch, is performed in [38]. They examined performances in terms of CPU, memory usage and power consumption for Android test applications. They concluded that PhoneGap is the less expensive framework in terms of memory, CPU and energy consumption.

Although many authors analyze framework for cross-platform mobile development, to the best of our knowledge authors address the problem of energy consumption, considering frameworks only from a static point of view, i.e. from a developer perspective. If we consider the user point of view, energy consumption is critical. For this reason, energy consumption has been investigated for mobile native applications in [100], [25], [113], in particular to find a way to correctly measure it. WattsOn [100] is an energy emulation tool that allows the developers to estimate energy consumed by their apps using their workstation. WattsOn can simulate different device models and to determine which component of the application consumes more energy. Since it uses a static approach, using a power modeling techniques, the measurement suffers of an error, which was estimated, using the Monsoon PowerMonitor[1], between 4% and 9%.

As stated by [100], the use of external power monitor is tedious and expensive, so many authors put much effort to create good power model. Caroll and Heiser [25] performed an analysis of power consumption of a particular smartphone and tried to design a model of energy consumption in different scenarios.

Hao et al. [69] combined program analysis and per-instruction energy modeling to estimate energy consumption of mobile application. This novel approach is called eLens. Even in this case, the authors use a static approach, and the error of the estimation, measured with an external monitor, is about 10%.

PowerScope [52] is an Android tool that is able to analyze energy consumption of applications. The tool measures the energy consumed by each application. It also provides a feedback and a comparison between different applications. Since the tool is an application that works in background, it is not possible to understand and measure the overhead introduced by the application itself on energy consumption. Moreover, it is not even possible to know if, and how, this application can influence energy consumption of a particular task or application, meaning that the measures provided by the application cannot be considered too precise (or not affected by an unpredictable error).

Although the use of a external power monitor can be very expensive, related works discussed so far show that the static approach necessarily makes an error, therefore the use of external monitor allows a more precise measurement, as stated in [20], where the authors

[1]Monsoon Power Monitor, https://www.msoon.com/LabEquipment/PowerMonitor/

propose a combined approach. Balasubramanian et al. [11] followed this suggestion. They started by measuring energy consumption for mobile applications which use networking technologies, in particular 3G, GSM and WiFi. Based on the acquired data, they developed a power model for energy consumed by network activity, TailEnder, which can be used by developers to minimize the consumed energy.

Thiagarajan et al. in [131] analyzed energy consumption of popular websites like Facebook, Apple, YouTube, etc. when rendered using a mobile browser and an Android device. Using a digital power multimeter, they were able to understand and analyze energy consumption of different elements of a web page, like Javascript code, CSS style rendering, image downloading, etc. The main conclusion of their work is that Javascript code or CSS stylesheet should be page specific, meaning that eliminating unnecessary code helps in saving even more than 30% of battery energy.

Friedman et al. [60] analyzed power and throughput performances of Bluetooth and Wifi communication on smartphones. They experimented different scenarios and setups, considering both performances and power consumption associated with each experiment. The results showed that, in several conditions, WiFi performances are better than the ones using Bluetooth, contradicting previous researches. To measure energy consumption, they used an external power supply, measuring the voltage across a resistor that was serially connected to the smartphone.

## 4.2 Methodology

In this Section, we provide some details about the hardware and software setup used during the experiments. In order to cover the wider set of devices and potential users, we deploy our applications both for the Apple and the Android platforms. These two operating systems together cover about 96% of the mobile devices market [133].

### 4.2.1 Hardware setup

The test devices, used during our experiments to measure performance in terms of consumed energy, were two Android and two Apple smartphones. For the Android platform, we used a Samsung Galaxy Nexus i9250 and a Samsung Galaxy S5. The Galaxy Nexus i9250 is equipped with a Dual-core 1.2GHz CPU, 1GB RAM, a 720x1280px display with 16M colors, and a Li-Ion 1750mAh battery. The Samsung Galaxy S5 is equipped with a Quad-core 2.5Ghz CPU, 2GB RAM, a 1080x1920px display with 16M colors, and a Li-Ion 2800mAh battery.

Considering the Apple platform, we performed our tests using an iPhone 4 and an iPhone 5. The iPhone 4 is equipped with a 1GHz Cortex CPU, 512MB RAM, a 640x960px display with 16M colors, and a Li-Po 1420mAh battery. The iPhone 5 is equipped with a

Dual-core 1.3GHz CPU, 1GB RAM, a 640x1136px display with 16M colors, and a Li-Po 1440mAh battery.

To measure the energy consumed by each application, we do not use software tools installed on the smartphone but the Monsoon PowerMonitor[2]. The main function of the PowerMonitor is to measure the energy that is requested by the smartphone (or other devices that uses a single lithium battery). The necessary circuit is created isolating the Voltage (positive) terminal of the battery and creating a bypass between the PowerMonitor `Vout` to the device. The circuit is finally closed connecting directly to the Ground (negative) terminal on the battery. An example of hardware setup for the Android device is provided in Figure 4.1.

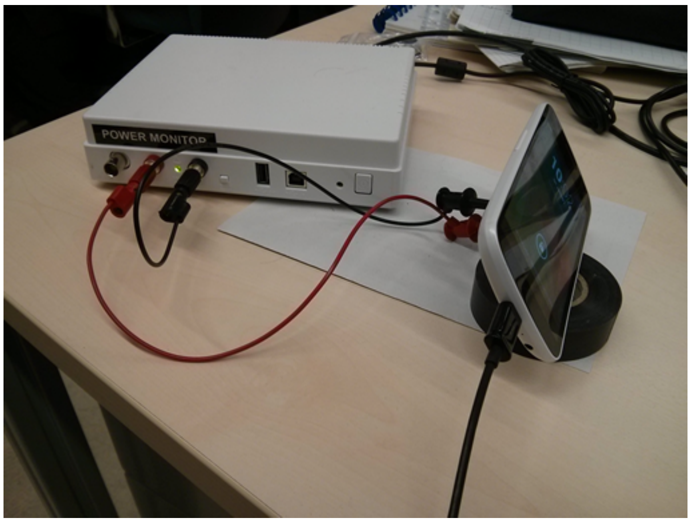


Figure 4.1: Hardware setup with one of the Android device

During the experiments, the smartphones do not use the energy provided by their battery, but they use energy provided by the PowerMonitor. This means that even if the battery is connected to the device, the battery is not able to provide energy (due to the isolation of the positive terminal), and the PowerMonitor provides (and measures) the requested energy.

This tool is provided with a software interface which reports several information like consumed energy (measured in $\mu$Ah), average power (measured in mW), average current (measured in mA), average voltage (measured in V) and expected battery life (hrs). The most important measure for our purposes is the consumed energy: comparing two different tasks along the same time interval and in the same test conditions, the more a task is energy expensive, the more the energy consumption will be higher. This lets us have a comparison between different frameworks and sensors. Even if interesting, the expected battery life value is an estimation of the software (based on the battery size value provided

[2]Monsoon Power Monitor, https://www.msoon.com/LabEquipment/PowerMonitor/

to the software as input) and cannot provide an objective information about the real energy consumption.

The use of PowerMonitor is very simple if the smartphone provides access to the lithium battery, so we choose two Android devices with this features (not always present in modern devices). Some difficulties raised with Apple devices. We opened an iPhone 4 and an iPhone 5 to access the battery, but even in this case, the terminals of the battery were unreachable. So we created the necessary circuits using the connectors between the battery and the smartphone.

We did not choose a software monitor (some examples are discussed in Section 4.1) since, running on the device, it wastes energy in its turn, so it may influence the measurements that are then affected by an error very difficult to predict. According to [100], this error can vary between 4% and 9%, thus affecting, in a significant way, the provided results.

We created an application, and therefore a test, for each considered framework and for each sensor. Each test lasted 2 minutes and we repeated each test three times. The final results for each test have low standard deviation, meaning that the recorded data is very informative and is not affected by underground noise. To have comparable results among all the tests, it was necessary to define a "base" smartphone conditions that could be used during all the experiment with a good change to produce the same energy consumption of the smartphone, i.e., it cannot be affected by external factors. For this reason, all devices were used in "Airplane mode" (without mobile or Wifi network), with the luminosity of the screen set to the lowest value and, when not tested, the GPS is disabled.

### 4.2.2 Software setup

These experiments compared together the four different cross-platform approaches described in Section 2.1, e.g.,*Web Approach*, *Hybrid Approach*, *Interpreted Approach* and *Cross-Compiled Approach*: a web application for the *Web Approach*, PhoneGap for the *Hybrid Approach*, Titanium for the *Interpreted Approach* and MoSync for the *Cross-Compiled Approach*, considering both the C++ and the Javascript implementation.

The critical issue for software setup is to be able to reproduce the same test conditions among different frameworks. As an example, in [131] the authors reported that the usage of support libraries when working with HTML5 and Javascript applications, e.g., jQuery, increases the final energy consumption of the application. Therefore we define, as baseline used to build our test applications, the following conditions:

- we used only Javascript and not any supporting libraries to manipulate the DOM when developing test application for the *Web* and the *Hybrid Approach*;
- the background of each application was black, the foreground color for the text was

white and the font size 15px.

We choose black and white as background and foreground colors, because colors representation are standard (e.g., RGB), but how colors are rendered in screen can change according to the device. This means that brightness of a particular color can be different, and so the amount of consumed energy [27]. For this reason, black and white which roughly correspond to on and off, are the only colors which are not affected by this problem and we can assume to cause the same energy consumption.

The choice of the font size is arbitrary and could be different, what is important is that the selected conditions are reproducible on all the tested applications and devices. However, using this configuration our results can be considered as a lower bound of energy consumption, since the black background is the less expensive.

Since our purpose is to explore how cross-platform frameworks influence battery consumption when acquiring data from mobile sensors, the conditions mentioned above are sufficient to assure the same testing conditions for all the frameworks when we analyze energy consumption without updating the User Interface (UI), but only acquiring data from a particular sensor.

If we also update the User Interface, the situation is more complex. Let us consider, for example, the light sensor. In this case, it is clear that even the minimum change in the light environment, e.g., a movement of a person, could influence the data acquired from the sensor and so if the application updates or not the interface. This makes quite impossible to reproduce the same test conditions among all the frameworks, since it is not possible to fully control the environmental conditions, unless closing cell phone and energy monitor inside an opaque box, but in this case the UI is not updated at all since data acquired never changes. Therefore, for some sensors, i.e., light, proximity sensor and GPS, we decided to analyze only the case when the User Interface is not updated, to best guarantee equity between tests. For the audio test, we simply record audio from the microphone, and no User Interface is updated during this test.

Despite the native solution, that clearly supports the access to all the native device features and sensors, this is not true for the cross-platform frameworks. The support offered by each framework to a particular device feature depends on the framework itself and, sometimes, on the chosen platform for deployment. This means that, for the most common features like the accelerometer or the GPS, the support is almost complete, while many frameworks do not provide support for other features like light sensor or the device orientation.

Moreover, even the way a framework provides support to a particular feature can change between frameworks, e.g., the update frequency of a sensor value. For example, the updating frequency of the accelerometer value could be the same between the *Web*

*Approach* and the *Interpreted Approach* or, a framework could not allow the developer to change the updating frequency from the default value. In our test we use the following policy:

- if the framework allows to change the value of the updating frequency, we test different fixed updating frequencies to understand how energy consumption changes in relation with the update frequency;
- otherwise, we recorded data only at the supported frequency.

Table 4.1 (for the Android platform) and Table 4.2 (for the iOS platform) provide a resume of the supported features of each framework. We can note, for example, that the sets of available browsers are different, i.e., Chrome, Firefox and Opera for the Android platform and Chrome and Safari for the iOS platform. Moreover, Titanium supports the proximity sensor and audio recording only for iOS devices.

| **Sensor** | **Web App.** | | | **Hybrid App.** | **Interpreted App.** | **Cross-platform App.** | |
|---|---|---|---|---|---|---|---|
| | **Chrome** | **Firefox** | **Opera** | **PhoneGap** | **Titanium** | **MoSync (C++)** | **MoSync (Javascript)** |
| **Accelerometer** | Yes | Yes | Yes | Yes (FC) | Yes | Yes (FC) | Yes |
| **Device orientation** | No | Yes | No | No | No | Yes (FC) | No |
| **Compass** | No | No | No | Yes (FC) | Yes | No | Yes |
| **Proximity** | Yes | No | No | No | No | No | No |
| **Light** | Yes | No | No | No | No | No | No |
| **GPS** | Yes | Yes | Yes | Yes | Yes | Yes | No |
| **Camera** | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| **Audio Record** | Yes | Yes | Yes | Yes | No | No | No |

Table 4.1: Features supported by the cross-platform frameworks considering the Android platform. By "FC" we mean that the framework lets the developer choose the update frequency.

During the tests, our devices were equipped with Android 5.1.1 (Samsung Galaxy S5) and Android 4.4 (Samsung Galaxy i9250), while for the Apple platform we tested the iPhone 4 with iOS 7.3 and the iPhone 5 with iOS 8. Software used during the tests were Chrome v. 41.0, Firefox v. 31, Opera v. 22.0, MoSync v. 3.8, Titanium v. 3.2.3 and PhoneGap v. 3.2. The first idea was to provide the same ambient for the tests in the different devices, i.e., the same operating systems and software. Unfortunately, it is not possible neither to update the iPhone 4 to iOS 8 since Apple does not allow it, nor to downgrade the iPhone 5 to iOS 7.3 for the same reason. The same situation takes place for the Android devices. For this reason, we decided to run our tests in the most updated operating system according to the device in use.

| **Sensor** | **Web App.** | | **Hybrid App.** | **Interpreted App.** | **Cross-platform App.** | |
|---|---|---|---|---|---|---|
| | **Chrome** | **Safari** | **PhoneGap** | **Titanium** | **MoSync (C++)** | **MoSync (Javascript)** |
| **Accelerometer** | Yes | Yes | Yes (FC) | Yes | Yes (FC) | Yes |
| **Device orientation** | Yes | Yes | No | No | Yes (FC) | No |
| **Compass** | No | No | Yes (FC) | Yes | Yes (*) | Yes |
| **Proximity** | No | No | No | Yes | Yes | No |
| **Light** | No | No | No | No | No | No |
| **GPS** | Yes | Yes | Yes | Yes | Yes | No |
| **Camera** | No | No | Yes | Yes | Yes | Yes |
| **Audio Record** | No | No | Yes | Yes | No | No |

Table 4.2: Features supported by the cross-platform frameworks considering the iOS platform. By "FC" we mean that the framework lets the developer choose the update frequency. Framework marked by (*) retrieves 3 values for the coordinates $x$, $y$ and $z$. Other frameworks provide only a value which is a combination of the coordinates.

## 4.3 Experimental Results

In this Section, we report the results of our experiments. As already discussed, each experiment runs the application for two minutes, and we repeated each test three times. Then we calculated the mean value of the "Consumed Energy" of the three runs and the standard deviation, denoted by $\pm\, n$ where $n \in \mathbb{N}$. The "Consumed Energy" value returned by the Power Monitor interface is measured in µAh and is the most representative value of power consumption, so it can be used to compared performances of the different frameworks.

To evaluate the effective cost of each framework, in terms of power consumption, we need a reference value as baseline. The ideal baseline value would be the smartphone on, without any other application or background process running, and where the "computational workload" is the lowest possible. To reach this lowest value as closely as possible, we put the smartphones in "Airplane Mode", without any running application, with the lowest luminosity of the screen and black background of the home screen. Moreover, we minimized the number of graphical elements on the screen, i.e., the icons. For the Android devices, we were able to remove all the graphical elements on the screen (despite some small icons and the time in the top right of the screen), and we measured a consumption of $5126_{\pm 11}$ µAh for Galaxy Nexus and $2213_{\pm 15}$ µAh for Samsung S5. For the Apple devices, we created this ideal "quiet state" by setting a black background of the screen, and choosing a black icon for an application, since it is not possible to remove all the graphical elements from the Home screen, but the "quiet state" of the iOS system has at least one icon in the upper part of the screen. The consumption value for the Apple devices were $2777_{\pm 10}$ µAh for the iPhone 4 and $2558_{\pm 6}$ µAh for the iPhone 5.

| | | Web App. | | | Hybrid App. | Interpreted App. | Cross-compiled App. | |
|---|---|---|---|---|---|---|---|---|
| **Sensor** | **Native** | **Chrome** | **Firefox** | **Opera** | **PhoneGap** | **Titanium** | **MoSync (C++)** | **MoSync (Javascript)** |
| **Only App.** | +0,06% | +3,69% | +2,51% | +3,74% | +3,03% | +1,06% | +0,38% | +1,72% |
| **Accelerometer** | +36,58% | +115,38% $^{(50)}$ | +168,01% | +123,74% $^{(50)}$ | +75,01% | +79,29% | +240,88% | 74,07% |
| **Device orient.** | +45,84% | - | +180,15% | - | - | - | +250,12% | - |
| **Compass** | +30,87% | - | - | - | +47,72% | - | - | +31,69% |

Table 4.3: Data recorded during tests using the Galaxy Nexus, updating the User Interface with sensor data, expressed in term of percentage of increase. $^{(x)}$ indicates the different update frequency measured in ms.

| | | Web App. | | | Hybrid App. | Interpreted App. | Cross-compiled App. | |
|---|---|---|---|---|---|---|---|---|
| **Sensor** | **Native** | **Chrome** | **Firefox** | **Opera** | **PhoneGap** | **Titanium** | **MoSync (C++)** | **MoSync (Javascript)** |
| **Only App.** | +0,09% | +3,17% | +2,11% | +3,67% | +2,74% | +1,28% | +1,01% | +1,98% |
| **Accelerometer** | +115,08% | +675,37% $^{(50)}$ | +764,25% | +691,59% $^{(50)}$ | +159,62% | +334,59% | +774,04% | 129,87% |
| **Device orient.** | +123,17% | - | +485,78% | - | - | - | +686,24% | - |
| **Compass** | +120,61% | - | - | - | +162,31% | - | - | +143,47% |

Table 4.4: Data recorded during tests using the Galaxy S5, updating the User Interface with sensor data, expressed in term of percentage of increase. $^{(x)}$ indicates the different update frequency.

We must note here that the Apple devices has similar baseline, while the Android devices have not. Even if this is an interesting aspect to notice, this difference in energy consumption does not affect the results and the conclusions we made for the frameworks, since energy consumption increase is calculated using this reference value that is the same for all the tested frameworks given the same device. All tables and figures in this section reported the consumed energy expressed in terms of percentage increase, with respect to this baseline. For this reason, even the native solution does not report a zero value since also a basic application has an energy consumption superior, by definition, to the baseline.

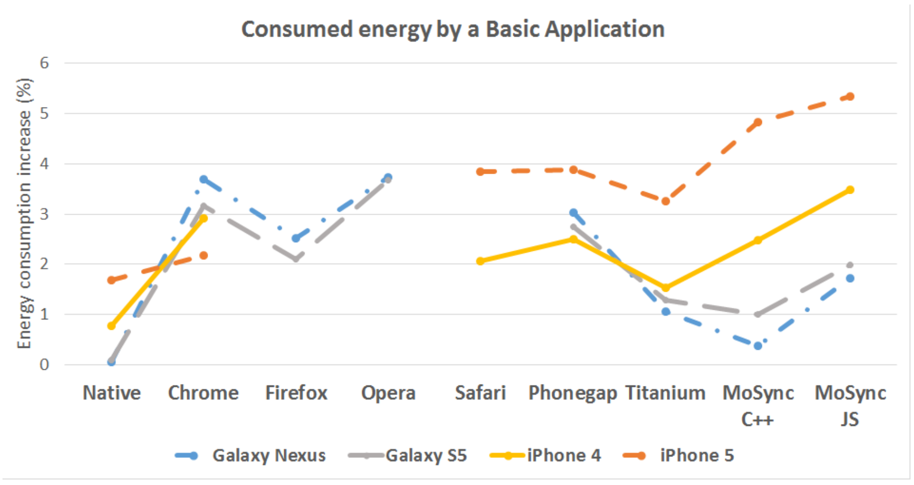


Figure 4.2: Consumed energy, expressed in terms of percentage increase, by a basic application developed with the different cross-platform frameworks.

| | | Web App. | | Hybrid App. | Interpreted App. | Cross-compiled App. | |
|---|---|---|---|---|---|---|---|
| **Sensor** | **Native** | **Chrome** | **Safari** | **PhoneGap** | **Titanium** | **MoSync (C++)** | **MoSync (Javascript)** |
| **Only App.** | +0,78% | +2,91% | +2,06% | +2,5% | +1,53% | +2,49% | +3,49% |
| **Accelerometer** | +8,68% | +94,48% (50) | +94,47% (50) | +79,10% | +26,38% (100) | +190,94% | +25,38% (150) |
| **Device orient.** | +15,21% | +950,4% (50) | +49,98% (50) | - | - | +180,05% | - |
| **Compass** | +51,98% | - | - | +69,51% | - | - | +42,79% (140) |

Table 4.5: Data recorded during tests using the iPhone 4, updating the User Interface with sensor data, in term of percentage of increase. $^{(x)}$ indicates the different update frequency.

| | | Web App. | | Hybrid App. | Interpreted App. | Cross-compiled App. | |
|---|---|---|---|---|---|---|---|
| **Sensor** | **Native** | **Chrome** | **Safari** | **PhoneGap** | **Titanium** | **MoSync (C++)** | **MoSync (Javascript)** |
| **Only App.** | +1,69% | +2,17% | +3,85% | +3,88% | +3,26% | +4,83% | +5,34% |
| **Accelerometer** | +5,34% | +86,78% (50) | +144,84% (50) | +52,16% | +15,01% (100) | +298,28% | +16,52% (150) |
| **Device orient.** | +14,26% | +86,53% (50) | +147,07% (50) | - | - | +286,47% | - |
| **Compass** | +30,25% | - | - | +48,25% | - | - | +43,97% (140) |

Table 4.6: Data recorded during tests using the iPhone 5, updating the User Interface with sensor data, in term of percentage of increase. $^{(x)}$ indicates the different update frequency.

The first experiment wants to measure the energy consumed only by the use of a particular framework to create an application without any particular behavior. Therefore, for each considered framework, we developed a basic application, which does not perform any operation, without any graphical elements or underground task. Then, we compared the energy consumption of all these applications with a native application with the same features. Results are provided in Figure 4.2, expressed in terms of increase percentage with respect to the baseline measured for the used device. Since Firefox and Opera cannot be installed on the Apple devices, data are missing. In the same way, since Safari is not available for the Android environment, we did not test this combination of browser and operating system.

As we can see, the adoption of an application developed using a cross-platform framework has a cost in terms of energy consumption. If we do not consider the *Cross-Compiled Approach* which has very different performances in the two platforms, the more expensive approaches are the *Web Approach* and the *Hybrid Approach*, that have to run a browser engine, that is clearly more expensive than a simple application. Instead, the performances of the *Cross-Compiled Approach* are affected by the chosen development language and platform. In both operating systems, i.e., Android and Apple, performances measured for the application developed using Javascript are worst with respect to the ones collected using C++. But the two platforms perform very differently: while in the Android platform the C++ application is the best one among all other cross-platform frameworks, and the Javascript implementation performs better than the *Web* and the *Hybrid Approaches*, in the Apple platform the performances using Javascript are the worst.

Another important result of this first experiment is related to the generalization of these results. Even if the number of used devices is low, we can notice that, if we look separately at the two platforms, the curves trend is comparable, i.e., the curve trend of data for iPhone 4 is very similar to the curve trend of data for iPhone 5 and the same holds for the experiments with Android. This is very important because we can assert that these results are representative for both the platforms.

After this initial test, we analyzed the performances of each framework, when acquiring data from different sensors. Clearly, we tested only that sensors that were supported by at least one framework, and we compared the performances with the native solution. The initial idea was to compare frameworks using the same update frequency, but not all of them let the developer choose it, as already explained in the previous section. Therefore, data presented in the following tables and figures refer to the same update frequency (64ms), unless the framework does not allow for customization. In this case, the update frequency is indicated between rounded brackets in Tables 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9 and 4.10.

These experiments give two interesting results, the first one is, in some way, predictable: the adoption of a cross-platform framework to develop an application always implies an higher energy consumption with respect to the native solution, and this is true for all frameworks in both the considered device platforms. If we take as an example the acceleromenter, Figure 4.3 shows that the curve trend is always the same, independently from the used platform. The same behavior appears also for other sensors like compass (see Figure 4.4), proximity sensor (see Figure 4.5) and to collect data about device orientation (see Figure 4.6). Although the limited number of devices and the fact that they are made by only two producers, these devices are very different in terms of screen size, battery capacity and lifetime. Despite these differences, the charts which illustrate the results show the same curve trends. For this reason, we can argue that our results are generalizable to all devices of both the platform.

Figure 4.7 shows that the GPS sensors has the same behavior, except for the MoSync C++ implementation on iPhone 4 and 5[3], but we also note that the same implementation obtained the worst result also for other sensors like proximity sensor (see Figure 4.5), orientation (sse Figures 4.6 and 4.10) and accelerometer (see Figures 4.3 and 4.8). Therefore this implementation is actually not efficient to retrieve data from sensors in all platforms.

The second result achieved is that the increment in energy consumption is higher when the application updates the user interface showing the data retrieved from the sensor, while it is lower, but still present, when the user interface is not updated (see Table 4.3 vs. Table 4.7 for the Samsung Galaxy, Table 4.4 vs. Table 4.8 for the Galaxy S5, Table 4.5 vs Table

[3]Data about energy consumption for iPhone 4 and 5 for GPS are quite superimposable.

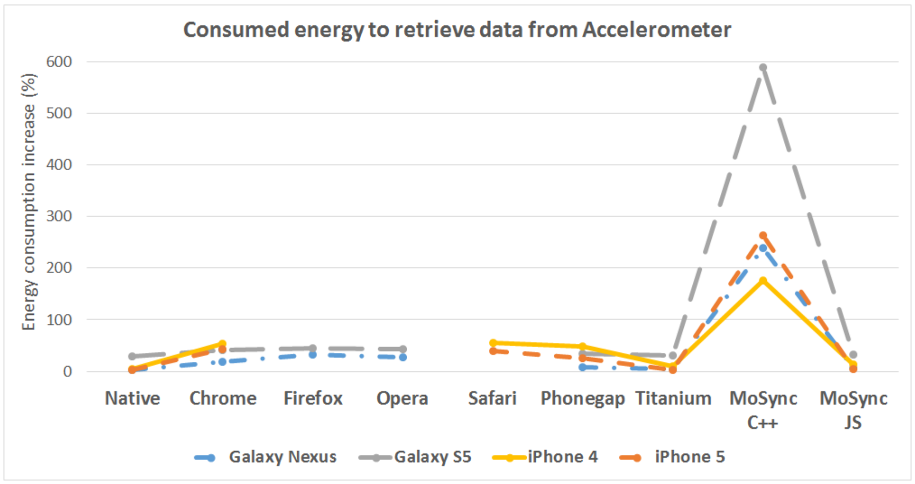


Figure 4.3: Consumed energy, expressed in terms of percentage increase, by an application which retrieves data from accelerometer.

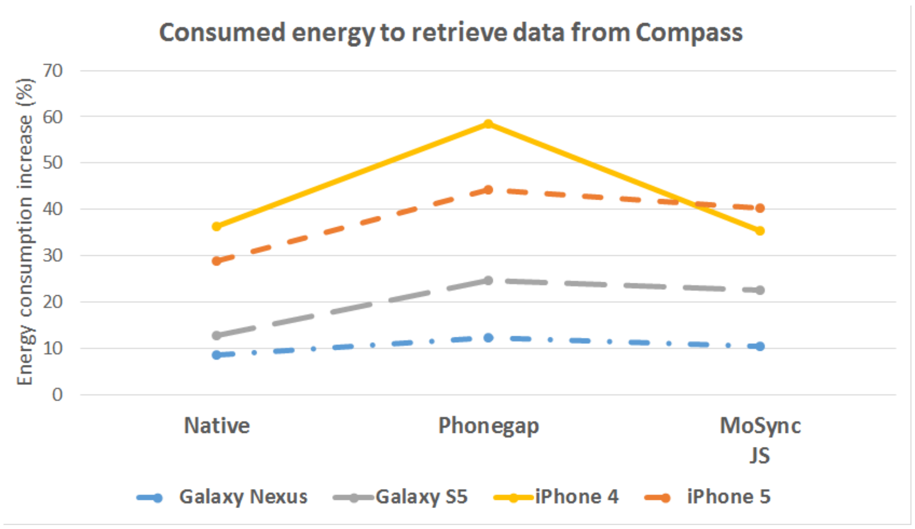


Figure 4.4: Consumed energy, expressed in terms of percentage increase, by an application which retrieves data from compass.

4.9 for iPhone 4 and Table 4.6 vs Table 4.10 for iPhone 5). This means that, for all these frameworks, the most expensive task is not to execute or interpret in real-time the code (as in the case of the Titanium framework), but the more expensive task is to update the graphical elements on the screen.

Let us consider, for example, the case of data acquisition from the accelerometer. If we compare Figures 4.3 and 4.8 we can notice that the difference in terms of consumed energy is particularly relevant for the *Web Approach* and the *Cross-Compiled Approach* using the C++ implementation of MoSync. Moreover, the difference can be higher: if we compare the difference between the native solution and the one using the PhoneGap framework in all the devices, the difference of the energy consumption increase required by PhoneGap

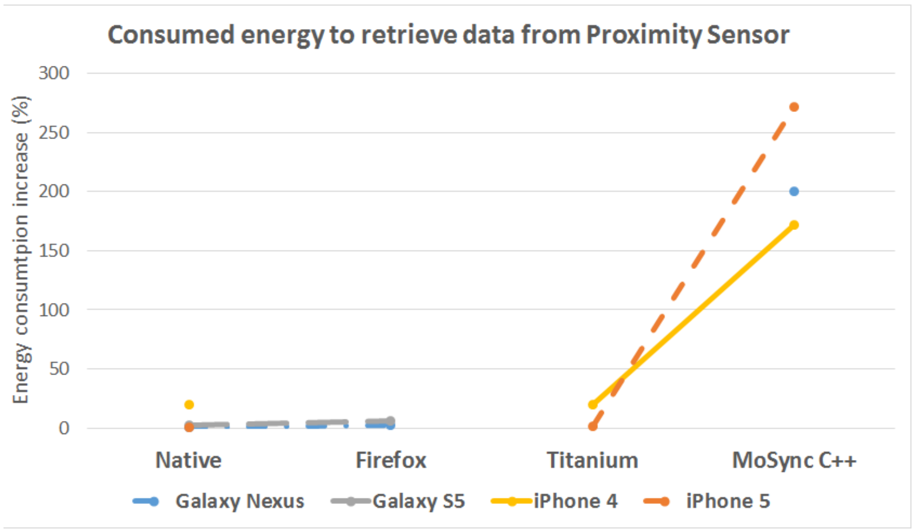


Figure 4.5: Consumed energy, expressed in terms of percentage increase, by an application which retrieves data from proximity sensor.

when updating the user interface with respect to the native solution varies between 38% and 71% (see Tables 4.3, 4.4, 4.5 and 4.6), while it lowers down between 2% and 3% (see Tables 4.7, 4.8, 4.9 and 4.10) when the user interface is not updated (and this is not one of the worst performing framework).

Considering the compass sensor, if we compare Figures 4.4 and 4.9 we can see that the curves trends are the same, but the curves are moved upper, since the consumed energy is augmented. This behavior deeply affects the measurement for the Galaxy S5, but it is also present for all the devices. Even in the case of data about device orientation, by analyzing Figures 4.6 and 4.10 we can see that the energy consumption increased, in particular for the *Cross-Compiled approach*.

| | | **Web App.** | | | **Hybrid App.** | **Interpreted App.** | **Cross-compiled App.** | |
|---|---|---|---|---|---|---|---|---|
| **Sensor** | **Native** | **Chrome** | **Firefox** | **Opera** | **PhoneGap** | **Titanium** | **MoSync (C++)** | **MoSync (Javascript)** |
| **Only App.** | +0,06% | +3,69% | +2,51% | +3,74% | +3,03% | +1,06% | +0,38% | +1,72% |
| **Accelerometer** | +3,04% | +18,05% $^{(50)}$ | +32,74% | +27,65% | +7,75% | +4,92% | +239,75% | +6,82% |
| **Device orient.** | +10,76% | +20,04% | +26,92% | +26,23% | - | - | +243,35% | - |
| **Compass** | +8,61% | - | - | - | +12,24% | - | - | +10,34% |
| **Proximity** | +0,90% | - | +2,00% | - | - | - | +200,65% | - |
| **Light** | +2,18% | - | +3,79% | - | - | - | - | - |
| **GPS** | +42,80% | +62,49% | +46,07% | +60,15% | +43,15% | +43,48% | +43,25% | - |
| **Camera** | +233,85% | +200,15%* | +208,26%* | +258,06% | +252,36% | +250,17% | +244,37% | +254,20% |
| **Audio record** | +20,56% | +48,97% | +53,37% | +49,22% | +21,76% | - | - | - |

Table 4.7: Data recorded during tests using the Samsung Galaxy without updating the User Interface with sensor data expressed in form of percentage. *: images from camera are not full screen but only a small section of the web page.

An interesting case study is represented by the MoSync framework. As already explained before, this framework follows the *Cross-Compiled Approach*, meaning that

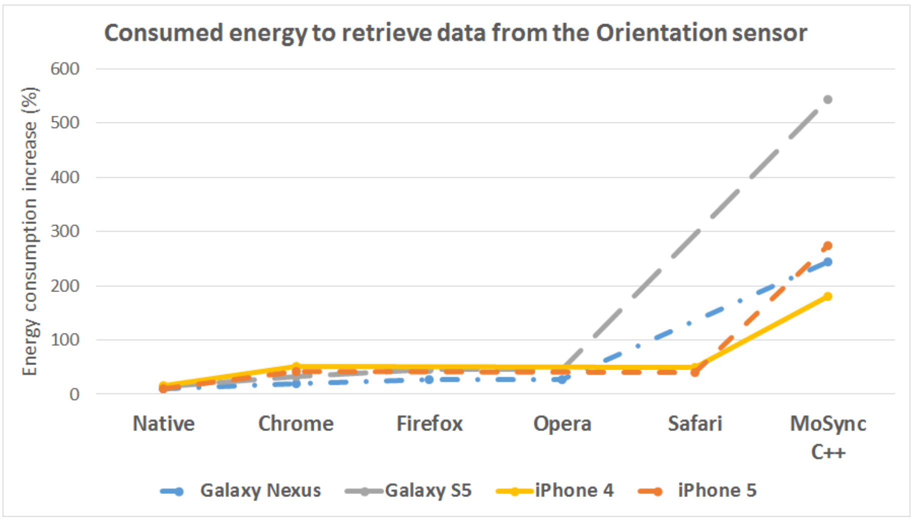


Figure 4.6: Consumed energy, expressed in terms of percentage increase, by an application which retrieves data about device orientation.

| | | **Web App.** | | | **Hybrid App.** | **Interpreted App.** | **Cross-compiled App.** | |
|---|---|---|---|---|---|---|---|---|
| **Sensor** | **Native** | **Chrome** | **Firefox** | **Opera** | **PhoneGap** | **Titanium** | **MoSync (C++)** | **MoSync (Javascript)** |
| **Only App.** | +0,09% | +3,17% | +2,11% | +3,67% | +2,74% | +1,28% | +1,01% | +1,98% |
| **Accelerometer** | +28,21% | +41,14% (50) | +44,32% | +43,37% | +33,74% | +30,14% | +588,86% | +32,14% |
| **Device orient.** | +13,08% | +33,15% | +46,36% | +45,58% | - | - | +542,65% | - |
| **Compass** | +12.74% | - | - | - | +24.54% | - | - | +22,63% |
| **Proximity** | +1,92% | - | +5,66% | - | - | - | +529,05% | - |
| **Light** | +6,59% | - | +53,82% | - | - | - | - | - |
| **GPS** | +80,56% | +93,43% | +84,26% | +97,84% | +82,69% | +83,1% | +81,7% | - |
| **Camera** | +544,95% | +219,17%* | +223,15%* | +214,61% | +557,05% | +552,24% | +551,49% | +563,89% |
| **Audio record** | +60,46% | +73,49% | +117,90% | +77,78% | +58,24% | - | - | - |

Table 4.8: Data recorded during tests using the Galaxy S5 without updating the User Interface with sensor data expressed in form of percentage. *: images from camera are not full screen but only a small section of the web page.

it generates a real native application, translating the code into real native code. MoSync supports two programming languages, C++ or Javascript. We tested both the possibilities.

Our experiments showed that the C++ implementation, when dealing with data retrieved from sensors, is much more expensive, both with respect to the Javascript implementation and to the native solution. The explanation of this result comes from the nature of C++: since it does not natively handle events (without using an external library), the final result is an application that uses *polling*, i.e., to retrieve data from sensors it implements an infinite cycle that runs for all the application lifecycle till its termination. This behavior is then translated into the final application and, from performances point of view, it is extremely energy expensive. This means that this kind of implementation could not be used "as is" when developing applications that retrieve and use data from sensors. A possible solution is to use it together with a C++ framework which provides support to events.

If we do not consider the C++ implementation of the MoSync framework, we can

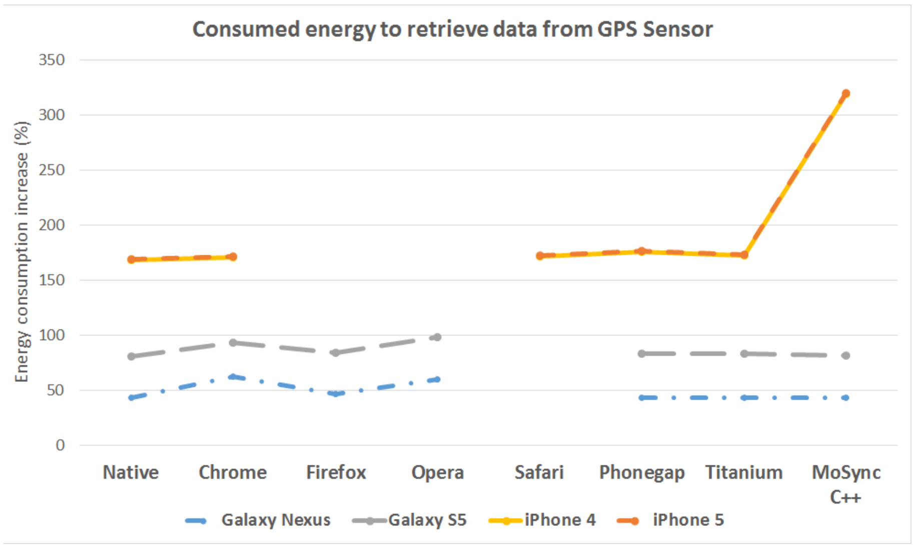


Figure 4.7: Consumed energy, expressed in terms of percentage increase, by an application which retrieves data from GPS. Note that data about measurements for iPhone 4 and iPhone 5 are quite superimposable.

notice that the *Web Approach*, where applications are rendered using a Web browser, is the most expensive one, even if it is used in offline mode, i.e., without being connected to the Internet, thanks to the new HTML5 features. There are different reasons to explain this result. The first one is the early stage of development of these features. The possibility to access smartphone sensors are far to be standard and fully supported by all browsers and devices so, at the time of writing, browsers and devices probably are not optimized for an heavy use, yet. Secondly, when a page requires sensor data, this request is made by the Javascript engine, that is interpreted by the browser and translated to a request into the native language of the OS, so consuming more energy. Finally, the first experiment showed that the execution of a web browser is more expensive than a simple native smartphone application, even without retrieving data from sensors. Therefore, since the *Web Approach* combines together all these aspects, it is clear that, in the end, the energy consumption of this particular approach is heavy and not suitable for smartphone applications.

Another interesting aspect that we can notice from the analysis of the data, is how the updating of the user interface affects the energy consumption of the frameworks. Let us consider, as an example, PhoneGap and Titanium frameworks used to develop an application which retrieves data from accelerometer, since this is the only sensor supported by both the frameworks. If we update the User Interface (see Figure 4.8), PhoneGap is less expensive than Titanium in the Android platforms and viceversa in the Apple platform. If we do not update the User Interface (see Figure 4.3), Titanium is less expensive than PhoneGap in both platforms. This means that the mapping between the Javascript code and User Interface elements made by Titanium during compilation and runtime is more

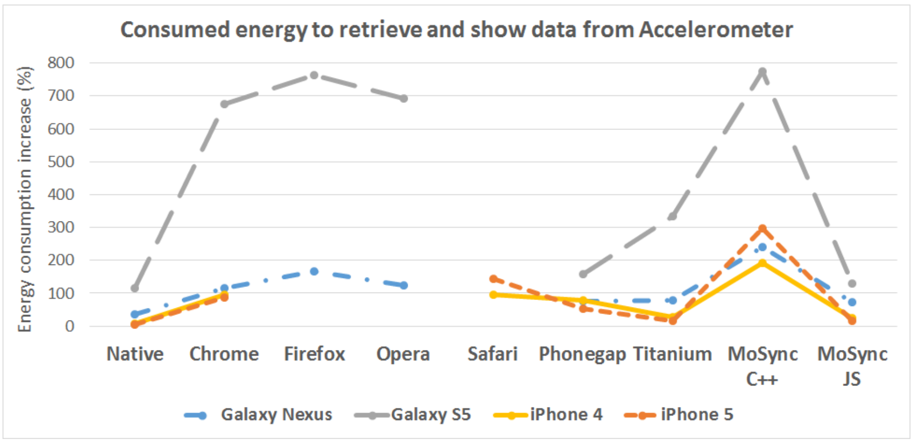


Figure 4.8: Consumed energy, expressed in terms of percentage increase, by an application which retrieves and shows data from accelerometer.

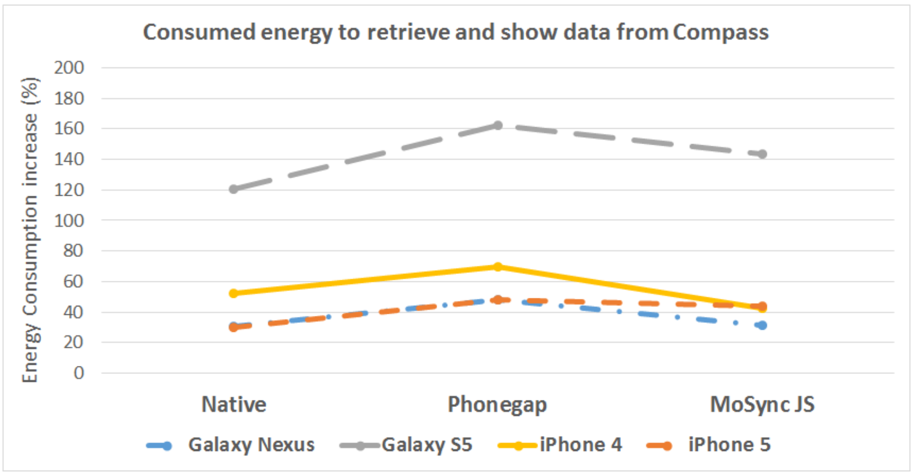


Figure 4.9: Consumed energy, expressed in terms of percentage increase, by an application which retrieves and shows data from compass.

expensive than the *Hybrid Approach* of PhoneGap in Android devices. If we analyze the frameworks PhoneGap and Titanium and their performances in both the platforms we can conclude that Titanium has better performances on the Apple devices, while on the Android platform, PhoneGap is sometimes better. This behavior takes two different considerations. The first one is that probably Titanium is much more optimized for the Apple platform instead of the Android one. The second one is that if we must choose between Titanium or PhoneGap as developing framework, it is not possible to find the best solution in terms of energy consumption. This means that the choice should consider how users are distributed among the different platforms, and choose the developing framework depending on this distribution (and this is against the idea of cross-platform development). Since cross-platform frameworks can have different behaviors in different platforms in terms of energy consumption, we think that their current state-of-art does not guarantee a

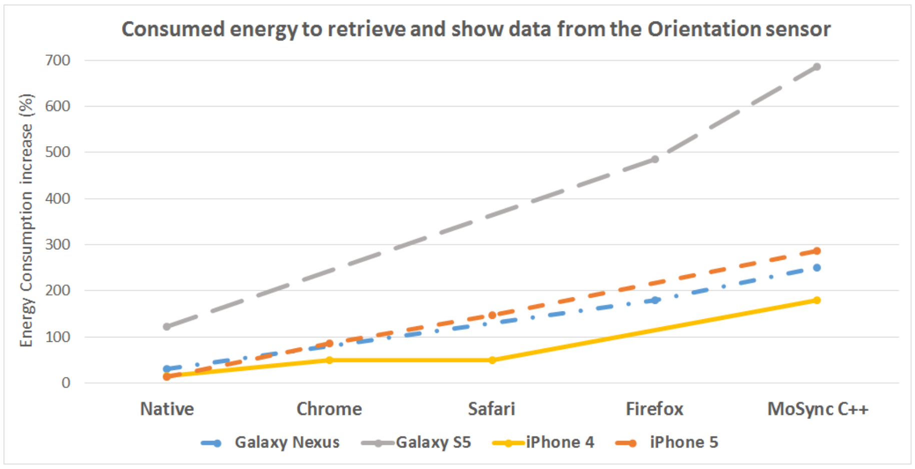


Figure 4.10: Consumed energy, expressed in terms of percentage increase, by an application which retrieves and shows data about device orientation.

real cross-platform development.

However, the choice of which framework to use must take into account the number and type of supported features, since cross-platform frameworks do not support all the native features, e.g., browsers do not support the compass sensor (i.e., the *Web Approach*), Titanium and MoSync (using C++).

If we must choose a winner for the best framework among the considered ones, it is clear that MoSync, using the Javascript implementation, when available, is the one with best performances in terms of energy consumption. The resulting code and final application leads to better results with respect to other approaches like the *Hybrid* or the *Interpreted* one, where code is executed in an encapsulated environment (a web browser) or where the code is interpreted at runtime. It performs better than the C++ implementation of the same framework due to the fact that C++, as already discussed before, is not able to handle alone the events efficiently, meaning that this implementation of MoSync may improve using a library which handles the events. The Javascript implementation is not affected by this problem and so the results are better.

Even if this result could seem to be predictable, since the final application is a native application, therefore there are no software layers that can lower performances, this is not true since the experiments made with the C++ implementation of MoSync showed that the performances, in terms of power consumption, are deeply affected by the quality of the compilation process that translates the application from the language used for development into the native language of the platform. Moreover, this analysis allows to measure the impact of this framework.

Details about data recorded during each single experiment are reported in Tables 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9 and 4.10. Tables 4.3 and 4.4 present the results obtained with the two Android smartphones when the developers need to update the User interface as additional percentage of consumed energy with respect to the quiet state of the smartphone. The update frequency is 64ms for frameworks that allows to define the update frequency.

Tables 4.7 and 4.8 present the results with the Android device without updating the User Interface, i.e., the application only retrieves data from the considered sensors. Tables 4.5 and 4.6 present data obtained with Apple devices, updating the User Interface with sensors data and Tables 4.9 and 4.10 present data recorded with the iPhones when the User Interface is not updated.

| | | **Web App.** | | **Hybrid App.** | **Interpreted App.** | **Cross-compiled App.** | |
|---|---|---|---|---|---|---|---|
| **Sensor** | **Native** | **Chrome** | **Safari** | **PhoneGap** | **Titanium** | **MoSync (C++)** | **MoSync (Javascript)** |
| **Only App.** | +0,78% | +2,91% | +2,06% | +2,50% | +1,53% | +2,49% | +3,49% |
| **Accelerometer** | +4,93% | +54,28% (50) | +55,94% (50) | +47,69% | +9,03% (100) | +175,50% | +12,92% |
| **Device orientation** | +15,21% | +50,4% (50) | +49,89% (50) | - | - | +180,05% | - |
| **Compass** | +36,36% | - | - | +58,48% | - | - | +35,27% (140) |
| **Proximity** | +19,55% | - | - | - | +20,21% | +171,97% | - |
| **GPS** | +167,81% | +170,19% | +171,54% | +175,43% | +171,85% | 318,42% | - |
| **Camera** | +254,39% | - | - | +314,78% | +315,81% | +306,46% | +286,46% |
| **Audio record** | +7,51% | - | - | +14,91% | +15,40% | - | - |

Table 4.9: Percentage of energy consumption increase recorded during tests with the iPhone 4 without updating the UI with sensor data.

| | | **Web App.** | | **Hybrid App.** | **Interpreted App.** | **Cross-compiled App.** | |
|---|---|---|---|---|---|---|---|
| **Sensor** | **Native** | **Chrome** | **Safari** | **PhoneGap** | **Titanium** | **MoSync (C++)** | **MoSync (Javascript)** |
| **Only App.** | +1,69% | +2,17% | +3,85% | +3,88% | +3,26% | +4,83% | +5,34% |
| **Accelerometer** | +2,56% | +42,53% (50) | +39,98% (50) | +25,18% | +2,47% (100) | +263,94% | +4,91% |
| **Device orientation** | +10,19% | +41,81% (50) | +39,92% (50) | - | - | +274,24% | - |
| **Compass** | +28,75% | - | - | +44,23% | - | - | +40,25% (140) |
| **Proximity** | +0,33% | - | - | - | +1,58% | +271,96% | - |
| **GPS** | +168,68% | +171,57% | +171,94% | +176,33% | +172,74% | 319,79% | - |
| **Camera** | +271,76% | - | - | +321,67% | +311,38% | +286,52% | +296,70% |
| **Audio record** | +11,42% | - | - | +15,73% | +21,34% | - | - |

Table 4.10: Percentage of energy consumption increase recorded during tests with the iPhone 5 without updating the UI with sensor data.

## 4.4 Final remarks

The results described so far can be read in different ways, some correct and some incorrect. The experiments showed that energy consumption performances of cross-platform frameworks are still far from the native approach. Therefore, these frameworks are currently difficultly adoptable when the applications acquire data from sensors (a condition that is often true especially for ubiquitous applications) and update the User Interface using these data.

This problem is difficult to solve, and will require a lot of work in the next years, but it is not correct to state that cross-platform frameworks cannot improve in the future. In fact, the classification of frameworks, described in Section 2.1 of this thesis, shows the overall architecture of the frameworks thus highlighting also which are the components that consume more energy, since our experiments showed that the more consuming task is the User Interface rendering.

If we consider the *Web Approach*, it requires a browser for rendering the application. Therefore, these frameworks can improve their energy consumption only if the future implementations of the browsers become more efficient. The *Hybrid Approach* needs a WebKit engine, so a component of a browser, for rendering. Even in this case, improvements in energy consumption do not depend on the frameworks implementation but on the implementation of the WebKit engine.

A different situation is represented by the other two classes of frameworks: in this case they do not rely on external components for rendering, therefore they can improve their energy consumption by improving their implementation. The *Interpreted Approach* contains a real-time interpretation step. This software layer increases the amount of CPU operations with respect to the native approach, thus requiring more energy, but research can contribute to lower this amount of CPU operations.

We consider the *Cross-Compiled Approach* the most promising one, so we suggest to use it for implementation of new frameworks. But, it is crucial that the development of this kind of framework focus its attention on producing application code that is efficient and with performances comparable to native code. In particular, developers of this type of frameworks should pay special attention to the optimization of events manager. Otherwise, a “bad” implementation can waste the advantage of this class of frameworks, as in the case of the C++ implementation of MoSync. Instead, the Javascript implementation of MoSync shows that this goal is achievable.

# III Emotion assessment using smartphones

# 5. Smartphone interaction for stress assessment

The analysis described in the previous Chapter showed how actually cross-platform frameworks are not the best solution for mobile applications, both because they do not provide all the features available to native applications, and even because their energy consumption is higher and negatively affects user experience. In particular, less features available means that applications developed with a cross-platform framework have access to a smaller set of data that would be natively available, while higher energy consumption means a worst user experience, since it would require the user to charge his/her smartphone with higher frequency, making the application too intrusive.

As mentioned before, thanks to smartphone sensors, i.e., accelerometer, barometer, rotation sensor, etc., it is possible to acquire a huge amount of data and information. This data can be used for several interesting purposes. For example, it can be used to infer user activity like walking, running, etc., using data from the accelerometer or the barometer. Otherwise, as presented by Picard in [114], computers (and smartphones) can be used to understand and infer user moods, i.e., happiness, sadness, stress, etc. The use of the smartphone for activity recognition will be discussed in Chapter 7. Here, we focus our attention on users' mood analysis, in particular, to stress assessment. The problem of stress experienced during life is a very well known problem. Even if experiencing stress among the entire life is normal for short periods of time, long exposure to stress can interfere with normal life and become unhealthy. It can lead to an inability to concentrate or sentiments like irritability, anxiety, depression, etc. [33], which may strongly increase the probability of the appearance of some diseases, and so the rise of health care costs

[112]. Several research papers already showed that some physiological traits like heart rate, skin conductance, etc., change between relaxed and stress states. The easiest approach to constantly monitor stress levels of people during the entire life, as already shown in several research papers [123] [129], is to use portable and external sensors like portable ECG monitors, smart watches or skin conductance monitors to collect data and analyze user emotional state. But people usually feel uncomfortable when obliged to wear such medical devices.

Since our research focuses especially on smartphone and how to reduce the intrusiveness of health systems in people' life, we explored the possibility to use only smartphones, without the usage of any other external sensors, for stress assessment. The usage of only the smartphone for data collection and analysis strongly reduces the intrusiveness, thus increasing user acceptance, of the system, since smartphones are already present people' pockets and accompany them in everyday life.

Differently from other approaches, that use privacy-related information like calls, SMSes, location, etc., we focused our attention on human-smartphone interaction. In particular, we started from the idea that when we are stressed, our interaction with the smartphone differs from when we are relaxed. Moreover, since the analysis of how we digit, 'tap', 'scroll' or 'swipe' on the smartphone is something absolutely not intrusive, even user acceptance of this kind of possible constant monitoring is much higher with respect to an application that monitors the number of messages or calls made among the days, or event with respect to external sensors attached to the body. As we will see later, we focus our attention on common interactions made with smartphone, e.g., 'touch', 'scroll', 'swipe' and writing messages.

Part of this chapter has been published in [];

## 5.1 Related works

Stress detection and monitoring using external sensors and smartphones has gained a lot of interest from the research community, due to the increasing diffusion of this mental condition and the possibility to use non-intrusive sensors and devices to monitor people during their everyday life activities.

Gunawardhane et al. [67] proposed stress analysis and detection using only a keyboard, searching differences in how the user digits on a keyboard, during stress or non-stress moments. They asked 20 students to write several sentences using Microsoft Word during an exam period or a free period. They analyzed several features related to digits, i.e., time distance between two digits, error rate, etc. To create their ground-truth, they used a pulse wave sensor to capture the heart rate variability.

Paredes et al. [110] initially proposed the usage of ubiquitous technologies in affective computing and stress management. In particular, they showed the most important and positive aspects of this technology: *accessibility*, *unobtrusiveness*, *long-term and in-situ monitoring* and *application and content neutral monitoring*. They divided their presentation into two different situations and available sensors: static and mobile environment. The static environment refers essentially to a desktop computer usage, where the main sensors that could be used are mouse, keyboards, and camera. In the mobile environment, the smartphone becomes the unique device used for data acquisition, using gesture and voice as main input. The main positive aspect of this solution is user acceptance, since no other external or wearable devices are used to monitor stress, and common usage devices become the main sensors, without affecting people behavior.

Sun et al. [130] explored the possibility to identify stress in people analyzing common computer mouse operations and building a model of the hand and muscles with a *mass-spring-damper* system. They asked users to perform three different activities: *Point-and-click* (click inside two target rectangles), *Drag-and-drop* (move one rectangle over another), and *Steer* (draw a line inside a tunnel as much linear as possible). Each user had to perform these tasks both in a relaxed and a stressed state. They concluded that stress measures can be more indicative using mouse-derived metrics than electrocardiogram signal analysis, in particular when considering the within-subject model. Moreover, they showed that a small amount of data (about 10 minutes of interaction for the user in a given stress state) can be used to infer stress state with 70% of accuracy.

Hernandez et al. [72] evaluated the possibility to use keyboards and mouses to detect stress in users. They used a pressure-sensitive keyboard and a capacitive mouse, focusing analysis on the touch pressure. The tasks that each of the 24 participants had to perform were: text input (pre-specified text, as fast as possible), expressive writing (writing memories for 5 minutes) and mouse clicking (clicking on different horizontal bars of different sizes). The results showed that stress significantly influences the keyboard touching pressure, in more than 83% of the users. Considering the mouse, the pressure intensity was not significant, but the amount of contact area increased during stress moments.

Sano and Picard [125] researched physiological or behavioral markers for stress; they collected data from an accelerometer, GSR (Galvanic Skin Response) sensor and mobile phone usage. In particular, they collected data about screen ON/OFF, SMSes (length, number of receivers, number of messages, etc.), calls (person called, duration, etc.) and the location of the user. With a precision of 75%, they were able to discriminate stress/no-stress state using information about when the screen is ON, mobility data, calls data and user activity (sitting, walking) information. This is, however, a very privacy-sensitive context.

The Interstress project [117] aims at helping people to gain information about their personal stress level and help therapists to follow their patients during their treatment. The system is essentially based on a mobile system that uses a smartphone and external devices, i.e., heart and respiratory rate sensor, to acquire information about the user. This raw data is then elaborated by the smartphone to extract information about activity level, location or number of messages and calls. This data is then sent to a server that adds other historical and personal data of the user to extract stress levels. The mobile application also provides a stress treatment application that let the user fight stress using relaxing landscapes and environments.

Bauer and Lukowicz [14] researched behavior changes during stress and no-stress periods using a smartphone, leveraging data about the location of the user (using GPS and WiFi), his/her social interactions (using Bluetooth sensor), and calls/SMSes patterns. They involved seven students during two exams weeks (stress period) and the following two holiday weeks (no-stress period). By combining all the features of social interaction data, SMSs behavior, and call patterns, they were able to detect a change in behavior of about 86% of testers for stress/no-stress situations.

Lee at al. [93] proposed an emotion recognition system for social network posting able to recognize seven different emotions while the user is typing a post for Twitter. Their input data considers typing speed, backspace frequency, long touch counter, illuminance, location, weather, etc. They used a final number of 10 different features, building a Bayes Network classifier with a final accuracy of 67.52%. For emotions such as happiness, surprise and neutral the performances were appreciably better than chance, while for anger and disgust the model outperformed, at least, random classification.

Even if for a different purpose, Gao et al. [64] analyzed if touchscreen interaction can be used to know its user's emotional state (Excited, Relaxed, Frustrated and Bored). They collected data about finger stroke behavior during gameplay with an iPod touch game. The main data collected was: coordinates of each stroke, the contact area and the total time needed to complete the action. Accuracy in discriminating the four different emotional states was between 69% and 77%, while higher accuracy (89%) was achieved when discriminating between two levels of arousal and two levels of valence.

Differently from other approaches, we leveraged only human-smartphone interaction data to assess the stress state of an individual. We analyze only data from 'touch', 'scroll', 'swipe' gestures and 'text input' behavior, without attaching any external device to the user or the smartphone, thus lowering system intrusiveness. Moreover, no privacy-sensitive information are used with our approach, like user's calls, SMSes or location data. To the best of our knowledge, this is the first research that connects common gestures like 'tap', 'swipe' and 'scroll' with stress condition in users.

## 5.2 Protocol design

We designed our experiment to study the possibility of using information about the interaction that takes place between the user and his/her smartphone and accurately assess his/her stress state. The smartphone is instrumented for measurement and collection of interaction data. Participants completed several exercises under the relaxed (no stress) and stressed state while being at home or in another comfortable and static (i.e., controlled) environment. This section provides details about the protocol, the input data collected and which kind of interactions were analyzed.

### 5.2.1 Human-Smartphone Interaction

Every time we pick up our smartphone to perform usual everyday activities, e.g., make a call, write an SMS, read/write an email, what we essentially do, independently from the goal we are fulfilling, is that we are interacting with the smartphone. A smartphone gesture is defined as any contact with the screen that causes a change in the smartphone interface and its internal state. The main gestures are: 'tap', 'double tap', 'long press', 'scroll', 'swipe', 'pinch', 'zoom' and 'rotate'. Some of them can be considered as "application dependent", with much lower daily occurrence than others. For this reason, we decided to focus only on the interactions that occur much more frequently and that are independent from the currently running application: 'tap', 'scroll' and 'swipe'.

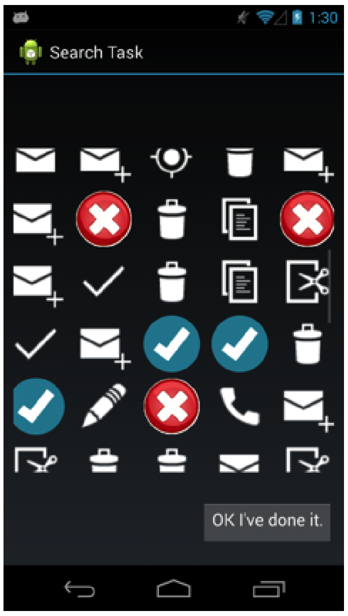
Another common task performed with the smartphone is 'text input', i.e., writing SMS or email. As shown previously [72], keyboard typing can be leveraged as an indicator of stress. We expand this idea considering smartphone typing.

### 5.2.2 Test Tasks

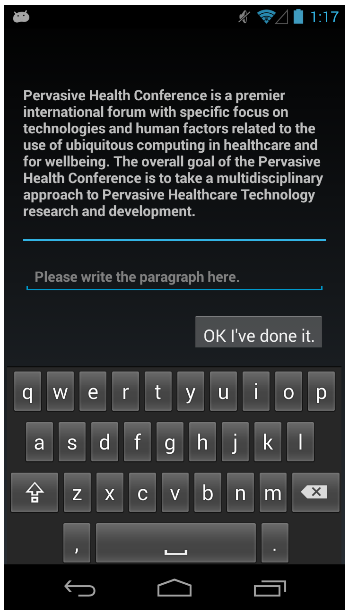
To acquire information about 'tap', 'scroll', 'swipe' and 'text input' interactions, we developed several exercises (called *Tasks*) requiring the participants to perform pre-defined set of actions: the *Search Task* and the *Write Task*.

#### Search Task

In this task, the participant has to search all the repetitions (30) of three randomly chosen icons on the smartphone screen within a total of 300 icons (there are ten different icons used). Since it is not possible to show all the icons on the screen at the same time, the participant has to 'scroll', i.e., execute a vertical movement on the screen, and 'swipe', i.e., execute a horizontal movement on the screen, to be able to inspect all the icons. Every time he/she clicks on one icon, there is a 'tap' interaction and immediate feedback is provided about the correctness of the choice. Figure 5.1a presents a screenshot of the application during the *Search Task*.



(a) *Search Task*



Pervasive Health Conference is a premier international forum with specific focus on technologies and human factors related to the use of ubiquitous computing in healthcare and for wellbeing. The overall goal of the Pervasive Health Conference is to take a multidisciplinary approach to Pervasive Healthcare Technology research and development.

(b) *Write Task*

Figure 5.1: *Search* and *Write Task*

### Write Task

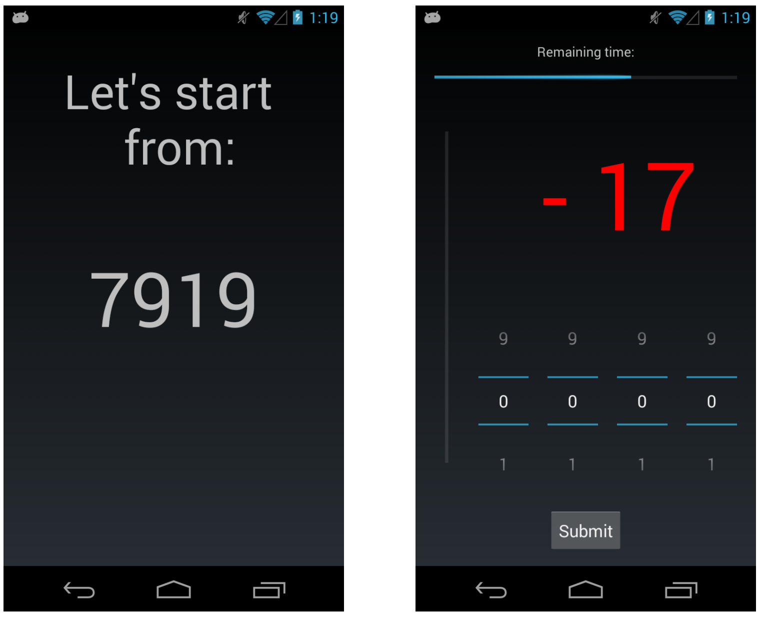
In this task, the participant is required to rewrite a paragraph that is shown on the screen. We decided to use English as text language, even with no native English speakers, since it does not have accents or other language-specific elements. Each sentence was composed only of words, commas, and dots; the most used elements when writing SMSes or emails. The participants used our custom virtual keyboard. In this way, we ensure the same test conditions for all participants, i.e., some smartphone users have a custom keyboard to write, instead of the standard QWERTY one. Our keyboard does not provide auto-correction or word suggestion, giving us the possibility to collect data about errors and user-based corrections during writing. During this task, we collect 'tap' and 'text input' data. A screenshot is provided in Figure 5.1b.

## 5.2.3 Stressor Task

To measure the differences in interaction between a relaxed and a stress state, we stressed the participant through a particular task. Several papers in literature address the problem of how induce stress in people [16] [44]. The main stressors that can be used are:

- *Cognitive stressor*: mathematical and memory tests, e.g., starting from a big prime number and going down by 7 or 13, without the possibility of taking notes;
- *Social pressure*: evaluation of the performance of the individual, in particular by an external person, e.g., via public speech;
- *Timing pressure*: giving a maximum amount of time to complete the task;

- *Random events*: generation of random events that could disturb the main user task, i.e., simulation of faults, unexpected results, etc.



(a) Starting prime number (b) Stressor task interface

Figure 5.2: *Stressor Task*

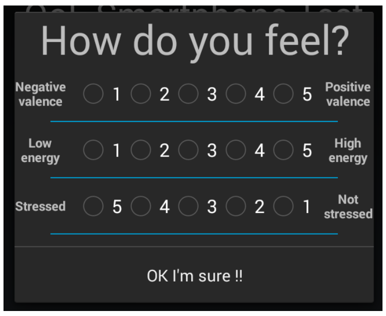
We have built a smartphone task that combines all the stressors mentioned above. Therefore, the participant is asked to make several mathematical calculations and to input the answer using the provided interface. We start from a random big prime number, and the user has to calculate the subtraction of this number by 7 or 13 (randomly). To submit the answer, the participant has a limited time. If he/she submits the correct answer, he/she will have to continue the task, and the available time will be further decreased. If the submitted answer is wrong, an annoying sound is played to underline the error, and the smartphone vibrates. After an error, the participant will have to restart from another prime number provided by the task, and we increase the available time to answer. Every correct answer let the participant earn points (that increase with the number of consecutive correct answers submitted), while wrong answers will decrease the current score. There is a predefined amount of points that let the participant finish the task (after a minimum amount of time).

To increase the stress of the individual, several random events can happen. For example, if more than four correct answers are submitted, the decrease number will change, randomly choosing over different possibilities, i.e., 5, 19, 21, etc. Furthermore, every two minutes there is a random possibility that the level of earned points necessary to stop the *Stressor Task* and proceed to the next step, are decreased by 10%.

The *Stressor Task* lasts, at least, five minutes, and ends if the participant has reached the minimum score necessary to proceed, or after 10 minutes, independently from the achieved score (this information is not provided to the participant). We used a maximum time for this task to avoid possible dropouts, i.e., avoid that participants decide not to continue the protocol because they are too frustrated with it. Figure 5.2 provides two screenshots of the *Stressor Task*.

### 5.2.4 Stress Measurement

To evaluate whether the two modalities of the tasks (no-stress and stress mode) elicited the intended stress in participants, each participant had to report his/her emotional state after each step of the protocol using the Experience Sampling Method (ESM) [91]. The inputs requested were: Valence, Energy (Arousal) and Stress levels on a 5-point Likert scale [95]. The provided survey is presented in Figure 5.3. From this survey, we expect that the higher stress values are associated with tasks in "stress mode", while more relaxed ones with the "relaxed tasks". Stress condition is usually associated with negative valence and higher energy.


How do you feel?

| | | | | | | |
|---|---|---|---|---|---|---|
| Negative valence | 1 | 2 | 3 | 4 | 5 | Positive valence |
| Low energy | 1 | 2 | 3 | 4 | 5 | High energy |
| Stressed | 5 | 4 | 3 | 2 | 1 | Not stressed |

Figure 5.3: ESM dialog prompted after each step of the protocol.

### 5.2.5 Protocol Overview

The protocol is divided into four different phases:

- "Relax phase": participants experience some relaxing music with several relaxing images on the screen [39]. This step is used to relax participants and baseline their stress state;
- "Calm phase": in this phase participants perform the *Search* and *Write Task* in a relaxed state. Each task is repeated three times to collect a sufficient amount of 'scroll', 'swipe' and 'text input' data. This phase has a variable duration, depending on the speed of the participant in finding the icons or writing the text;
- "Stressor phase": in this phase participants experience the stress-inducing task. It will last between 5 and 10 minutes, as explained in Section 5.2.3;
- "Stress phase": the participants are now stressed from the previous task and they repeat the *Search* and *Write Task* in a "stress state". Additionally, other stressors

are applied, i.e., a tic-tac sound is played. The participant has to find all the possible icons, or to write the text, before the time expires. Moreover, for the *Search Task*, if a wrong icon is selected, the available time is decreased and the smartphone vibrates, while if a correct one is selected, the time is increased. The usage of these stressors during the stress phase aims at keeping participants constantly stressed over all the second part of the protocol.

Figure 5.4 shows the overall protocol.

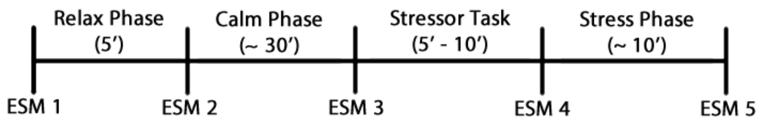


Figure 5.4: Overview of the study protocol.

### 5.2.6 Participants

Participants for this study were recruited through an email sent to several mailing list and individuals already enrolled in other studies at the "Quality of Life" group, University of Geneva. Participants were informed that our research focused on building better user interfaces for smartphones in collaboration with Google. We did not inform the participants about stress analysis and recognition to avoid a bias of their attitude and their interaction styles. In the email, we defined the requirements to participate in the study; the only one was to be an Android OS smartphone owner and user and to have one hour of not interrupted time to follow the protocol from the beginning to the end. The approximate duration of the whole experiment was about one hour, depending on the skills/speed of the participant using his/her smartphone, in particular during the *Write Task*.

In the recruitment email, we explained all the steps necessary to start the application to execute the protocol and to correctly complete it. Individuals that decided to participate first needed to download and install the application following the provided link. Before starting, they had to prepare themselves to correctly perform the exercises: be sure that they would not be disturbed for the following hour, sit on a chair where they could be comfortable, hold the smartphone as naturally as possible (keeping it in hand and not putting it on a table) and since there are associated sounds (music) in specific tasks, we have additionally advised participants to use an headset. After the test phase, we extract three different participants to receive a 50 CHF Amazon Gift Card to thank them for their participation.

In total, 13 participants (7 males and 6 females) answered to our participation request and completed the experiment. The average age was 26,38 ($\pm 2,53$) with a minimum of 22 and a maximum of 32 years old. The participants differed in level of skill in English speaking/writing and smartphone usage. We let the participants use their smartphone to avoid that a new one during the experiment could stress them.

## 5.3 Data collection

### 5.3.1 Object and action

Along each task of the protocol, the main data we collect was the position of the object(s) relevant to the task on the smartphone screen. The information gathered is stored as an n-tuple:

*(ID, type, x, y, width, height, visibility, text)*

where:

- *ID*: is the ID of the object;
- *type*: the type of the object, i.e., button, icon, text, etc.;
- *x*: x-coordinate of the top left corner of the object;
- *y*: y-coordinate of the top left corner of the object;
- *width*: the width of the object (in pixels);
- *height*: the height of the object (in pixels);
- *visibility*: if the object is visible or hidden;
- *text*: the text of the object (if any).

Given this information, together with information about the device type, i.e., model, screen size and resolution, we are able to reconstruct the state of the screen while the participant was performing the tasks, and understand for example how a participant's clicks' on the screen are related to the position of the objects on the screen.

Every touch interaction, that could be a 'tap', 'scroll', 'swipe' or 'text input' is made up by a set of *action* event. The first action event has its flag field set to "ACTION_DOWN", indicating a press of the finger on the screen. After that, there are zero or more "ACTION_MOVE" actions and the final "ACTION_UP" flag that indicates the end of the interaction. This set is the same for all the possible interactions we are recording. An example of the sequence of *action* readings for the 'scroll' interaction is provided in Figure 5.5.

Each action is correlated with a set of information, and is stored as an n-tuple:

*(action, timestamp, x, y, size, pressure, firstTimestamp)*

where:

- *action*: a flag to distinguish between "ACTION_DOWN", "ACTION_MOVE" or "ACTION_UP";
- *timestamp*: timestamp of the recorded *action* (in milliseconds);

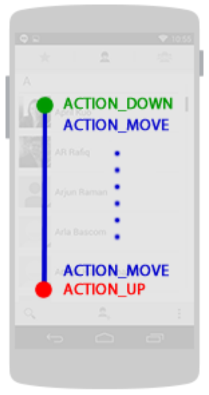


Figure 5.5: Example sequence of *action* readings during a 'scroll' interaction.

- $x$: x-coordinate of the center of the touch point on the screen from the top left corner (in pixels);
- $y$: y-coordinate of the center of the touch point on the screen from the top left corner (in pixels);
- *pressure*: the pressure applied with the finger, where pressure $\in ]0,1] \in R$ (unit less, granularity $0,01$);
- *size*: a measure of the contact surface of the finger with the screen, where size $\in ]0,1] \in R$ (unit less, granularity $0,01$);
- *firstTimestamp*: the timestamp of the first action ("ACTION_DOWN") of the interaction.

One may notice that 'text input' and 'tap' interaction should have no "ACTION_MOVE" flag, since there is no movement with the finger between the press (i.e., "ACTION_DOWN") and the release (i.e., "ACTION_UP") of the finger on the screen, while 'scroll' and 'swipe' interactions have a variable number of them. However, we noted that even in the case of a 'tap' there is the possibility that the participant makes a small movement, not releasing the finger in the same point where he/she initially started to touch on the screen. We also investigated if this movement is significantly different during stress state for each individual.

### 5.3.2 Feature Extraction

Once all the 'object' and 'action' data was collected from participants, we extracted several features from the *Search* and the *Write Task*.

#### Search Task features

For the *Search Task*, we collect features both for the 'tap' interaction and for the 'swipe' and 'scroll' ones. For the 'tap' interactions, i.e., when clicking on icons, we considered standard features like *average pressure*, *average size*, *movement* (how much the 'tap' moves when touching on the screen) and *time duration*. For the ‘scroll’ and ‘swipe’ interaction, we were interested in features that could describe in more details these kind of interactions with the smartphone. In particular, we calculated information about interaction *time* and *space length*, interaction *speed* ($\frac{space\_movement}{time\_length}$), mean distance of the interaction from the center or the top left corner of the screen and the overall *linearity* of the interaction.

| **Feature** | **Description** |
|---|---|
| **Tap pressure** | Pressure applied when ‘tapping’ on icons on the screen (value $\in ]0,1] \in R$) |
| **Tap size** | Size of the touch when ‘tapping’ on screen (value $\in ]0,1] \in R$) |
| **Tap movement** | Movement of the touch when ‘tapping’ on icons (in pixels) |
| **Scroll/Swipe Average pressure** | Average pressure applied while ‘scrolling’/‘swiping’ (value $\in ]0,1] \in R$) |
| **Scroll/Swipe average size** | Average size of the touch surface while ‘scrolling’/‘swiping’ (value $\in ]0,1] \in R$) |
| **Scroll/Swipe delta** | Space movement of the ‘scroll’/‘swipe’ (in pixels) |
| **Scroll/Swipe interaction length** | Space length of the interaction used to ‘scroll’/‘swipe’ (in pixels) |
| **Scroll/Swipe delta speed** | Speed of the ‘scroll’/‘swipe’ movement (in pixels / milliseconds) |
| **Scroll/Swipe interaction speed** | Speed of the interaction movement to ‘scroll’/‘swipe’ (in pixels / milliseconds) |
| **Scroll/Swipe distance from screen center** | Distance of ‘scroll’/‘swipe’ from the center of the screen (in pixels) |
| **Scroll/Swipe distance from top left corner of the screen** | Distance of ‘scroll’/‘swipe’ from the top left corner of the screen (in pixels) |
| **Scroll/Swipe linearity** | Linearity of ‘scroll’/‘swipe’ as horizontal/vertical distance between first and last point (unit less) |
| **Scroll/Swipe linearity as sum of every point** | Linearity of scroll/swipe as sum of the horizontal/vertical distance between two consecutive points (unit less) |

Table 5.1: Features derived for the *Search Task*

We must note here that a *scroll/swipe interaction* is different from a *scroll/swipe movement*. The first is the action performed by the participant to ‘scroll’/‘swipe’, meaning the sequence of press on the screen (“ACTION_DOWN”), movement on the screen (“ACTION_MOVE”) and the final release (“ACTION_UP”). Each *scroll/swipe interaction* generates the sequence of actions and n-tuples presented before, and the features referred to the *scroll/swipe interaction* refers to data calculated from this set of actions.

The *scroll/swipe movement* is the movement that occurs on a scrollable object on the screen, i. e., the movement of the interface, that changes the visible elements on the screen according to the direction and the speed of the ‘scroll’/‘swipe’. The *scroll/swipe movement* is the resulting interface movement that generates data about how much the interface is changing on the screen.

The linearity of the ‘scroll’ or ‘swipe’ interaction is derived in two different ways. The first way is the following one:

$$\text{Scroll linearity} = \frac{|x_l - x_f|}{W} \text{ and Swipe linearity} = \frac{|y_l - y_f|}{H}$$

where:

- $x_f$ and $y_f$ are respectively the $x$ and $y$ coordinates of the first point of the ‘scroll’/‘swipe’ (in pixels);

- $x_l$ and $y_l$ are respectively the $x$ and $y$ coordinates of the last point of the 'scroll'/'swipe' (in pixels);
- $W$ and $H$ are respectively the smartphone screen width and height (in pixels).

Since a 'scroll'/'swipe' interaction is a set of points $S = \{(x_{1..n}, y_{1..n}) | n \in N\}$, it is possible to calculate the linearity of a scroll/swipe interaction as sum of every individual point. Based on which, we can calculate the linearity as:

$$\text{Scroll linearity} = \frac{\sum_{i=2}^{n} |(x_i - x_{i-1})|}{W}$$

and

$$\text{Swipe linearity} = \frac{\sum_{i=2}^{n} |(y_i - y_{i-1})|}{H}$$

Since we let users use their smartphone, to cope with different screen resolutions, all features related to space length of the 'scroll' or 'swipe' interaction are normalized with the width or height resolution of the screen.

Table 5.1 presents all the features calculated from the data acquired during the *Search Task*. The first three features are calculated for each 'tap' interaction (when clicking on a particular icon on the screen). For some rows of the Table, when we write "Scroll/Swipe ..." means that the feature is calculated (separately) both for the 'scroll' and 'swipe' interaction.

### Write Task features

In addition to the standard features about 'tap' interaction already explained for the *Search Task* (*Pressure*, *Tap size* and *Tap movement*), with the *Write Task* we employ other features that are related to 'text input'. Specifically, we investigated the *number of errors*, the *number of corrections* and the *digits frequency*. Table 5.2 explains the features derived for the *Write Task*.

| **Feature** | **Description** |
|---|---|
| **Tap pressure** | Pressure applied when 'tapping' on a digit (value $\in ]0,1] \in R$) |
| **Tap size** | Size of the touch when 'tapping' on a digit (value $\in ]0,1] \in R$) |
| **Pressure / Size** | Ratio between pressure applied and size of the touch (unit less) |
| **Tap movement** | Movement of the touch when clicking on a digit (in pixels) |
| **Tap duration** | Time length of the touch when clicking on a digit (in milliseconds) |
| **Tap distance** | Time distance between touching two consecutive digits (in milliseconds) |
| **Back / All digits** | Ratio between BACK digits and all the clicked digits |
| **Wrong Words / All Words** | Ratio between wrongly written words and all the words of the sentence |

Table 5.2: Features derived for the *Write Task*

## 5.4 Results

In this Section we provide the statistical analysis of the data collected in our studies, both for the perceived stress data from participants' Experience Sampling Method, and from the features extracted along data collected in the two tasks.

### 5.4.1 Stress induction analysis

The first thing we analyzed is if the *Stressor Task* achieved its goal, i.e., to increase the participants' perceived stress. Moreover, we investigated if there is a significant difference between tasks performed in calm and stress state. Recalling our protocol (Section 5.2.5), participants have to fill a survey after each main step, providing their Valence, Energy and Stress state. Mean values of all the participants are reported in Table 5.3. To understand

| **Phase** | **Step** | **Valence (1÷5)** | **Energy (1÷5)** | **Stress (1÷5)** |
|---|---|---|---|---|
| **Before Relax phase** | **ESM 1** | 3,8 ± 0,9 | 3,2 ± 1,1 | 2,5 ± 1,3 |
| **Before Calm phase** | **ESM 2** | 4,2 ± 0,7 | 3,7 ± 0,8 | 1,8 ± 0,8 |
| **Before Stressor** | **ESM 3** | 3,6 ± 1,0 | 3,2 ± 1,0 | 2,5 ± 1,2 |
| **After Stressor** | **ESM 4** | 2,6 ± 1,1 | 3,1 ± 1,0 | 3,6 ± 1,1 |
| **After Stress phase** | **ESM 5** | 2,8 ± 1,2 | 2,7 ± 1,0 | 3,7 ± 1,2 |

Table 5.3: Valence, Energy and Stress perceived by participants among the protocol phases

if during our protocol the perceived stress by participants changed accordingly to the different task, we compared stress readings between ESM 3 and ESM 4 (before and after the *Stressor Task*) and between ESM 3 and ESM 5 (after the "Calm Phase" and after the "Stress Phase"), that we assume are statistically different, and between ESM 4 and ESM 5 (at the beginning and at the end of the "Stress Phase"), that we assume are not statistically different since the perceived stress should not change.

A paired, one-tailed t-test was applied to the "Stress" value of the ESM surveys. Comparison between ESM 3 and ESM 4 and between ESM 3 and ESM 5 have proved to be significantly different, while ESM 4 and ESM 5 were not, thus confirming our hypothesis. Results of the test are provided in Table 5.4, and significance level is $p = 0.05$.

| **Test** | **t(13)** | ***p*-value** |
|---|---|---|
| ESM 3 vs ESM 4 | 1,99 | 0,007 * |
| ESM 3 vs ESM 5 | -2,84 | 0,009 * |
| ESM 4 vs ESM 5 | -2,74 | 0,5 |

Table 5.4: Significance test between Stress value in survey of participants at different steps of the Protocol. (*) indicates significance at 0.05

The results show that stress values were significantly different between ESM 3 and ESM 4 and between ESM 3 and ESM 5. That means that the stress state differs from the calm state: participants were stressed. Moreover, differences between ESM 4 and ESM 5 were not statistically relevant (*p-value* $= 0.5$), which means that the perceived stress after the *Stressor Task* and the "Stress Phase" did not significantly changed. It means that our *Stressor Task* was effective, and the stress state of the participant did not change during the "Stress Phase".

### 5.4.2 Features analysis

Once proved that our *Stressor Task* (and the following "Stress Phase") effectively increased the perceived stress in participants, we analyze our features to understand how significant they are for stress state assessment.

#### Search Task

For the *Search Task*, we collected a total amount of 2937 instances, 1790 for the "scroll" interaction and 1147 or the "swipe" interaction. On average, each participant provided 225 instances. The first analysis we did was to identify features statistically different between no-stress and stress tasks. We built both a within-user model,considering only data from a single participant, and a global model, where we put together all the average values for all participants.

However, none of the features that we considered had a significant correlation in the global model (at the significance level of 0.05). Indeed, the "average swipe pressure" (*p-value*=$0,09$, $t(13) = 1,53$), "scroll distance from center" (*p-value*=$0,065$, $t(13) = 1,65$) and "scroll distance from top left" (*p-value*=$0,07$, $t(13) = 1,57$) resulted only with a weak correlation. Moreover, when considering significance applied within-user, only "scroll interaction length" was significantly different for 61% of participants. All other features were significantly different for less than 50% of participants, i.e., "pressure", "scroll time length" and "linearity" for about 30%, "average pressure" for about 40% and "scroll linearity as sum of every point" for 45% of participants.

Due to these results, instead of evaluating statistical significance between no-stress and stress state for each feature, we decided to use machine learning techniques to evaluate the precision of a stress prediction model. Even in this case, we have built a within-user model and a global model, both for 'scroll' and 'swipe' interactions. To evaluate our features, we used the Weka Software [68] for machine learning with different classifiers: Decision Tree (DT), k-Nearest Neighborhood (kNN), Bayes Network (BN), Support Vector Machine (SVM) and Neural Networks (NN). 10-Fold Cross-Validation was used to evaluate the model build for each participant, while Leave-one-out was used for the global model.

Table 5.5 and Table 5.6 report F-measure evaluation for each model, both for 'scroll' and 'swipe' interaction. Please note that some models for 'swipe' interaction for some participants are missing since during the first step of the protocol these participants did not understand how to 'swipe', and did not provide a sufficient amount of data for analysis.

On average, the F-measure of the 'scroll' within-user interaction model is 0,79 $\pm$ 0,02, while for the swipe interaction is 0,85 $\pm$ 0,03. kNN seems to be the most accurate technique for classification.

Results from our analysis show that 'scroll' and 'swipe' interaction features are accurate indicators for stress assessment and could be used to implement a real-time stress

| | **Scroll F-measure** | | | | |
|---|---|---|---|---|---|
| **User / Model** | **DT** | **kNN** | **SVM** | **NN** | **BN** |
| 1 | 0,852 | 0,874 | 0,769 | 0,879 | 0,91 |
| 2 | 0,771 | 0,825 | 0,812 | 0,869 | 0,763 |
| 3 | 0,764 | 0,694 | 0,71 | 0,673 | 0,749 |
| 4 | 0,664 | 0,779 | 0,809 | 0,725 | 0,675 |
| 5 | 0,77 | 0,819 | 0,774 | 0,774 | 0,73 |
| 6 | 0,832 | 0,894 | 0,868 | 0,87 | 0,873 |
| 7 | 0,798 | 0,769 | 0,835 | 0,767 | 0,835 |
| 8 | 0,645 | 0,605 | 0,636 | 0,633 | 0,572 |
| 9 | 0,921 | 0,941 | 0,875 | 0,915 | 0,91 |
| 10 | 0,918 | 0,932 | 0,945 | 0,966 | 0,897 |
| 11 | 0,653 | 0,672 | 0,752 | 0,672 | 0,492 |
| 12 | 0,714 | 0,701 | 0,789 | 0,77 | 0,574 |
| 13 | 0,957 | 0,9 | 0,914 | 0,936 | 0,986 |
| **Global** | **0,73** | **0,71** | **0,78** | **0,74** | **0,67** |

Table 5.5: Evaluation of ‘scroll’ interaction classification both for within-user and global model.

| | **Swipe F-measure** | | | | |
|---|---|---|---|---|---|
| **User / Model** | **DT** | **kNN** | **SVM** | **NN** | **BN** |
| 2 | 0,811 | 0,843 | 0,833 | 0,858 | 0,833 |
| 5 | 0,883 | 0,742 | 0,68 | 0,711 | 0,84 |
| 6 | 0,867 | 0,865 | 0,69 | 0,87 | 0,859 |
| 9 | 0,989 | 0,978 | 0,909 | 0,989 | 0,968 |
| 10 | 0,958 | 0,98 | 0,79 | 0,958 | 0,958 |
| 11 | 0,709 | 0,874 | 0,773 | 0,824 | 0,669 |
| 12 | 0,711 | 0,756 | 0,749 | 0,825 | 0,716 |
| 13 | 0,958 | 0,872 | 0,906 | 0,926 | 0,967 |
| **Global** | **0,92** | **0,75** | **0,81** | **0,82** | **0,77** |

Table 5.6: Evaluation of ‘swipe’ interaction classification both for within-user and global model.

assessment service that runs on a smartphone in background without affecting the user’s behavior and his/her interactions with the device.

Considering the global model for ‘swipe’ and ‘scroll’ interaction, we investigated which features are the most predictive ones to build our model. We evaluate them based on their *Information Gain* with respect to the classification problem. Table 5.7 reports the final rank of the most informative features for ‘scroll’ and ‘swipe’ interaction respectively.

From Table 5.7 we can conclude that “interaction length”, “mean distance from center”, “time” and “linearity” are predictive across both types of interactions, and are computationally cheap to derive.

| Rank | Scroll Feature | Swipe Feature |
|---|---|---|
| 1 | Interaction length | Interaction length |
| 2 | Time length | Swipe delta |
| 3 | Mean distance from center | Average touch size |
| 4 | Scroll delta speed | Average touch pressure |
| 5 | Scroll delta | Mean distance from center |
| 6 | Speed interaction | Time length |
| 7 | Mean distance from top left | Linearity as sum every point |
| 8 | Linearity as sum every point | Mean distance from top left |

Table 5.7: Rank of ‘scroll’ and ‘swipe’ features based on their *Information Gain*.

#### Write Task

Applying machine-learning techniques to each single ‘digit’ or ‘tap’ interaction during the ‘text input’ task on the smartphone is not a practical operation, since each single ‘tap’ is a too short operation to extract a sufficient amount of data for analysis. Therefore, we have decided to consider only statistical difference between no-stress and stress data for the features explained above (Table 5.2). In this case, we performed both a within-user and a global model test.

Considering within-user model, results show that “digits size” (the area clicked by the finger on the screen) is significantly different for 64% of the participants and the “ratio between pressure and size” for 55% of participants. Averaging data for each participant and building a global model, significance analysis shows that “ratio between wrong words and all the words to write of the paragraph” (*p-value* = $0,028$, $t(13) = -2,15$) and the “digits time distance” (*p-value* = $0,012$, $t(13) = 2,67$) prove to have a significant difference between no-stress and stress condition, meaning that, in stress state, we tend to write faster, but with more errors. Finally, “digits duration” feature, e.g., the time length the finger presses on the screen to tap on the digit, has only weak correlation between stress and no stress condition (*p-value* = 0,08, $t(13) = 1,51$).

The acquired results are promising and motivate us to make further analysis of stress vs. no-stress state assessment in the future. For example, since the number of misspelled words increases during the stress condition, the auto-correction feature of smartphone keyboards would correct much more frequently text messages with respect to a no-stress state. Moreover, simply measuring the time elapsed between two input (digits) may serve as a simple stress assessment method during text input.

## 5.5 Final remarks

As we can see from our experiments, human-smartphone interaction has a lot of data and information that can be used for several purposes. In particular, our stress assessment using human-smartphone interaction analysis shows how it is possible to infer stress condition of users without relying on privacy related information.

This method has several interesting implication and can be the basement for more complex systems. In particular, due to its pervasiveness, it is possible to constantly monitor people' stress level without interfering with their life. In particular, it is clear that if a user is not aware that a background service is monitor his/her stress level, we avoid that his/her behavior could be biased by this system or to cause even more stress. In this way, in the future it could be possible to build a system that can be used by doctors to constantly monitor patients among the day. Moreover, an application could provide immediate feedback and suggestions to people to reduce their stress level and decrease possible negative effects of a long exposure to stress.

# IV *Serious games* for medical purposes

# 6. *Serious games* for behavior change

We already explored one of the challenges of *ubiquitous computing*, e.g., using smartphone interaction analysis to assess stress of users, also called *affective computing*. Another interesting aspect that we have studied is related to the possibility to suggest behavior change using, as before, only a smartphone to perform all the necessary steps to accomplish this task.

The idea is to help the users to change their behavior to improve their lifestyle. As an example, sedentary life often increases the occurrence of cardiac diseases. An application which encourages the user to perform more physical activity can improve his/her health state.

An application which aims at suggesting a better behavior must follow two main steps:

- The first one is what is called activity recognition, i.e., to determine which is the activity performed by the user (walking, running, standing, etc.) and
- if necessary, the application must adopt a strategy to suggest to the user to change his/her behavior for a better one, i.e., less sedentary, with a higher amount of physical activity performed, etc.

For activity recognition, usually, the standard approach is to acquire data from sensors available on the smartphone, i.e., accelerometer, barometer, etc., and to analyze this data to give as output the activity currently performed. Usually, this is made training offline a classifier that will later provide its evaluation using the currently recorded data. Once the classifier provides its output, e.g., the recognized (predicted) activity, this output can be used to suggest a behavior change for a better lifestyle.

Clearly, if we want that people change their behavior, it is important not only to understand how the person behave, but which is the more efficient manner to suggest him/her to change behavior. Providing only a feedback, e.g., the total amount of physical activity performed among the day, is not sufficient, since people are not incentivized to change their behavior [145]. Even just saying which could be the benefits of having a different lifestyle could not be enough [103].

A possibility that seems to work to incentivize people in changing their behavior is to use *serious games* or *gamification* techniques. In this way, behavior change is not a suggestion, it is not imposed, but the expected behavior is transformed into a funny moment. Therefore, the user is not concentrated on the behavior and its motivation, but on the game, and the result is that the new behavior can become something natural, that is exactly what we aim to reach when we use *ubiquitous computing* for activity recognition and behavior change. *Serious games* can provide fun to people, both with single player or multiplayer games, that can engage people and in the meanwhile change (or suggest to change) a wrong behavior.

We start by providing a general overview of *serious games*, how they work, which are the psychological aspects which are used by the *serious games* approach and which are the best techniques that can be used to change behavior depending on the target behavior and users.

After that, we will show two different application of *serious games* into two different contexts. The first one is exactly an example of behavior change, suggesting people to make, when possible, the healthier choice, that is, the one more physically active with the respect to a more sedentary one.

The second one is the application of *serious games* to the medical field, in particular to the diagnosis of the amblyopia in preschool children. In this case, the usage of *serious games* is required to engage even children into medical exercises and tests that are usually designed for adults. In particular, the more the child is engaged in a particular game, which corresponds to a medical test, the longer this test could last and more accurate will be the answers provided by the child, since he/she will be engaged in the game, and so he/she is motivated to give the right answers to questions that will help the doctor in formulating a diagnosis.

Part of this chapter has been published in [2] [3] [61].

## 6.1 *Serious games* overview

The concept of *serious games* is not defined in an absolute way and does not have a precise definition, since it can change specification and objectives depending on the application context in which the term is used.

Trying to find some common characteristics, *serious games* can be defined as games or application, developed both for computers and mobile devices like tablets or smartphones, which have a different objective from the pure player entertainment, but to develop particular behaviors and skill hiding them under the game. Usually a *serious game* is simply a simulation that has the same appearance of a normal videogame, but it is simply the simulation a sequence of events and processes of real life. The final objective is usually to teach or educate users, but they can also be used in other contexts like marketing and advertising, providing to people a user experience much more engaging with respect to the standard printed advertising.

The proliferation of game technology and the commercial success of mobile devices endowed with sensors and communication capabilities is fostering the creation of new software systems able to ubiquitously engage and entertain users. At the same time, one interesting part of this process is represented by its potential in generating mobile *serious games* able to amuse players while providing benefits to them or even to larger communities [50] [108].

Even if there is not a demonstration that *serious games* can be useful and efficient in all the application areas, there are some evidences that, if correctly designed and developed, *serious games* are actually considered an excellent tool to support training and learning. The basis of *serious games* is the principle of *learning by doing*. This learning idea has been showed to be one of the most effective learning strategies in several contexts, moving from the concept of memorizing something with the idea of understanding something. Clearly, this principle comprehends not only the fact that the person has to do something aspect, but it is necessary to support it even with the task of learning and thinking. Otherwise, the user could simply learn to do something or a set of steps without understanding why.

In the following, we will presents some areas of applications for *serious games*, to provide an overview of how they can be used and which are some of the possible benefits, with particular interest to the *serious games* implemented for smartphones.

### 6.1.1 Exergames

The first category of games that we want to describe, the *Exergames*, uses the game paradigm to push users into increasing physical activity. The user movements are used as a form of interaction with the game, and they use sensors to recognize users' movement. In this category, the success of Nintendo's Wii platform is very well known, but also mobile platforms provide the capabilities of exploiting users' real movements as a form of interaction (e.g., rotating the iPhone to have a virtual car steering).

A first example of *serious game* applied to exercises for rehabilitation is DroidGlove [40], [41]. Actually, wrist rehabilitation is currently accomplished through very expensive and bulky machinery that cannot be moved from hospitals or through exercises that the patient is supposed to perform at home but with no way for the doctor to verify the patient's dedication and performances. Instead, DroidGlove is a serious game for Android and iPhone platforms that proposes to the users several movement tasks. The user can perform the exercises anytime and anywhere, the smartphone can remind her/him to exercise with a certain frequency, while the gyroscope is utilized to determine the accuracy of the user's movements so as to assign a score. Both the assigned movement exercises and the accomplished scores can be sent in real-time to the patient's doctor for a comprehensive supervision.

Serious games are also used to train people to do something, for example, in the military field, they are employed to train soldiers using virtual environments that reproduce real-world scenarios, i.e. "Army Battlezone". The main scope is to prepare soldiers to situations and obstacles they may encounter in the real world, making them able to take decisions faster and safer. Serious games can be used in the governmental field to simulate the population's reaction to political decisions [47] and in the educational field to increase children' learning abilities, as well as to train employees [147] [70].

### 6.1.2 Serious games in the medical field

The second category of games includes games developed to help doctors and patients. The first use the game and train to execute correctly particular procedures or to be exposed to real-life situations [48]. Patients can be pushed into rehabilitative exercises hidden by games, e.g., to perform specific upper limb movements [97] or to offer telerehabilitation to post-stroke patients so that they can perform the long series of exercises at home [43]. Furthermore, games and social communities can improve patients' recovery and motivation [121]. For instance, Re-Mission is a game that improves young patients understanding of cancer by employing game avatars representing drugs which destroy cancer cells, additionally providing a forum where patients can discuss and support each other [82].

Ciman et al. [32] designed a serious game to help the rehabilitation process from CVI (Cerebral Visual Impairment). The game can adapt the rehabilitative exercises to each child, also following the improvements of the patient, to reduce the influence of her/his disability in future life. The system also helps doctors to perform a proper assessment of a patient and to create a rehabilitation program.

Other examples of *serious games* used in the medical field regard the use of *serious games* in identifying the risk of dyslexia in children even at preschool age so as to intervene as soon as possible [62], [134]. In particular, in [62] the authors developed a set of serious games that, thanks to a cross-platform approach, can be played both on desktops and on tablets. The set of games shares an appealing underwater environment with different sea creatures used to engage the player in activities that stimulate those cognitive capabilities involved in the reading acquisition process. The games are intended to capture children attention to achieve more accurate measurements than those obtainable with non-entertaining tests.

Similar in spirit to the previous works, DYSL-X integrates dyslexia predictors in a tablet game [134]. The authors evaluate several existing games for preschoolers to derive a set of guidelines so as to design an optimal tablet game for five years children; then, these guidelines were used to develop Diesel-X, a game about a robot dog (Diesel) which has to fight against a gang of criminal cats.

Instead, Letterprins [128] is a reading game designed to improve the reading development of children with reading disorder through a series of reading tasks. The game requires the children to pronounce letters or words, while a parent or a caregiver has to indicate the correctness of the child's answers. The game allows parents to facilitate the children during the tasks and to record a message to be played at the end of the game.

These examples show that serious games are extremely useful for children since a game can change a boring rehabilitation task into an interesting activity. They can be used during assessment, rehabilitation and telerehabilitation programs. An engaging and easy-to-use interface is a key issue for this kind of games. One solution is to use the so-called tangible interfaces, which use physical artifacts for accessing and manipulating information [116]. To this end, Forlines et al. [58] investigated the differences between mouse and direct touch input, both regarding quantitative performance and subjective preference. The work shows that touch interfaces, even if they may not lead to greater performance, especially for speed and accuracy, are preferable for other considerations like fatigue, spatial memory, and simplicity. This is particularly true for children, even called digital native speakers, who find touch interaction very natural, thus avoiding the need for long training sessions to learn how to interact with touch applications.

### 6.1.3 Improving accessibility through serious games

Mobile serious games could also be used to support people with impairments. In particular, consider social communities composed of thousands (or even much more) mobile users present in each city. The combination of games with social networks, crowd-sourcing and mobile users with sensor-equipped mobile devices could create a major force able to tackle serious challenges that can be considered too complex for single users and/or difficult to automate. As an example, imagine how a serious game could help visually impaired users. Bringing the Google Image Labeler serious game [77] into the real world, where users were engaged in labeling random images to improve the quality of Google's image search results, we could design a social community where mobile users are asked to play a game involving the labeling of the surrounding environment (e.g., crossroads, architectural barriers, parks, stores). Their idea is to add digital tags to real objects, creating a social community similar to Panoramio [76], where users upload their geolocated picture to add new information to Google Maps. The "serious" advantage would be that of having a participatory, augmented reality environment where a visually impaired person could perceive the surrounding real world through her/his mobile device able to retrieve the aforementioned digital labels and transform their format not suitable for visually impaired users (text, image, video, etc.) into audio, thus improving her/his autonomy.

Researchers already presented some algorithms supporting mobility-impaired pedestrians, providing means to generate specific urban routes that consider the accessibility of roads and curbs [15], [81]. However, one of the most complex challenges is related to the accessibility assessment of roads and curbs [124], and without an initial assessment, there would be no data to feed to the aforementioned algorithms. To this end, some approaches focused on the possibility to autonomously and anonymously detect favorite routes chosen by people with certain impairment (e.g., being in a wheelchair) and consider them as preferable when someone else in the same condition searches for a route in the neighborhood [22], [106]. However, we have to mention that games have been considered as well to involve citizens in playing a mobile serious game whose goal is achieved by labeling as many accessible or no accessible roads, curbs, pedestrian crossing, traffic lights, etc. [107], [122]. Aiming at improving as much as possible the inclusion of people with impairments in our society, the best approach is probably a mix of the two approaches: automatic detection and serious game. Moreover, even not impaired people can benefit from this particular system since, for example, accessibility can help mothers with strollers, or help to find the path with the highest amount of shadow.

## 6.2 Persuasive Technologies

*Persuasive Technologies* is currently used to indicate a multidisciplinary research field very active in the last few years which focuses on the design and development of interactive technologies that aim at changing habits or behaviors through persuasion and social influence but without coercion or deception.

These technologies are used in several fields, like marketing [80], education [57] or safety [29]. Recently, research interests moved to behavior change, especially for health and wellness. In this field, the figure of the *health coaching* has gained lot of importance, that has the role of assisting people in behavior change with specific techniques and encouragements. This role has been imported even in the technological field, with the creation of the *digital health coach*: the scope is to use computers as a persuasive technology to increase the wellness of the patient.

The most effective interventions are modeled on the *digital health coach* that requires to the users to set their goals; in this way it is possible to educate them based on the consequences of their behaviors and after to encourage them to record their progresses and check them with their goals. These kinds of technologies can clearly become very important since they can strongly reduce health costs, e.g. better lifestyle means decreasing the possibility of diseases and so of medical treatments that have a cost, meaning that health insurances and governments are strongly interested in invest in this research field.

Actually, most of the researches are concentrated on interactive technologies with computational capabilities, included computer, web services, videogames and mobile devices. For example, in the last years we have seen the development of a huge number of videogames that requires to the players to perform physical activities, i.e., use the bicycle or a virtual tennis racket. Some of these videogames were created only as a different form of entertainment, using physical activity as a different way of interaction. Others aim at helping people to move from a sedentary life to a more active one. Some commercial examples of these games are "Dance Revolution" for Xbox (see Figure 6.1c) or "Wii Fit" (see Figure 6.1a) and "Wii Sports" (see Figure 6.1b) for Nintendo Wii. This shows how it is possible to create funny games that introduce some physical exercises.

(a) Usage of Nintendo Wii Fit

(b) Usage of Nintendo Wii Sports

(c) Usage of Xbox Dance Revolution

Figure 6.1: Wii and Xbox games for an active life.

Persuasive technologies can reach users with other products different from video games, using, for example, web applications, electronic devices or even smartphones. Indeed, new *mobile computing devices* have growing computational capabilities, interactive displays and are clearly always available during everyday life of users, thanks to their portability. Moreover, these devices provide even network connectivity. All these characteristics allow persuasive technologies to gain access to areas that "human persuasors" cannot reach. Finally, thanks to the sensors available on these devices and algorithms of context-awareness, it is possible to deduce information about the context where the user is living and the activity he/she is performing. This makes possible to provide messages and persuasive feedback to the users at the right timing and places.

Several companies like Nike, Runtastic or Fitbit have exploited potentialities of smartphones from a commercial point of view, creating both applications and wearables, i.e. *fitness trackers* or *smartwatch* that can record the physical activity of users and let them set their daily goals. This kind of approach was such a success that other companies like Google and Apple, the most important companies that lead the smartphone market, have developed their own native applications, e.g., Google Fit[1] (see Figure 6.2a) and HealthKit[2] (see Figure 6.2b), that are able to directly connect to these external devices (or use data coming from sensors of the smartpone) and collect information all together into one application. Moreover, these native applications provide APIs to let developers use this information to create their own application that requires this kind of data, i.e., a fitness app.

### 6.2.1 The Fogg Behavior Model (FBM)

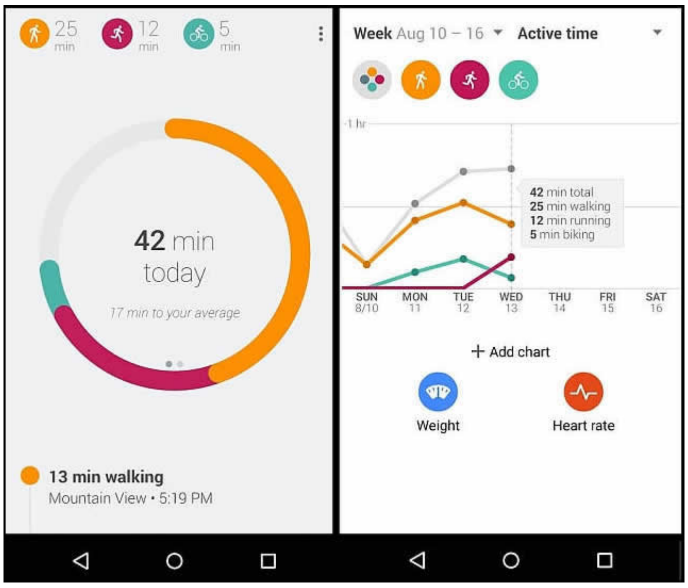
In the *persuasive technologies* field, studies by Prof. B.J. Fogg are the most important and recognized. In particular, he developed a behavior change model for human behavior, that should drive both design and development of applications that aims at changing people behavior.

Following the Fogg Behavior Model (FBM), three different elements have to converge at the same time to obtain a particular behavior from an individual, as shown in Figure 6.3:

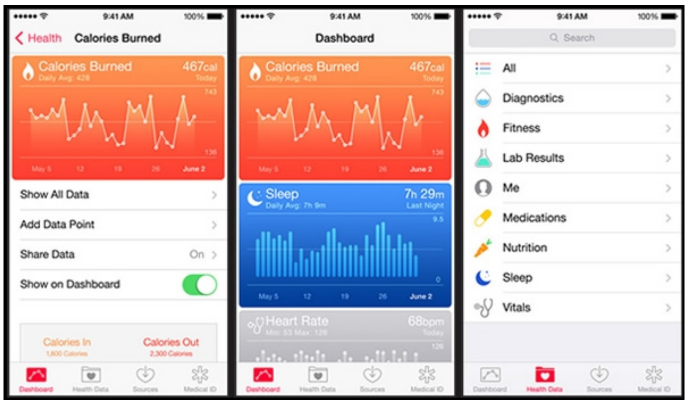
- **Motivation:** the person has to be sufficiently motivated to change his/her behavior;
- **Ability:** the person has to be able to perform that particular behavior (meaning that this behavior has to be sufficiently easy to be performed). This ability can be time, attention, mental capability or any other resource necessary to complete the behavior. Without these resources, the person would not be able to reach the target behavior;
- **Trigger:** to perform a particular behavior, the person has to be motivated to do that.

To be sure that the person performs the target behavior, he/she needs to have a minimum level of *Ability* and *Motivation*. That level is called *activation threshold*, and only if this minimum level is passed the *Trigger* will help the person to perform the target behavior. Under this *activation threshold*, the *Trigger* will be ineffective and useless (see Figure 6.4).

[1]Google Fit: https://fit.google.com/u/0/
[2]HealthKit: http://www.apple.com/ios/health/



(a) Google Fit



(b) Apple HealthKit

Figure 6.2: Google and Apple native applications for healthcare.

Each of these elements has its own subcomponents: specifically, the FBM defines three different “core motivators”, six different “simplicity factors” and three different triggers. The concept of simplicity is strongly linked with the one of ability: indeed, the *persuasive design* is strongly based on the power of simplicity, i.e., the “One-click shopping” by Amazon, but it is always important to remember that if something is easy for some people, could not be so easy for others.

The three “core motivators” are:

- *Pleasure/Pain*: is the most immediate one, since users react to what is happening in that moment. Their power is related to the fact that pleasure and pain are primitive reactions of human behavior;
- *Hope/Fear*: this motivator is characterized by the anticipation of a result (hope anticipates something positive, fear something negative);

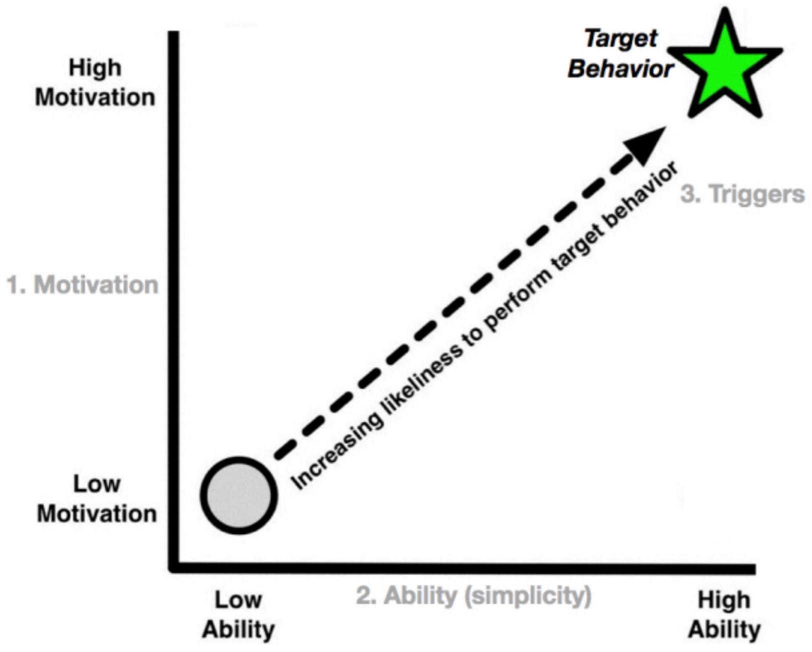


Figure 6.3: The Fogg Behavior Model [53].

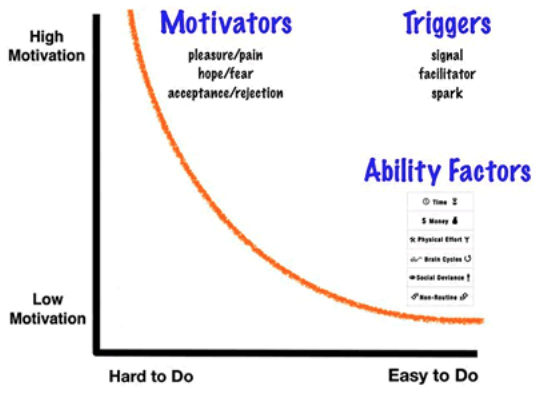


Figure 6.4: The Fogg Behavior Model: the *activation threshold* [56].

- *Social Acceptance/Rejection*: this motivator has consequences on everything, from clothes we wear to the language we speak. Indeed, people are motivated doing something that let them gain social status and acceptance and avoid any negative consequences, like being excluded.

There is not a rank of the "core motivators" and every time is necessary to choose the best one depending on the context.

The six simplicity elements are:

- *Time*: if the target behavior requires an amount of time that he/she does not have, that behavior is not easy;
- *Money*: if the person does not holds lot of money, target behaviors that require

money cannot be considered easy;

- *Physical effort*: target behaviors that require physical effort from the person could be difficult;
- *Brain cycles*: if the target behavior requires long brain efforts or to think in an unusual way, this could be considered hard by the users;
- *Social deviance*: if the target behavior requires behaving against the rules of the society, that behavior is not easy;
- *Non-routine*: if the target behavior is a routine, that behavior is easy, otherwise not.

Finally, the three possible *Triggers* are:

- *Spark*: when the person lacks the motivation to perform the target behavior, it is necessary to fire the *Trigger* with a motivational element;
- *Facilitator*: this kind of *Trigger* is useful with high motivation but low ability: the goal is to trigger the easiest behavior for them. So, it is necessary to remark to the person that the target behavior is easy to perform and do not need any additional resources;
- *Signal*: it is fundamentally a *reminder*, since it is intended for people with high ability and motivation.

### The Fogg Behavior Grid (FBG)

B.J. Fogg described 15 different ways how a behavior can change. Each of these one uses psychological strategies and different persuasive techniques, since to transform a particular behavior from happening only one time to a routine requires different interventions. However, each of these strategies are based on the FBM [55] [54].

Fogg identifies five different behaviors, each one associated with a different color:

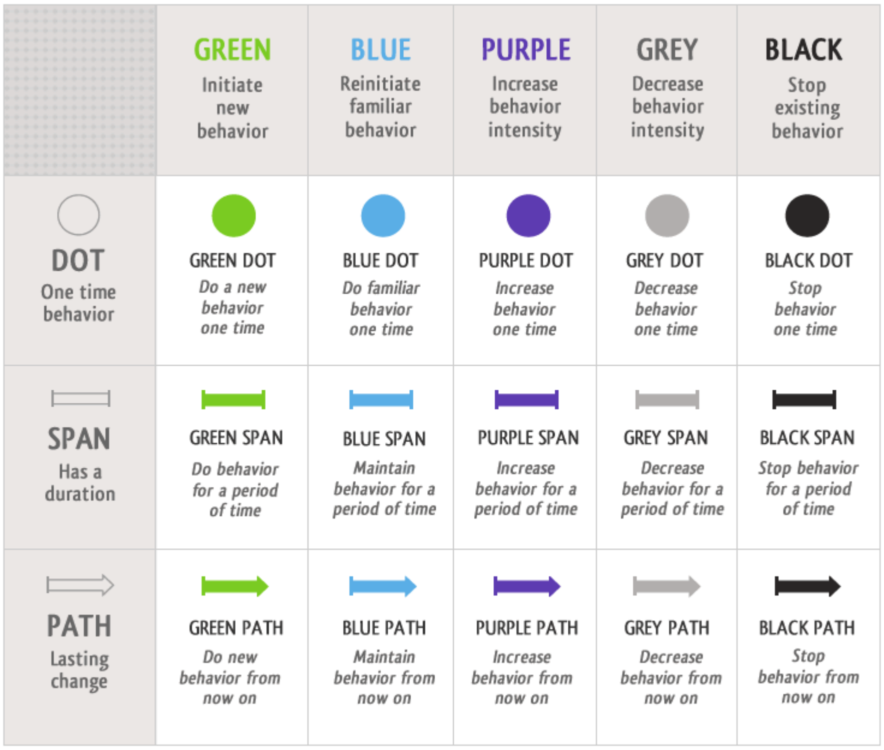
- **Green**: maintain a particular behavior;
- **Blue**: maintain a familiar behavior;
- **Violet**: increase behavior intensity;
- **Grey**: decrease behavior intensity;
- **Black**: stop a behavior.

Moreover, he defines three different time intervals when the new behavior has to be kept, associating this time with a particular geometric shape:

- **Dot**: perform the new behavior only one time;
- **Span**: maintain the new behavior for a period of time;
- **Path**: maintain the new behavior from now ongoing.

The final table resulting from this description is depicted in Figure 6.5, and is called *Behavior Grid*. Each behavior requires a different balancing of each FBM component depending on its position on the grid.

The combination of the FBM and the *Behavior Grid* has led to the *Behavior Wizard*



| | GREEN<br>Initiate new behavior | BLUE<br>Reinitiate familiar behavior | PURPLE<br>Increase behavior intensity | GREY<br>Decrease behavior intensity | BLACK<br>Stop existing behavior |
|---|---|---|---|---|---|
| ○<br>DOT<br>One time behavior | ●<br>GREEN DOT<br>*Do a new behavior one time* | ●<br>BLUE DOT<br>*Do familiar behavior one time* | ●<br>PURPLE DOT<br>*Increase behavior one time* | ●<br>GREY DOT<br>*Decrease behavior one time* | ●<br>BLACK DOT<br>*Stop behavior one time* |
| ▭<br>SPAN<br>Has a duration | ▬<br>GREEN SPAN<br>*Do behavior for a period of time* | ▬<br>BLUE SPAN<br>*Maintain behavior for a period of time* | ▬<br>PURPLE SPAN<br>*Increase behavior for a period of time* | ▬<br>GREY SPAN<br>*Decrease behavior for a period of time* | ▬<br>BLACK SPAN<br>*Stop behavior for a period of time* |
| ⇨<br>PATH<br>Lasting change | ➡<br>GREEN PATH<br>*Do new behavior from now on* | ➡<br>BLUE PATH<br>*Maintain behavior from now on* | ➡<br>PURPLE PATH<br>*Increase behavior from now on* | ➡<br>GREY PATH<br>*Decrease behavior from now on* | ➡<br>BLACK PATH<br>*Stop behavior from now on* |

Figure 6.5: *Fogg Behavior Grid* [54].

formulation. With this word we refer to a systematic method to identify specific kind of target behaviors and the best solution to obtain the behavior change.

The *Behavior Wizard* [54] is divided into three different steps:

1. identify the target behavior using the *Behavior Grid*;
2. identify how the target behavior is triggered: indeed, this behavior can belong to the *"Cycle" behavior*, that are behaviors that have to be done at a specific time or following a particular plan, i.e., daily, weekly, etc., or *"Cue" behavior*, behaviors that happen as answer of other particular events, not predictable or scheduled;
3. identify theories, models and solutions for the target behavior we identified: at this step, the *Behavior Wizard* generates some relevant information for those who aim at creating a persuasive experience. These information are called *Resource Guide*.

### 6.2.2 Behavior Change Support System (BCSS)

Fogg results in the field of persuasive technologies have been extended with the concept of *Behavior Change Support System (BCSS)*, which was introduced in [104], which is defined as:

" *A behavior change support system is an information system designed to form, alter or reinforce attitudes, behaviors or an act of complying without using deception, coercion or inducements.*"

| | C-Change | B-Change | A-Change |
|---|---|---|---|
| **F-outcome** | Forming a compliance | Forming a behavior | Forming an attitude |
| **A-outcome** | Altering compliance | Altering a behavior | Altering an attitude |
| **R-outcome** | Reinforcing compliance | Reinforcing a behavior | Reinforcing an attitude |

Table 6.1: Outcome/Change Design Matrix

The types and results of changes have been divided into different categories and are presented in Table 6.1.

To build a BCSS both psychological and design skills of interactive systems are required. Currently, the state of the art for the design and the development of the BCSS is the *Persuasive System Design Model.*

#### Persuasive System Design Model

The *Persuasive System Design (PSD)* model, presented by Oinas-Kukkonen and Hariumaa in [105], was build based on empirical data and other important psychological studies. This framework aims at discussing the design and evaluation process of persuasive systems and to describe which kind of functionality and contents should be present in the final product. This model has been analyzed in several contexts [84] [90] and suggests that not all the design principles should be used in each application, but should be selected every time depending on the considered problem and the background theories.

The model describes what are called *key issues*, a *process model* and *design principles* to develop and evaluate these systems. Several technological possibilities to design persuasive systems are presented and reported, and strategies that motivate people.

Before implementing or evaluate the entire system, the model requires the comprehension of the fundamental questions behind the persuasive systems and that are fundamental for the following steps of analysis and design. The second phase consists of analyzing the context of the persuasion, identifying the final objective, the event and the strategies for the usage of a persuasive system. Finally, the real qualities of the new system are designed, or the characteristics of already existing systems are evaluated.

### 6.2.3 Gamification

The term *gamification* has been defined in different ways. In [42] it is defined as follows:

“ *[..] the usage of game design elements in non-game context.* ”

Similarly, Werbach and Hunter in [143] provide this definition:

“ *[..] the use of game elements and game design techniques in non-game contexts.* ”

What these definitions have in common is the fact that *gamification* focuses on “non-game contexts”, excluding in this way *serious games.*

However, another important definition is provided in [42]

“ *[..] the use of game thinking and game mechanics to engage users in solving problems and increase users’ self contributions.* ”

This definition is more wider and includes *serious games*. The difference between an application with *gamification* and *serious games* is still not well defined and is currently under discussion.

*Gamification* is characterized by taking place in real life, considering the game a second aspect of the task to complete. The user is persuaded using typical techniques of *persuasive technologies*, e.g., *tunneling*, *tailoring*, *suggestions*, etc., where the application or the information system acts as a interlocutor. These techniques influence the behavior change during the task, making it easier.

Usually, a system with *gamification* is not *immersive*: the game is essentially a casual game, requiring only partial attention from the user, i.e., the user gains points and badges answering particular questions on a forum.

*Gamification* techniques try to exploit natural people desire to socialize, learn or master something, to compete, to succeed and show his/her own status. The *gamification* strategies use rewards for the players that complete the required tasks, or competitions to attract other players. Different types of rewards include points, achievements, badges, levels, the progress showed using some progress bars or providing to the player a kind of virtual currency. To give a reward for the completion of particular tasks and make it visible to other players or provide ranks is a strong way to encourage people to compete with each other. Another approach to *gamification* expects to provide some mechanisms of onboarding with a tutorial, to let the player make some important choices, to increase the number of challenges and to add a story [148].

### Gamification and Persuasive Technology

How to add game elements is one of the most challenging and interesting strategies to transform the Information and Communication Technology (ICT) into a system that persuades and motivate users to behave in a particular way. To design games and use *gamification* elements opens the doors to the creation of final persuasive products [12]. This conclusion is presented in [142]. In this article, *gamification* is defined as the process necessary to transform activities more “game-like”. In particular, they pose the focus on the creation of entertainment experiences, avoiding superficial approaches that can easily become manipulative. This approach connects *gamification* to persuasive design and to *serious games*. The author says that “game-like” experiences can promote both motivation (making activities more engaging) and ability (promoting learning, accomplishment and sense of confidence). This means that *gamification* can work as a specialized tool that can increase the behavior change interventions that even Fogg and others described (see Section 6.2.1). This last sentence has been confirmed by different theories, especially from the "Flow" theory by Csikszentmihalyi [37]. He defines the "flow" as:

" *The satisfying, exhilarating feeling of creative accomplishment and heightened functioning.* "

This happens when a person is completing a task and is completely absorbed by it. Following authors idea, this lacks in everyday life but is strongly present in videogames. If the task is too easy, the user will be bored and not engaged by it, otherwise, if the task is too difficult, he/she will feel anxiety and will be demotivated.

Each one that designs a game has to take into consideration skills and the level of ability and challenge of the user, that has to be slowly increased while the player gains experience. Moreover, Csikszentmihalyi says that a game is naturally persuasive, due to its voluntary and goal-directed nature. Indeed, games push the player to objectives, not in a coercive way, like persuasive technologies. This link is underlined more strongly in [19], where they introduce the terms of "persuasive games", since games have a persuasive power that is stronger than any other form of persuasive techniques. They are able not only to support social and cultural positions already existing, but can even change them, taking even to a potentially and significantly long-term social change.

However, it is important to remember that to define an application as a *serious game* (as a real game) or simply as an application with *gamification* elements is not easy, since the difference between the two classifications is still under discussion.

# 7. *ClimbTheWorld* to promote physical activity

*Serious games* play a fundamental role to promote behavior change since, as explained for example in [103], to simply provide feedback to the user without any incentive to change unhealthy behavior for a healthy one is not sufficient.

For this reason, we developed a *serious game* called *ClimbTheWorld*. The idea of this *serious game* and the necessity of behavior change comes from the fact that, according to the World Health Organization, at least 3.2 million people die every year due to heart diseases, diabetes, cancer obesity (both in adults and children) [144]. Insufficient physical activity is one of the leading causes of these diseases: in the last few decades, the technological progress has completely changed people's lifestyle, making everyday activities much easier. Unfortunately, this progress also moved people to a sedentary life, thus increasing the occurrence of these diseases and the medical costs for their treatment.

The availability of many infrastructures that reduce movements necessary to reach a destination is not balanced by the presence of tools which help people to modify unhealthy behaviors and to avoid negative consequences of physical inactivity. Since everyday choices between taking a bus and a short distance walk, or between stairs and elevator could greatly affect the total amount of activity made, the idea behind our game is to incentivize people in taking stairs instead of elevators or escalators, that means made more physical activity among the day. The idea underlying the game is simple: the user has to climb real world buildings, e.g., the Empire State Building or the Eiffel Tower, making stairs during his/her everyday life. Once started, the game records and analyzes data from the accelerometer and count the number of stairsteps made by the user. The game provides both single-player and multiplayer modes, to satisfy player needs and preferences (different users could prefer single mode instead of multiplayer mode and vice-versa), and uses Facebook to create competitiveness between friends with multiplayer challenges and to provide the possibility to share personal results and achievements.

An experiment, which was part of "The Fun Theory" [136] project provided by Volkswagen, shows that the use of *serious games* is the right way to incentivize people to use stairs. In particular, the installation named “Piano stairs” [137] tries to transform stairs into a piano, where each step is a key of the piano, that sounds every time a person climbs into. The authors experimented this idea by properly equipping the stairs of a subway station. As a result, more than 66% of the people preferred the “Piano stairs” to the escalators.

Despite its success, this solution has a big problem, i.e., its portability. In fact, this installation cannot be “moved” easily to another place, but requires the installation of many devices, sensors, actuators and speakers, together with related masonry and painting. Moreover, it is not always applicable, e.g., in a historical building, or the installation can be very expensive. For this reason, even if the principle is correct and evidence show the goodness of this approach, its widespread application is extremely difficult.

Starting from these good results regarding people engagement, our idea is to improve this system, to make it easier to use everywhere.

To be efficient, the game must fulfill two requirements: it must be able to count stairsteps without immediately exhausting the smartphone battery, and it must be able to persuade the users to change their behavior. *ClimbTheWorld* implements a new method for stairsteps recognition and counting. The game requires a fine-grained classification, that is a real-time counting of the number of stairsteps along with real-time feedback. Moreover, it does not impose any constraints on the smartphone position, and it does not require people to buy expensive tools (e.g., the Nike+ FuelBand bracelet[1]) but it uses only the smartphone sensors.

To avoid exhausting the smartphone battery, the whole design and implementation process has taken into account the energy consumption issue, since several studies have shown that energy consumption is a critical aspect for mobile applications [101, 111] and can influence performances on some user experience metrics [18]. In particular, battery lifetime is one of the most important aspects considered by users when dealing with applications on mobile devices. For this reason, our approach gathers strictly necessary information from the lowest number of sensors, and avoiding the use of a high number of features that could lead the final classification to become too expensive.

[1] `http://www.nike.com/us/en_us/c/nikeplus-fuelband`

## 7.1 Related Works

Before the usage of smartphone sensors to perform activity recognition (and, in particular, step counting), an analysis of performances reached by pedometer during stair climbing was made by Ayabe et al. [10]. The purpose of their experiments was to understand how well pedometers perform during stair climbing and descending. They evaluate three different commercial pedometers and different stepping rate (from 40 to 120 steps$\cdot$min$^{-1}$). Although they do not distinguish between steps and stairsteps, results show that pedometer can assess stairstep counting within an error rate of $\pm5\%$, being a great tool to count the number of stairsteps (and steps) made by each user.

Thanks to the increasing performances of smartphones and their ability to acquire a lot of data from the surrounded environment, mobile applications that use data from sensors for activity recognition and to promote better lifestyle received considerable interest from the research community.

Anjum and Ilyas [6] developed an application for *online* activity recognition like walking, running, climbing stairs, descending stairs, cycling, driving and remaining inactive. They provided an analysis of 5 seconds length windows using several classification algorithms like kNN, Naive Bayes, Decision Trees and Support Vector Machines. They collected data from the accelerometer, the gyroscope, and the GPS. The precision of the results ranged between 79% (using Support Vector Machine) and 94% (Decision Tree). Their analysis also showed that data retrieved from the gyroscope do not provide any useful information, therefore they removed all the features coming from this sensor. Moreover, they stated that the recognition of stairs climbing and descending is a really difficult task, achieving a much lower precision, and often stair climbing is confused with walking (precision of 84.6% for going upstairs, 90,5% for going downstairs).

Shoaib et al. [126] analyzed activity recognition using the accelerometer, gyroscope, and magnetometer, alone or combined at a frequency of 50Hz. They considered four different positions of the smartphone (arm, belt, pocket and wrist). They showed that the magnetometer, both alone or in combination with other sensors, performs poorly since it causes overfitting for the classifier. Moreover, they found that the accelerometer performs better than the gyroscope in recognizing the six different activities, but, in contrast with the results provided by Anjum and Ilvas, this work showed that the combination of accelerometer and gyroscope data performs better than the two taken individually.

Brajdic and Harle [21] considered walk detection and step counting algorithms with data acquired from a smartphone; the user can choose to carry the smartphone in six different positions. This work discussed some issues in common with our approach, but in our case, we count stairsteps (so we need to discriminate between steps and stairsteps, which is a more difficult task). The authors evaluate several algorithms using a dataset of

| **Authors** | **Use of cell phone** | **Real-time** | **Sensors** | **Orientation** | **Window** | **Activity** | **Stairstep counting** |
|---|---|---|---|---|---|---|---|
| [10] | No | Yes | Acc., Gyr. | 1 | - | stairs | Yes |
| [6] | Yes | Yes | Acc., Gyr., GPS | 4 | 5 *s* | stand, walk, run, cycle, stairs | No |
| [126] | Yes | Yes | Acc. | 1 | 4 *s* | stand, walk, cycle, drive, run | No |
| [21] | Yes | No | Acc. | 6 | (*) | Walking | No |
| [146] | No | No | Acc., Gyr. | 1 | 2 *s* | Stand, walk, run, stairs | No |
| [17] | Yes | - | Acc., GPS, WiFi | Free | - | Walk | No |
| [35] | No | Yes | Acc., Bar. | Fixed | - | Cardio, walk, resistance, flexibility | No |
| [89, 96] | Yes | Yes | Acc., GPS | Free | - | Walk, drive, stand, run | No |
| [45] | Yes | No | Acc., GPS | 4 | - | Walk, run, stand | No |
| [79] | No | Yes | Bracelet | Free | - | Walk, run, dance, sleep | No |
| [36] | Yes | Yes | Acc. | Free | - | Steps | No |
| [7] | Yes | Yes | Acc., Alt., GPS | Free | - | Walk, run | No |
| [132] | Yes | Yes | Acc., GPS | Free | - | Walk, run, riding | No |
| [99] | Yes | Yes | Bracelet | Free | - | Walk, run, sports | No |
| [51] | Yes | Tes | Bracelet, GPS | Free | - | Walk, run, sleep | No |
| Our solution | Yes | Yes | Acc., Rot. | Free (**) | (***) | Stairstep | Yes |

(*) [21] allows different window sizes
(**) The only constraint is not to use the trouser pocket
(***) Our solution uses a data dependent window size

Table 7.1: Resume of the different approaches presented in Section 7.1 compared with our solution

130 walks traces from 27 different users, getting that most of the algorithms can detect walking within a trace that contain only walking and idle periods, with a best median error of 1.3 using thresholding technique, but no one of the algorithms is 100% reliable.

Wu et al. [146] analyze activity recognition using an iPod Touch, acquiring data from the accelerometer and the gyroscope. They recognize activity like sitting, walking, jogging, going upstairs and downstairs. They extract both time, frequency and Fast Fourier Transform features. They make data analysis offline, using a 2-second window size with data acquisition at 30Hz. They achieved the best results using kNN. When sitting, walking and jogging, the accuracy is very high (90,1%-94,1%), while up and down stair walking is classified with a lower precision (52.3%-79.4%).

Ubifit Garden [35], one of the first mobile persuasion system for improving physical wellbeing, uses the wallpaper of mobile phones to provide dynamically feedback about the different types of physical exercises performed by the user.

BeWell [89] is a smartphone application that aims at monitoring user behavior and wants to promote wellbeing. The application continuously tracks activities that impact several aspects of daily life of people like social interactions or physical activity using smartphone's sensor, and provides several feedback to promote better health. The application is based on a *serious game* and displays the "state" of the user as an aquatic ecosystem that changes the behavior of different animals depending on the changes in user wellbeing.

The entire system is based on a cloud infrastructure used to analyze the data acquired with the smartphone and to build a model of the user based on his/her behavior. The second version of BeWell, called BeWell+, introduces several improvements [96]. Firstly, daily activity goals are adjusted depending on the person. This means that, for example, expected activity for an older man is different from a young one, or from a student and an employee. Secondly, energy consumption is considered, reducing resources used by the application for monitoring if the behavior of a user tends closely to healthy norms. In this way, fewer system resources are required.

HEALTHYLIFE [45] is a smartphone application that automatically recognizes users activities. It recognizes activities like walking, running, driving and staying still. They use the accelerometer data and the GPS signal. Moreover, they apply *Ambiguity reasoning* to increase classification results and to disambiguate data. The idea is to apply a set of weak constraints that attaches to any possible answer of the classifier a violation cost that depends on the number and type of violated constraints. The right classification answer will be the one with violation cost equals to 0. The final precision of the system ranges from 100% (staying-still) to 73% (walking).

MOVE2PLAY [17] is a smartphone application which encourages a healthier lifestyle and motivates users to participate in daily physical activities. It combines daily targets, for short term motivation, with longer term targets, mainly using social elements, i.e., targets sharing, competitions, etc., to avoid dropouts from the game and to constantly try to keep users engaged.

We also analyzed some commercial applications and wearable devices used to monitor physical activity. Jawbone Up [79] is a wearable system to monitor physical activity, sleep and resting heart rate. It uses a bracelet to record physical activity and a mobile application to monitor it. It does not distinguish between stairsteps and simple steps, it simply measures the amount of physical activity.

Accupedo [36] is a smartphone application which implements a pedometer using an intelligent 3D motion recognition algorithm. The user is not forced to use the smartphone in a particular position, so this work seems to be similar to our approach, but Accupedo is not able to distinguish between stairsteps and simple steps, but it counts both as steps.

Apple Health [7] is another smartphone application for monitoring physical activity. The application tracks simple steps and the distance of walk or run. Apple Health simulates stairstep counting: it recognizes a flight of stairs, using an altimeter, as approximately 10 feet (3 meters) of elevation gain, and it counts it as approximately 16 stairsteps. Therefore, the application does not recognize stairstep but flight of stairs. Moreover, it uses the altimeter sensor that requires more battery energy.

Endomondo [132] is a smartphone application which can be used as a fitness tracker.

It can track a wide set of fitness activity, e.g., running, walking, riding, but it does not perform a fine-grain recognition of this activity, i.e., is not able to count the number of steps, and therefore, the number of stairsteps. Moreover, it is not able to recognize a stairstep versus a simple step. Moreover, it uses external devices like bracelets or a cardio frequency meter.

Misfit and Fitbit produce bracelet for activity and sleep tracking. Misfit Flash [99] is a sporty fitness and sleep tracker. The bracelet can be worn anywhere, is connected to a smartphone application and can to recognize different activities like running, walking, cycling, playing tennis, soccer or basketball. Unlike our application, it does not recognize and count stairsteps, not even flight of stairs. Bracelets produced by Fitbit [51] works in a very similar manner, but they are also able to recognize flight of stairs. Even in this case, the application does not perform a fine-grained activity recognition and does not count stairsteps, it only recognizes flight of stairs and count the number of climbed floors.

To the best of our knowledge, commercial applications are not able to count in real-time the number of climbed stairsteps and address a different task, monitoring physical activity and sleep. These works do not count stairsteps but simple steps and, in fact, they either are not able to distinguish a step from a stairstep [79, 36, 132] or they are only able to calculate the number of climbed floors using altimeter [7, 51] or to approximate the number of stairsteps using the altimeter and the average height of a stairstep [7]. Moreover, these applications often used an external (intrusive) device like bracelet [79, 132, 99, 51], while we aim at creating a more pervasive solution which does not require anything more than a smartphone which, nowadays, is present in almost all users' pockets.

Table 7.1 provides a comparison of our approach and the others presented in this section. The analysis of the related works suggests several open issues which need a further study. First of all, since we are pursuing a smartphone based real-time recognition, energy consumption is a fundamental aspect, never considered before. For example, the usage of multiple sensors at the same time (accelerometer, gyroscope, magnetometer and GPS) will rapidly drain the battery, causing a lot of stress to the user. For this reason, we will rely on the accelerometer and the rotation sensor only, avoiding the usage of much more expensive sensors. Moreover, differently from other solutions that recognize an activity over a large window of time, we aim at recognizing each single stairstep, and distinguish it from a simple step. As shown in Figure 7.3b and Figure 7.3c, this is not an easy task, since the signal obtained by a stairstep and a step has a similar behavior. Finally, most of the previously proposed approaches impose a set of fixed positions for the smartphone[2] and the training of a different classifier for each supported position. As we will see, we do not

[2]We must note here that even if most of the commercial apps do not impose a fixed position to wear the smartphone, they often require an external device, which is even worst in term of cost and intrusiveness of the system.

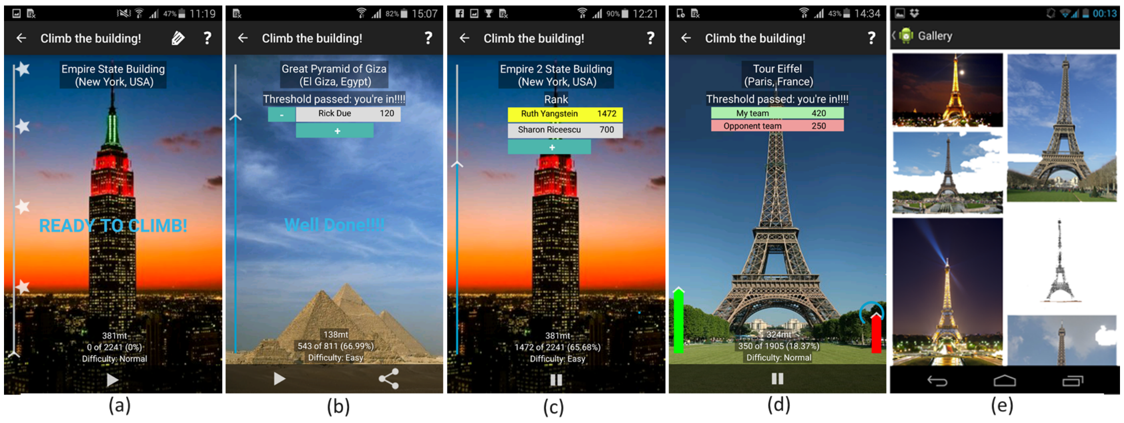


Figure 7.1: Five screenshots of *ClimbTheWorld*

impose any position of the smartphone, we only ask to avoid the trousers pocket, and no (expensive) external devices are required to acquire data.

## 7.2 Game Design

The aim of *ClimbTheWorld* is to persuade people to choose stairs instead of escalators, thus increasing their physical activity, using *gamification*, i.e., providing a way to have fun while climbing stairs. The whole design process of the game followed the Fogg Behavior Model (FBM) to improve game's persuasiveness, as already discussed in Section 6.2.1. FBM shows that three elements must converge at the same moment for a behavior to occur: *Motivation*, *Ability* and *Triggers*. Since the game is not intended for impaired people, *Ability* is not a problem since our target audience can climb the stairs. But it is also true that, even if climbing the stairs can be considered an easy activity, it can become tiring, or even frustrating, if the goal is too far away. For this reason, we add *sub-goals* to *ClimbTheWorld*, denoted by the stars in Figure 7.1(a), to encourage users to never give up.

In the same way, to avoid discouraging the users, the game proposes different difficulty levels: easier levels correspond to a lower number of stairsteps necessary to reach the top of the building. Each stairstep in real life corresponds to one (or more) stairstep in the game: higher difficulty means less stairsteps in the game. Once the user reaches the top of the building, a slideshow of pictures is displayed, showing the view from the top of the building. Different difficulty levels also bring different quality and number of provided photos. Figure 7.1(e) shows a screenshot of the gallery of the Eiffel Tower.

To improve *Motivation* and *Ability*, we design four different game modes, which can involve or not the user's friends[3]. The first game mode requires the user to climb a building alone (see Figure 7.1(a)). Since some building may have a big number of stairsteps, the

[3]We required the user to connect to the social network Facebook and to give the application the right to explorer his/her network of friends.

user can invite friends to help him/her to climb the building. This second mode is called *Social Climb*, since all the users contribute to climb the same building. Figure 7.1(b) shows a screenshot where the user has invited one of his/her Facebook friends to help him/her to reach the top of the Pyramid of Giza. This game mode improves *Ability*, since it lowers the required number of stairsteps.

*Motivation* is strongly affected by two other game modes, *Social Challenge* and *Team vs. Team*. The first one implements a challenge between two (or more) players. Differently from *Social Climb* the players do not collaborate but compete. The winner is the first user to reach the top of the building. Figure 7.1(c) shows a screenshot during a challenge between three players. The game reports the number of climbed stairsteps for each user. The player with the yellow background is the one who started the challenge and can decide to add or remove other players.

The *Team vs. Team* mode implements a challenge between teams of an equal number of players. In Figure 7.1(d) the two bars on the sides show the progress of the two teams.

Since smartphones are not always connected to the Internet, the game continues working even in absence of the network connection. If the game is in one of the multiplayer modes and some players are not connected, the game pauses at the end of the climbing and waits for data from all the users before declaring the winner.

Another problem related to multiplayer modes is the management of *freeloaders*, i.e., players who join a team or a social climb but do not contribute to the climb with stairsteps. To avoid this type of players, *ClimbTheWorld* imposes a threshold (see Figure 7.1(b,d)): players who do not contribute with a minimal set of stairsteps are not rewarded even in case of victory.

The last elements of the Fogg Behaviour Model, the *Triggers*, are implemented by a push notification that remembers the player to use stairs when he/she lowers the number of stairsteps climbed during the day with respect to the day before. According to the FBM, this type of triggers are called *signal*.

To increase the engagement of the user, the game provides a set of bonuses, depending on the performances of the user. For example, if the user improves his/her performances with respect to the day before, he/she gets a 30% increase on the total number of stairsteps made. These bonuses are used to encourage constantly and help the user.

Finally, the game gives the possibility to a user to share his/her performance with friends through Facebook to further increase the user engagement and *Motivation*. Therefore the application, which records confidential data, e.g., user movements and physical activity, has also the possibility to post a message on the Facebook wall of the user. This situation could rise privacy issues, therefore we decide to separate data about the user and his/her friends from data about his/her movements recorded through the smartphone sensors,

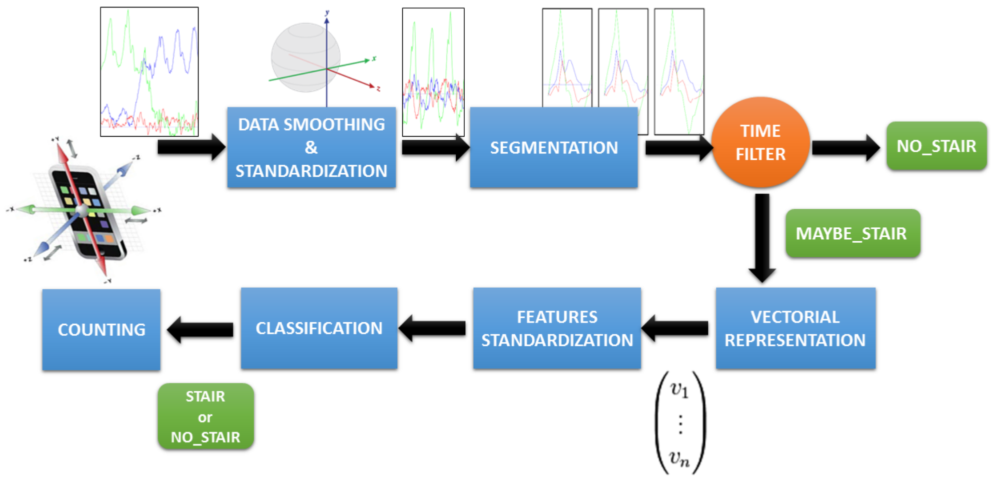


Figure 7.2: Pipeline overview of the system

which are recorded and stored separately. The application has only the possibility to share game scores and results, e.g., the winner of a challenge or the result of a climbing, and not the user location or other data about his/her movements. Data about stairsteps and physical activity are only stored locally on the smartphone.

## 7.3 Pipeline overview

In this section, we give a broad overview of our system for recognition and counting of stairsteps whose modules will be discussed in detail in the following sections.

The application's pipeline is shown in Figure 7.2. When active, the application constantly acquires data from smartphone sensors to identify whether the user is climbing or descending stairs. Since it is very important to provide real time feedback to increase user engagement, we do not rely on a server that analyzes data and provides classification output, but the smartphone performs everything. In this way, we avoid problems of network availability and delay. Moreover, the use of a server can create a bottleneck in the system. On the other side, if all the computational tasks reside on the smartphone, then energy consumption have to be considered and deeply investigated to avoid wasting energy and drain smartphone battery.

The first step of the pipeline is data acquisition: the smartphone acquires data about movements of the device through its accelerometer and rotation sensor.

The second step of the pipeline, called "data-smoothing & standardization", initially cleans data by reducing sensor noise and variability. This operation helps to increase the accuracy of the classification. Then data is standardized, since data acquired by sensors can change a lot depending on how the user carries the smartphone as we will discuss in Section 7.5, and we do not impose to the user a smartphone position. To overcome this problem, we translate smoothed data to a fixed coordinate system, using information from the rotation sensor. In this way, the same user activity data have the same pattern independently from the orientation of the smartphone.

The segmentation step splits the standardized data into consecutive segments or windows. In this module, a window segmentation technique based on data analysis (i.e., without fixing time size of a window) is used to reduce the computational cost, thus reducing energy consumption, and improve the accuracy. The proposed method allows considering time as a parameter. In this way, windows that do not fall into a predefined time duration, i.e., the time necessary to complete a stairstep, can be automatically labeled as not a stairstep without forwarding it to the next step (thus, reducing energy consumption).

After segmentation, the extraction and standardization of the features for the classifier takes place.

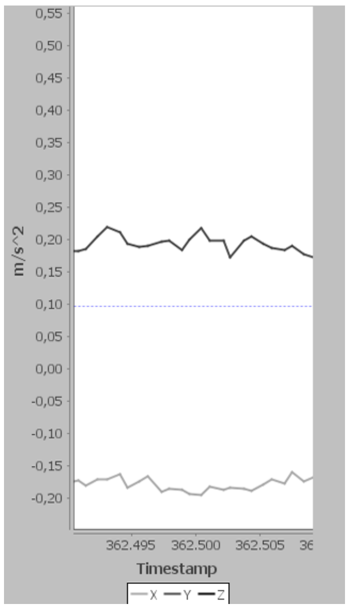
We cast the stairstep recognition task in a classification setting where we have two possible labels (“STAIR” and “NO_STAIR”) and the examples consist of vector representations of segmented windows. Note that, the training of the model is performed offline only once (one single model shared by all different users), while the recognition phase has to be made in *real-time*. This implies that the classification outputs must be promptly provided to the user without delays. We must note here that a simple high-pass filter or a peak detection algorithm would not be sufficient in our context. In fact, as it is shown in Figure 7.3b and in Figure 7.3c, a stairstep and a step have almost the same shape, that is, a data peak on one axis. For this reason, we need a more complex method able to distinguish between this two different activities.

In the following sections, we provide a deep analysis of the main modules compounding the complete system. Then, we present the results of the classification algorithms in a simulation of the system and an analysis of issues related to energy consumption.

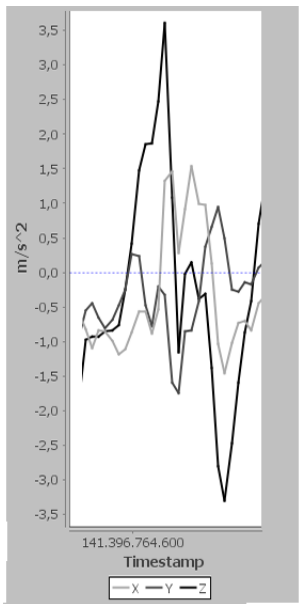
## 7.4 Data smoothing

One of the biggest problems that affect data retrieved from the accelerometer and any other sensor of a smartphone acquiring real-time information from the environment is the noise that affects these data. This noise is caused by sensor precision and sensibility, or by the external environment itself. Let us consider, for example, data from the accelerometer. If we put the smartphone on a table with the screen facing out the sky, and we record data from the accelerometer, what we expect to register is a series of vectors equal to $v = (0.0, 0.0, 9.8)$, where X and Y axes should record values equals to 0.0 $m/s^2$, since there is no movement, and with the gravity force that influences only the Z axis, whose value should be equal to 9.8 $m/s^2$.

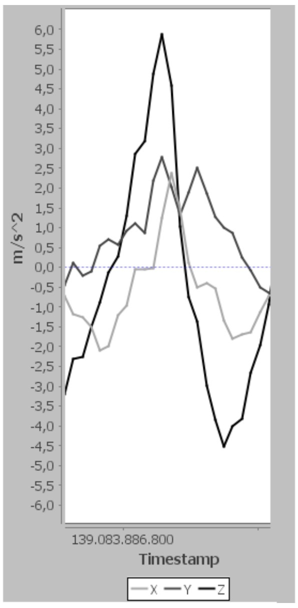
What happens is shown in Figure 7.3a. Each accelerometer axis is affected by both



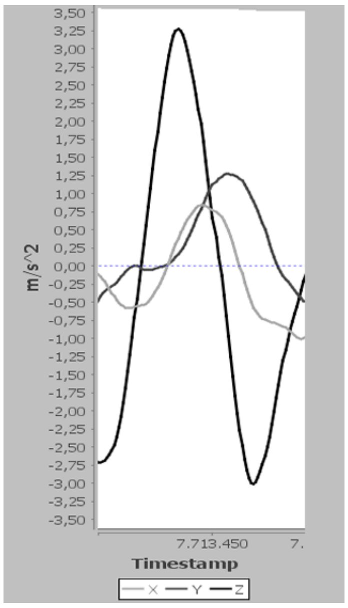
(a) Noise that affects accelerometer data



(b) Example of a step



(c) Example of a stairstep

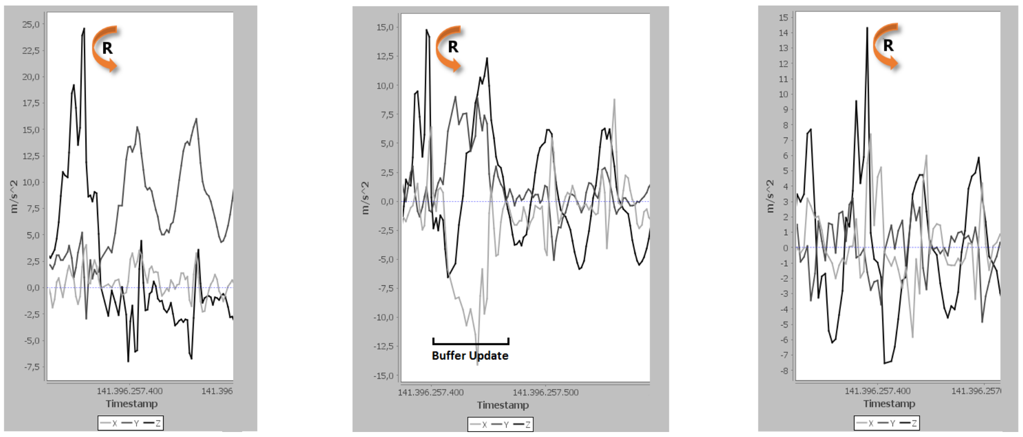


(d) Example of a stairstep using data smoothing

Figure 7.3: Several examples of data acquired from the accelerometer

background noise and sensor sensibility that make the signal very unstable. The presence of this noise makes the classification task more difficult as data is affected by this background error that disturbs the recording of the real movement that we want to measure. With this step we want to reduce the effect of background noise and hence to have cleaner data.

The idea is very simple. We use a buffer which stores the accelerometer records related to the last 300ms. The time length of the buffer is short enough to keep all useful



(a) Raw data acquired from the accelerometer: same activity but different axis values after rotation of the smartphone

(b) Our method applied to a walking activity while rotating the smartphone

(c) Native linear acceleration plus rotation method

Figure 7.4: The comparison among the raw signal (a) and the signal from the methods for the orientation-independence: our method (b), Linear (c). The letter "R" denotes the instant in which the rotation takes place. Each line represents data acquired for the axis X, Y and Z.

information for the classification task (e.g., it is shorter than the time length of a stairstep). Let us suppose that a new vector value $a = (a_x, a_y, a_z)$ of acceleration data is received from the accelerometer sensor. If the buffer is not full, i.e., its time length is lower than 300ms, and then we simply store the vector $a$ into the buffer and nothing more happens. When the buffer is full, we calculate the average vector $m = (m_x, m_y, m_z)$ of every reading stored in the buffer and the vector $m$ is sent forward to the data standardization step. Then, we store the initial vector $a$ into the buffer. In particular, we use a cyclic buffer, so the first element of the buffer is deleted when a new record is added to the buffer. In this way, the background noise is reduced since variations of a single reading are smoothed from all the other readings.

The result of this technique in a stairsteps pattern is shown in Figure 7.3, where it is possible to see the same pattern of a stairstep without (Figure 7.3c) and with (Figure 7.3d) the data smoothing step. As we can see, the signal is much less disturbed and cleaner after the data smoothing step, making the learning task, and hence the classification task, easier and more accurate.

## 7.5 Data standardization

One of the main issue it is necessary to address when working with activity recognition using data from smartphone sensors like the accelerometer, is smartphone orientation. This problem is related to the fact that data from motion sensor changes depending on how the smartphone is carried on by the user. Consider for example a walking activity as depicted in Figure 7.4a. When the smartphone is rotated to a different position from the initial one (the "R" arrow), the data completely changes. In particular, we can observe how accelerometer data completely switches data of the three axes. Consequently, the learning task of the classifier becomes extremely difficult. One of the possible solutions to this problem is to force users to keep the smartphone in a particular position. In fact, some earlier works followed this approach, but this is clearly a strong limitation, since imposing a fixed smartphone position reduces pervasiveness of our application, acceptance of the users, and its persuasive power. Another solution would be to train the classifier with every possible orientation of the smartphone. This is clearly an unpractical solution, since there are infinite possible orientations and thus training a classifier of this type becomes infeasible.

The solution we propose aims at standardizing data to a fixed coordinate system, independently from the orientation of the smartphone. In this way, the same activity is represented by a similar signal behavior, meaning that how the user carries the smartphone becomes an insignificant information, and the task to train an accurate classifier becomes easier.

Mizell in [102] proposed one method to try to solve the problem we just discussed. The idea is the following: given a sampling interval, the gravity component $\mathbf{g} = (g_x, g_y, g_z) \in \mathbb{R}^3$ on each axis can be estimated by averaging over data read on each axis. When an accelerometer produces the original signal $\mathbf{a} = (a_x, a_y, a_z) \in \mathbb{R}^3$, it is possible to calculate the so called *dynamic component* of $\mathbf{a}$ as $\mathbf{d} = (a_x - g_x, a_y - g_y, a_z - g_z)$ where the influence of the gravity is eliminated. Finally, the vertical part $\mathbf{p}$ of the dynamic component $\mathbf{d}$ (parallel to the gravity) is computed as $\mathbf{p} = (\frac{\mathbf{d} \cdot \mathbf{m}}{\mathbf{m} \cdot \mathbf{m}})\mathbf{m}$, and the horizontal part (orthogonal to the gravity) as $\mathbf{h} = \mathbf{d} - \mathbf{p}$.

As a positive aspect, we note that this method requires only to acquire data from the accelerometer, and this makes it less expensive in terms of energy consumption. On the other hand, it may lose relevant information since it translates a three-dimensional coordinate system (the one of the devices with acceleration on three different axes) into a two-dimensional one (vertical and horizontal directions).

Here, we propose a new method to increase the precision of the transformation and to preserve the orientation-invariance property. How this method works and how it changes the signal acquired by the accelerometer is shown in Figure 7.4b. The fixed target coordinate system is the following: the $X$ axis is defined as the one tangential to the ground pointing approximately toward East. The $Y$ axis is tangential to the ground pointing toward the geomagnetic North. Finally, the $Z$ axis is orthogonal to the ground plane and points toward the sky[4]. To implement our method, we need the rotation vector sensor in addition to the accelerometer of the smartphone, and a buffer used to estimate the gravity component acting on each axis.

Firstly, we remove the gravity component from the accelerometer signal using a buffer which stores data acquired during the last 500ms, and averaging over the axes. The vector $\mathbf{g} = (g_x, g_y, g_z)$ is used to compute the dynamic component vector $\mathbf{d} = (a_x - g_x, a_y - g_y, a_z - g_z)$ where $\mathbf{a} = (a_x, a_y, a_z)$ is the original accelerometer reading. Once we have calculated the vector $\mathbf{d}$, it is rotated to the fixed coordinate system using data acquired from the rotation vector sensor. This sensor provides information about the current orientation of the device with a three-component vector, where each component represents a rotation angle around an axis. Now, we can apply a three dimensional rotation to the vector $\mathbf{d} = (d_x, d_y, d_z)$ to obtain a new vector $\mathbf{d}' = (d'_x, d'_y, d'_z)$ representing the real movement of the user with respect to our target coordinate system. As we can see in Figure 7.4b, the signal remains affected from the rotation for a very short interval of time only which is bounded by the buffer size, before going back to the correct coordinate system.

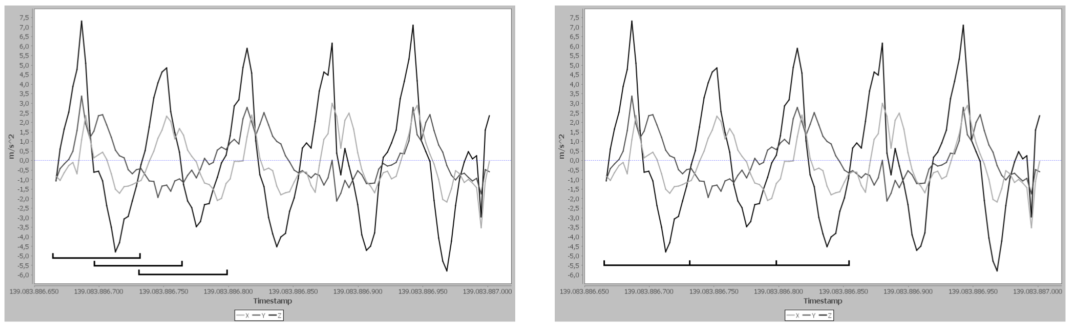
Another possibility to retrieve data about the real movement of the user is to use another sensor, the linear sensor, available on each smartphone. With this sensor, it is possible to retrieve data about the "real" movement of a person without gravity influence. An example of a stairstep data retrieved using the linear sensor is provided in Figure 7.4c. In the following, we will compare our method for gravity removal with the native solution that uses the linear sensor both from a stairstep recognition and an energy consumption point of view and we will show how our solution can reach better performances in terms of $F_1$ evaluation while consuming less energy.

## 7.6 Segmentation with data-dependent window

Once data is standardized to a fixed coordinate system, then the data segmentation step is used to divide data into time segments, i.e., *windows*, from which we extract the features that the algorithm uses to understand if the user is climbing stairs or not.

The standard method to perform activity recognition is by classification of *sliding windows*. This approach is simple: since during data acquisition we do not know when a particular activity starts, a window is defined as the set of temporally consecutive data that lasts a fixed amount of time. When this maximum amount of time is reached, all data belonging to the window is analyzed and classified. With this approach, activities which start in the middle of a particular window can be missed. A natural extension is to consider *overlapping sliding windows*. In this case, windows are built of the same duration and

[4]`http://developer.android.com/guide/topics/sensors/sensors_overview.html`



(a) Sliding windows using time (b) Data-dependent sliding windows

Figure 7.5: The comparison between the segmentation in windows of possible stairsteps obtained from the time-driven sliding windows (a) and the data-dependent sliding windows (b)

overlap. In this way, it is possible to reduce the number of missing activities due to the unknown starting time. Figure 7.5a gives an example of a segmentation of sliding windows with overlapping of 50%.

This kind of approach has shown nice results in different contexts. However, it is not suitable for fine-grained classification problems as recognizing single stairsteps which last for very short intervals of time. The first problem is related to the presence of different user behaviors. Since every user is different from another one, how a person makes a particular activity, e.g., a stairstep, can have different duration and frequency. With a fixed sliding window approach, it is not possible to manage this variability, i.e., a window could be too short, or too long, to give good results for a particular user. Moreover, an initial calibration step which asks the user to perform a particular activity, at his/her speed, to calibrate the size of the window, could decrease the predisposition of the user to use the application.

Another problem of the fixed overlapping sliding window is related to the training phase and involves the input data for the classifier: if we use examples of positive elements, i.e., stairsteps, which perfectly fits inside a single window, the training set is no longer adherent to reality. If we use examples which do not fit into a single window, it is more difficult to discriminate between positive and negative instances.

Finally, the overlapping sliding window approach brings a problem concerning energy consumption. In fact, since data is analyzed twice, due to the overlapping, the total number of analyzed windows increases and negatively affects the duration of the battery.

To reduce the input errors and the computational cost required by classification on a smartphone, we decided to change the way windows are segmented from a *time-based* segmentation to a *data-based* segmentation. In particular, we consider the typical behavior of the signal when a user climbs a stairstep. As shown in Figure 7.3c, the movement is characterized by a positive local maximum followed by a negative local minimum for the $Z$-axis, while the $X$-axis and the $Y$-axis get a much lower variation. Note that, with our method, this pattern is invariant to device orientation. Hence, our assumption is that a window which is suitable to be a stairstep always presents data with this particular pattern. For this reason, we decided to use a *data-driven* approach and define the window when the pattern above is observed.

Using the approach described above, the duration of a window becomes an important additional feature that gives further information and can also be used during classification. In particular, we added a simple filter that discards all the windows that do not respect a predefined time interval, i.e., the time necessary to perform a particular activity (in our case, to make a stairstep). To fix this time interval, we have roughly analyzed our training set and obtained that making a stairstep without running requires, at least, 300ms, and at most 2 seconds. Using this information, the filter moves to the next step all the windows that respect this time interval, discarding all the others. Empirically, the adoption of this simple filter significantly reduces the number of windows to consider, thus saving energy.

Summing up, our method has several advantages with respect to the time-based sliding window. The first one is that we are sure that each stairstep will fit in a window and this makes the learning task much easier. In this way, it reduces the possible errors in the training set. The second advantage is that just the windows suitable for stairsteps are taken into account for the analysis, reducing the total computational cost for the smartphone. Finally, this approach allows to deal with user variability. In fact, even if different users require a different amount of time to complete a stairstep, the window will be appropriately re-sized to contain it without the need of calibration.

## 7.7 Representation and features standardization

In the representation phase, the dynamic window resulting from the segmentation phase is transformed into a vector of real values representing the information obtained by the device sensors contained in that particular time window. This mapping will permit a natural application of standard machine learning methods to our task of recognizing whether the user is climbing stairs or not. The way we represent data is crucial for the effectiveness of a learning algorithm. Specifically, a good representation method will be able to maintain the information which is really relevant for the task and reduce the noise in the data as much as possible.

In the following we formally describe our choices for the representation module. In particular, let the frequency of sampling be fixed. Then, each window obtained by the segmentation step will consist of a sequence of $n$ vectors, $\mathbf{v}_i = (x_i, y_i, z_i) \in \mathbb{R}^3$, $i \in \{1, .., n\}$, each one consisting of the standardized values of acceleration with respect to the three axes.

We also consider two additional computed values that, in our opinion, can carry precious information. Firstly, we added the norm of the vector $\mathbf{v}_i$ which well represents the global amount of activity being performed and, secondly, the average value of horizontal accelerations, namely $\frac{x_i+y_i}{2}$ which combines the horizontal activities in our fixed coordinate system in a single value.

The 79 features are created evaluating standard statistics (i.e., average, standard deviation, variance and difference between minimal and maximal values) which are very common in time series analysis or evaluating the ratio among the statistical features above and the correlations from different sources [127, 115].

Table 7.2 presents a detailed description of the entire set of features we decided to compute to represent a sequence of acceleration measures in a dynamic window. The first set of 20 features considers standard statistics, i.e., average, standard deviation, variance, etc. The next group of 40 features considers the ratio of the same statistics computed on different rows. An additional group of 9 features (61-69) takes in account other important ratio statistics (among vertical and horizontal activities and the norm of the total activity), and two other features, the Magnitude Area (MA) and the Signal Magnitude Area (SMA). Finally, the last ten features correspond to the value of the correlation between pairs of axes. For reasons of computational complexity and energy consumption, we did not use the features in the Fast Fourier Transform (FFT) family.

To fairly compare our orientation-independence supporting method with the other methods, we used a similar representation mapping. In particular, the same mapping is used for the Linear method. Concerning the Mizell method, that returns two measurements vectors instead of one, we calculated features for both vectors, thus obtaining a vector of length $79 \times 2 = 158$.

Finally, it is well-known that the classification algorithms are badly influenced by features with different orders of magnitude. For this reason, we have rescaled all the features from the dynamic windows used to train the classifiers between $[-1,1]$ with a linear transformation, to improve the classification results from the classifiers. The linear transformations (that are fixed and different for each feature) are applied to all the features before the classification takes place to improve the classification performance.

## 7.8 Experiments and results

In this section, we present experimental results we have obtained considering classification performances, energy consumption, *Triggers* firing and users evaluation.

### 7.8.1 Classification setting and results

The first problem to evaluate and solve is related to stairstep classification, evaluating performances of the three different methods (Mizell, Linear and our method) to find the

| Features No. | Description |
|---|---|
| $1-5$ | $Ave(\mathbf{S}_j),\ j \in \{1,\ldots,5\}$ |
| $6-10$ | $Std(\mathbf{S}_j),\ j \in \{1,\ldots,5\}$ |
| $11-15$ | $Var(\mathbf{S}_j),\ j \in \{1,\ldots,5\}$ |
| $16-20$ | $Max(\mathbf{S}_j) - Min(\mathbf{S}_j),\ j \in \{1,\ldots,5\}$ |
| $21-30$ | $\frac{Ave(\mathbf{S}_j)}{Ave(\mathbf{S}_h)},\ j,h \in \{1,\ldots,5\}, j > h$ |
| $31-40$ | $\frac{Std(\mathbf{S}_j)}{Std(\mathbf{S}_h)},\ j,h \in \{1,\ldots,5\}, j > h$ |
| $41-50$ | $\frac{Var(\mathbf{S}_j)}{Var(\mathbf{S}_h)},\ j,h \in \{1,\ldots,5\}, j > h$ |
| $51-60$ | $\frac{Max(\mathbf{S}_j)-Min(\mathbf{S}_j)}{Max(\mathbf{S}_h)-Min(\mathbf{S}_h)},\ j,h \in \{1,\ldots,5\}, j > h$ |
| 61 | $\frac{Max(\mathbf{S}_3)}{Max(\mathbf{S}_5)}$ |
| 62 | $\lvert\frac{Min(\mathbf{S}_3)}{Min(\mathbf{S}_5)}\rvert$ |
| 63 | $\sqrt{Ave(\mathbf{S}_1)^2 + Ave(\mathbf{S}_2)^2 + Ave(\mathbf{S}_3)^2}$ |
| 64 | $Ave(\lvert\mathbf{S}_1\rvert + \lvert\mathbf{S}_2\rvert + \lvert\mathbf{S}_3\rvert)$ |
| 65 | $\frac{Std(s_1)^2}{Std(s_4)}$ |
| 66 | $\frac{Std(s_2)^2}{Std(s_4)}$ |
| 67 | $\frac{Std(s_3)^2}{Std(s_4)}$ |
| 68 | $\frac{(Std(s_1)+Std(s_2))^2}{Std(s_3)}$ |
| 69 | $\frac{(Std(s_1)+Std(s_2))^2}{Std(s_4)}$ |
| $70-79$ | $Corr(\mathbf{S}_j,\mathbf{S}_h),\ j,h \in \{1,\ldots,5\}, j > h$ |

Table 7.2: The 79 features extracted from dynamic windows

best one. To train our classifier, we collected data from 6 different users with different smartphones. This data is recorded keeping the devices consistent with the body movement[5] (e.g., handheld, in a backpack or a handbag) and without other constraints with respect to where or how to keep the device. All the methods presented in Section 7.7 have been used to compute a vector of features. For each dynamic window contained in the whole dataset we computer a vector of features. After that, we manually labeled recorded data to use it with the supervised learning algorithms.

To tackle this task we use algorithms from three different families:

- *Decision Trees (DT)* generated using the C4.5 algorithm,
- *K-nearest neighbors (KNN)* and
- *Kernel Optimization of the Margin Distribution (KOMD)* that is a kernel machine (SVM-like) described in [4], with RBF as the kernel.

For the KOMD algorithm, we used our own implementation, while for the decision trees

[5]For our task, we consider a device movement consistent with the body movement whenever the device movement is negligible relatively to the body's barycenter.

| | **Frequency 20Hz** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Mizell** | | | **Linear** | | | **Rotation method** | | | **Smoothed** | | |
| | Prec | Rec | $F_{1\pm std}$ | Prec | Rec | $F_{1\pm std}$ | Prec | Rec | $F_{1\pm std}$ | Prec | Rec | $F_{1\pm std}$ |
| **DT** | 0.81 | 0.84 | $0.825_{\pm 0.012}$ | 0.67 | 0.70 | $0.685_{\pm 0.024}$ | 0.77 | 0.79 | $0.780_{\pm 0.015}$ | 0.73 | 0.71 | $0.720_{\pm 0.010}$ |
| **KNN** | 0.80 | 0.77 | $0.785_{\pm 0.010}$ | 0.87 | 0.77 | $0.817_{\pm 0.024}$ | 0.80 | 0.75 | $0.774_{\pm 0.012}$ | 0.81 | 0.81 | $0.810_{\pm 0.009}$ |
| **KOMD** | 0.83 | 0.84 | $0.835_{\pm 0.011}$ | 0.83 | 0.83 | $0.830_{\pm 0.018}$ | 0.84 | 0.85 | $0.845_{\pm 0.008}$ | 0.84 | 0.90 | $0.869_{\pm 0.007}$ |

| | **Frequency 30Hz** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Mizell** | | | **Linear** | | | **Rotation method** | | | **Smoothed** | | |
| | Prec | Rec | $F_{1\pm std}$ | Prec | Rec | $F_{1\pm std}$ | Prec | Rec | $F_{1\pm std}$ | Prec | Rec | $F_{1\pm std}$ |
| **DT** | 0.82 | 0.86 | $0.840_{\pm 0.012}$ | 0.80 | 0.78 | $0.790_{\pm 0.024}$ | 0.82 | 0.73 | $0.772_{\pm 0.015}$ | 0.73 | 0.71 | $0.720_{\pm 0.010}$ |
| **KNN** | 0.83 | 0.84 | $0.835_{\pm 0.010}$ | 0.88 | 0.61 | $0.721_{\pm 0.024}$ | 0.90 | 0.78 | $0.836_{\pm 0.011}$ | 0.87 | 0.85 | $0.860_{\pm 0.009}$ |
| **KOMD** | 0.89 | 0.85 | $0.870_{\pm 0.010}$ | 0.86 | 0.83 | $0.845_{\pm 0.018}$ | 0.90 | 0.88 | $0.890_{\pm 0.008}$ | 0.91 | 0.91 | $0.910_{\pm 0.007}$ |

| | **Frequency 50Hz** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Mizell** | | | **Linear** | | | **Rotation method** | | | **Smoothed** | | |
| | Prec | Rec | $F_{1\pm std}$ | Prec | Rec | $F_{1\pm std}$ | Prec | Rec | $F_{1\pm std}$ | Prec | Rec | $F_{1\pm std}$ |
| **DT** | 0.70 | 0.77 | $0.733_{\pm 0.012}$ | 0.82 | 0.89 | $0.854_{\pm 0.023}$ | 0.81 | 0.75 | $0.779_{\pm 0.015}$ | 0.86 | 0.83 | $0.845_{\pm 0.009}$ |
| **KNN** | 0.82 | 0.85 | $0.835_{\pm 0.009}$ | 0.81 | 0.84 | $0.825_{\pm 0.024}$ | 0.90 | 0.78 | $0.836_{\pm 0.011}$ | 0.82 | 0.85 | $0.835_{\pm 0.009}$ |
| **KOMD** | 0.85 | 0.88 | $0.865_{\pm 0.010}$ | 0.79 | 0.86 | $0.824_{\pm 0.018}$ | 0.91 | 0.90 | $0.905_{\pm 0.008}$ | 0.92 | 0.92 | $0.920_{\pm 0.006}$ |

Table 7.3: Experimental results of Precision (Prec), Recall (Rec) and $F_{1\pm std}$ score for the algorithms and methods used with frequency of sampling of 20Hz, 30Hz and 50Hz. The result highlighted has the largest value of $F_1$

and the k-nearest neighbors we used the Weka [68] implementation.

We sampled raw data at three different frequencies, namely 50Hz, 30Hz and 20Hz, with the aim to explore the relation between performances and energy consumption and finding the best trade-off. Higher frequencies correspond to the more accurate information given to the system, but obviously, they also correspond to a higher energy consumption for the device.

In our application, we face with the binary classification task of stairstep recognition. The distribution of different labels, in this case, is very imbalanced since we have far more negative examples than positive ones. For this, we evaluated *Precision* and *Recall*, instead of accuracy, to obtain a more proper performance estimation and comparison of different experimental settings. Then, we used the F$_\beta$-*score* (with $\beta = 1.0$) to combine *Recall* and *Precision* in a single aggregated effectiveness score. The analytic formulas for these measures are:

$$\begin{array}{ll}
Precision & \dfrac{TP}{TP+FP} \\
Recall & \dfrac{TP}{TP+FN} \\
F_1\text{-}score & 2 \cdot \dfrac{Precision \cdot Recall}{Precision + Recall}
\end{array}$$

where $TP$ stands for True Positives, $FP$ as False Positives and $FN$ as False Negatives.

Starting from the original datasets with manually assigned labels for each dynamic window, we performed our experiments using a nested stratified 10-fold cross validation. Firstly, we divided our dataset into 10 partitions (each partition with the same distribution of stairsteps). Afterward, we selected the best parameter for each algorithm using the classical 10-fold cross validation among the windows contained in 9 out of the 10 partitions (i.e., the training set). Finally, we trained the machine learning algorithms to generate a model using the best parameters and we evaluated the classification statistics among the windows contained in the remaining partition, used as test set. We repeated this procedure selecting all the 10 partitions as test set and evaluating the average of the performance. The results of our experiments over the test set are summarized in Table 7.3.

These results show that data smoothing improves our proposed method for the support to the orientation-independence. This technique, when combined with the KOMD classification algorithm, obtains the best performances against any other combination of methods and algorithms. The results are satisfactory considering the high difficulty of the classification task and given the small training sample obtained on each single dynamic window. We are also interested in finding the best trade-off between energy consumption and performances. On this respect, we can see that we obtained the best compromise at 30Hz, since we can notice that the difference in performances between 50Hz ($0.920_{\pm 0.006}$) and 30Hz ($0.910_{\pm 0.007}$) is not statistically significant considering the standard deviation. Moreover, our method with KOMD shows a higher $F_1$ score at 30Hz than any other classifier at 50Hz.

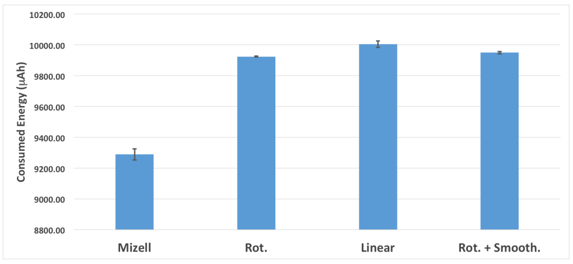
### 7.8.2 Energy consumption results

Energy consumption is one of the key aspects to consider when dealing with smartphone devices and applications. One of the most important aspect for mobile devices users is battery lifetime, meaning that it is not possible to develop an application that wastes a lot of energy and asks the user to recharge smartphone more than once a day. This kind of situation usually takes the user to remove the application.

A deep analysis of each step of the pipeline is a fundamental requirement during design and development. In our particular case, we wanted to study how resources are used and which are the computationally expensive tasks. Furthermore, we wanted to understand how data frequency acquisition from the accelerometer and the rotation sensor affects energy consumption and in which measure, how much energy the data-smoothing step requires and how well the data segmentation technique performs with respect to the *sliding window* technique.

To measure the energy consumption, we used the Monsoon Power Monitor[6], since this is the best solution to measure energy consumption, as already explained in Section 4.2.1.

[6]`http://www.msoon.com/LabEquipment/PowerMonitor/`



| Energy consumption ($\mu$Ah) | | | |
|---|---|---|---|
| **Mizell** | **Linear** | **Rotation method** | **Smoothing and rotation method** |
| $9289_{\pm 36}$ | $10003_{\pm 21}$ | $9923_{\pm 3}$ | $9949_{\pm 8}$ |

Figure 7.6: Energy consumption of the four different methods

We performed our tests using a Samsung Galaxy i9250 with a 1750mAh battery. Even if this smartphone is not up to date, we considered this not a real problem since we do not want to consider absolute values of energy consumption, but we want to compare energy consumption of different methods to determine which is the less expensive.

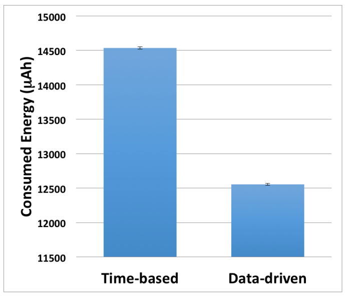
Figure 7.6 reports the energy consumption measurements. Here and in the following discussion we just consider the "Consumed Energy" value, which is the energy the smartphone is requesting to the battery when acquiring data from the sensors (accelerometer and rotation sensor) and running one of the methods we consider in our analysis. Each experiment lasted two minutes and we repeated each test three times for each method, as already done from cross-platform energy consumption analysis presented in Section 4.2.1. We tested our method using the target frequency of 30Hz.

The first evidence we can note from our results is that the Mizell method is the less consuming one. This is not surprising since this method gets data from the accelerometer sensor only, thus requiring less energy. On the other side, this method shows a significantly poorer performance (see Table 7.3).

Comparing our method, which uses both the rotation sensor and the accelerometer, and the linear method, which use the rotation sensor and the linear sensor, we can note that the usage of the accelerometer is less demanding than the usage of the linear sensor (both of them plus the rotation sensor). These results along with the ones presented in Table 7.3 demonstrate that our solution can improve both the accuracy in terms of stairstep recognition and the energy consumption with respect to the linear solution. Moreover, when adding the data smoothing phase, the performances in accuracy are further improved without affecting the energy consumption (only about +0,26%).

Another key aspect of our pipeline is the use of data-driven windows that, differently from standard time-based windows, uses the data pattern to determine how to segment data coming from sensors. To compare the two methods in terms of energy consumption, we tested both of them acquiring data at 30Hz and using KOMD as classification algorithm. For time-based windows, a time length of 500ms and overlapping at 50% were used. Final results are reported in Figure 7.7.

As we can see from our data, data-driven windows allow us to save more energy



| | Segmentation approach | |
|---|---|---|
| | **Time-based window** | **Data-driven window** |
| **Energy consumption ($\mu$Ah)** | $14535_{\pm17}$ | $12554_{\pm14}$ |

Figure 7.7: Energy consumption comparison between time-based and data-driven windows

(more than 15%) with respect to the time-based sliding windows. This is because in the time-based approach we have to classify every window, while, in our method, we take into consideration only the ones that contains the correct pattern. Moreover, the time filter let us further reduce the number of analyzed windows, and cannot be applied to standard windows. Clearly, less windows analyzed by the classifier means less energy consumption.

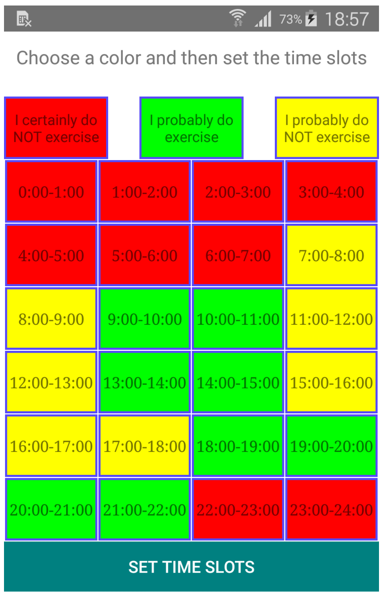
### 7.8.3 *Triggers* design

As already presented in Section 7.2, one of the most important aspect that influences the possibility or not to change people behavior are the, so called, *Triggers*. To be efficient, one of the most important requirement for *Triggers* is that they should happen at the right time. This is a very difficult issue, since, for example, if we want to suggest to our user to take stairs instead of elevators or escalators while he/she is working and cannot move from his/her desk, or simply there are no stairs that he/she can reach, this *Triggers* are absolutely useless, if not disturbing, and will not help in behavior change. Moreover, *Triggers* can also be used to keep users involved in the game.

The task of choosing the right time to fire a *Trigger* is hard to solve. The simplest solution, presented in [135], is to ask the user to set by himself/herself an alarm for the next day, at a time that he/she thinks could be the right moment to perform the target activity or, in general, some physical activity.

This very simple solution has some drawbacks: it is not ubiquitous, and it is boring for the user since it requires the person to set every day the alarm for the following day. Therefore, we propose something different, a solution which can adapt automatically to user habits, showing *Triggers* only when it is supposed to be the best moment of the day. This solution is ubiquitous and not intrusive, since it requires only an initial setup, and afterward, it can understand user habits and the best moments to show *Triggers*. The idea is to silently sense user activity, in order to learn his/her habits.

The first step of the algorithm, performed only at the first access to the application, is to ask to the user information about his/her daily routine. Figure 7.8 shows the user interface which allows to give an initial knowledge to the algorithm. The user has to indicate in



| 🟥 I certainly do NOT exercise | 🟩 I probably do exercise | 🟨 I probably do NOT exercise | |
|---|---|---|---|
| 🟥 0:00-1:00 | 🟥 1:00-2:00 | 🟥 2:00-3:00 | 🟥 3:00-4:00 |
| 🟥 4:00-5:00 | 🟥 5:00-6:00 | 🟥 6:00-7:00 | 🟨 7:00-8:00 |
| 🟨 8:00-9:00 | 🟩 9:00-10:00 | 🟩 10:00-11:00 | 🟨 11:00-12:00 |
| 🟨 12:00-13:00 | 🟩 13:00-14:00 | 🟩 14:00-15:00 | 🟨 15:00-16:00 |
| 🟨 16:00-17:00 | 🟨 17:00-18:00 | 🟩 18:00-19:00 | 🟩 19:00-20:00 |
| 🟩 20:00-21:00 | 🟩 21:00-22:00 | 🟥 22:00-23:00 | 🟥 23:00-24:00 |

Figure 7.8: Initial setup of time slots to provide initial information to the algorithm

which time slots (the time duration is of about one hour) performs/probably performs/do not perform physical activity. In this way, our algorithm can understand which are some time slots where no physical activity is performed (e.g., when the user sleeps). These slots are marked as red and our algorithm does not consider them. The other time slots will be defined as exploring intervals, where the algorithm will evaluate physical activity and understand which ones are interesting because the user is probably making stairsteps.

After the initial configuration, the algorithm defines two different kinds of exploration time slots: it marks as green the time slots in which the user declares to perform physical activity and as yellow the slots in which it declares a probability to performs some physical activity. We use our stairstep classifier, which is more expensive in terms of energy consumption, to sense the user activity during the green intervals of time, and the Google Activity Recognition service [65] to understand if, and in which measure, some physical activity is performed during the yellow intervals. Since the stairstep classifier is more expensive of the Google Activity Recognition Service in terms of energy consumption, we use our classifier only when we are sure that the user will perform physical activity.

The algorithm senses the user activity for a week and then decide when to listen his/her activity in the next weeks and when to fire *Triggers*. How the algorithm evolves is easy:

- if during a yellow time slot the Google Service returns an evaluation $v(x)$ so that $0 \leq v(x) \leq 0,5$, this time slot will not be considered the next week (it is marked as red);
- if the evaluation $v(x)$ is $0,5 \leq v(x) < 1$ this time slot will be used next week with the stairstep classifier (it is marked as green);

- if during a time slot marked as green the user make more than one stairstep, the time slot remain green, otherwise it is marked as yellow.

The user can change his/her behavior during time, or the initial description about his/her habits can be inaccurate. To allow an automatic adaptation of the algorithm to changes in user routines, a random exploration of the time slots marked with red is implemented. We introduce a random probability of mutation: a red interval can be mutated into a yellow one. In this way, we use Google Service to analyze physical activity, and the same interval can be better analyzed the following week if the user performs a sufficient amount of physical activity.

*Triggers* notifications are implemented considering intervals marked as green. First of all, to avoid to disturb too much the user, we decided to notify at most two *Triggers* during the day, separated by, at least, 6 hours. Moreover, we do not notify the user if he/she is already playing with *ClimbTheWorld*. If these conditions are satisfied, the *Trigger* is notified if the number of stairsteps made so far that day does not overtake the number of stairsteps done the day before plus 10. In this way, we aim at keeping constantly active the user and try to increase his/her performances among the days.

Finally, we want to underline that we kept into consideration even energy consumption, especially when analyzing time slots using the Google Service. In particular, if the battery level $L$ is $40\% \leq L \leq 55\%$, information about the current activity is recorded every 10 seconds, if $30\% \leq L \leq 40\%$, activity is analyzed every 20 seconds. If the level of the battery is lower than 30%, we stop the time slot analysis and so the *Trigger* notification. In this way, we avoid to be too intrusive, draining the battery.

### 7.8.4 Users Tests

In order to evaluate the persuasive power of *ClimbTheWorld*, we conducted a test with real users, at the end of which we provided a small questionnaire to understand their impression of the game.

The test involved 13 participants, 8 females and 5 males, aged between 24 and 30. The experiment lasted 9 days, and each participant used his/her smartphone. We created a Facebook account for each one to preserve their privacy (in this way they did not need to ask friendship to someone they did not know) and have full access to all functionalities provided by the game. We did not ask participants to change their daily routine, we asked them to freely use the application whenever they could or want.

We divided our participants into two different groups of 7 and 6 users, respectively. Each participant was randomly assigned to one of the two groups. We used these two groups to randomize our test. Since we provide two different game modes, singleplayer and multiplayer, we ask one group to use the singleplayer mode first and the multiplayer mode afterward, while the second group started with the multiplayer mode.

The complete organization and task order of the two groups is provided in Table 7.4. The first two days we asked participants to use the "Step counter" mode to record a baseline about the number of stairsteps made without a serious game. After the first two days, the first group played two days in singleplayer mode and the following three days in multiplayer mode, and vice versa for the second group. Both the groups concluded our experiment with other two days using only the "Step counter".

| **Day** | **Group A** | **Group B** |
|---|---|---|
| 1 | Stairstep counter | Stairstep counter |
| 2 | Stairstep counter | Stairstep counter |
| 3 | Singleplayer | Multiplayer (*Social Climb)* |
| 4 | Singleplayer | Multiplayer (*Social Challenge*) |
| 5 | Multiplayer (*Social Climb*) | Multiplayer (*Team vs Team*) |
| 6 | Multiplayer (*Social Challenge*) | Singleplayer |
| 7 | Multiplayer (*Team vs Team*) | Singleplayer |
| 8 | Stairstep counter | Stairstep counter |
| 9 | Stairstep counter | Stairstep counter |

Table 7.4: Game mode order for each of the two groups.

At the end of the experiment, each participant completed a 5-point Likert questionnaire, based on the *IBM Computer Usability Satisfaction Questionnaires* [94], with possible answers ranging from "Strongly disagree" to "Strongly agree". Our questionnaire was divided into two different main sections: the first one aimed at having a description of the participants that took part to the experiment, e.g., what they think about physical activity and being physically active, which kind of game players they are (single or multiplayer), etc. The second part of the questionnaire contains questions about *ClimbTheWorld*, the different game modes and their experience during the days.

Moreover, during the experiment, a background logger service was used to trace user activity with the game. We record activities performed with our game and, in particular, the number of stairsteps participants made during each day and with each game mode. In this way, we could understand how the number of stairsteps changed during the experiment, in relation with the game mode participants played.

The first part of the questionnaire showed that 53% of participants does not play any sport, while the other 47% performs an individual sport, and 77% of participants have positive feelings about being physically active. Moreover, about 61% does not frequently use elevators or escalators, but prefer to take stairs in order to be more active. Finally, 54% of them think they do not need any form of external stimulus to keep active.

From the obtained answers we can say that our participants were a difficult test case, since they already prefer to use stairs and have an active life, meaning that, actually, they do not need a *serious game* to increase physical activity. Considering questions and answers about their relation with video games, they defined themselves mainly as “casual players” (69%), that play almost alone (61,5%) or with another player in the same room (46,2%). Finally, only 23% of participants frequently play with mobile games.

Analyzing participants experience with *ClimbTheWorld*, about 70% of participants preferred to play in single mode, while only 38,5% preferred to play with his/her friends. Moreover, 92,2% of participants liked the “Solo Climb” mode and no one said he/she would not play again with it. These data are in accordance with participants description about their preference for singleplayer with respect to multiplayer: the multiplayer mode obtained less appreciation. In particular, within the multiplayer modes, the most preferred one was the “Social Climb” mode, since all the participants used it at least one time and 92,3% of them ranked it positively. The second preference was “Social Challenge”, played by 77% of participants and 80% of them liked it. Finally, “Team vs. Team” mode was played by 61,5% of participants and 63,6% would play again with it. This rank can be explained by the fact that this mode, that should be the most challenging and engaging one, has the drawback that it is difficult to set up, since it is necessary to find at least 4 users, active in the same interval of time, to be able to start the game (and this could take time that not all users are happy to wait for).

The second step of our analysis focused on the data acquired with our logger. Data logger registered how many stairsteps participants made during all the days, and which game mode they used.

We compared answers provided with the questionnaire with objective data, and performances of participants depending on the game mode used. Figure 7.9 shows the number of stairsteps made by all the participants among all the days of our experiment. Together with Table 7.4 the figure also shows the number of stairsteps made by partecipants depending on the used game mode: the number of stairsteps made using the serious game (both in singleplayer or multiplayer) is higher with respect to the number when using simply the stairstep counter. In particular, singleplayer mode increased the average amount of about 61%, while multiplayer of about 64%. This means that the game is effective in incentivizing people in taking stairs, and this is even more important since our test groups were made by people that think that they do not need to be incentivized to be physically active.

Moreover, we can note that, when used, the *Team vs. Team* game mode allows to reach the highest number of stairsteps made. This probably comes from the fact that this game mode combines both collaboration and competition among users, a combination that is able to engage participants and create high motivation. On the other side, the big difference between the two groups even shows the limitation of this game mode, since the

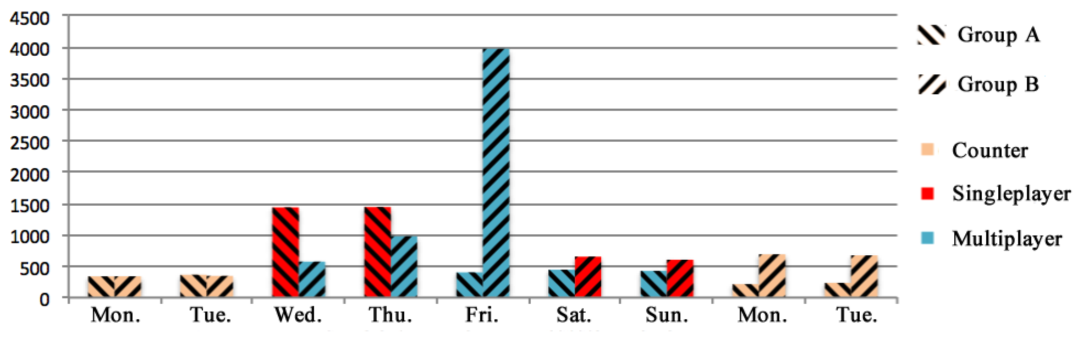


Figure 7.9: Number of stairsteps made by both groups each day of the test.

setup phase is longer than other game modes and could reduce users interest.

There is another important difference in the behavior of the two groups. In fact, the second group, the one that uses the *Team vs. Team* game mode, approximately doubled the number of stairsteps made with the simple counter in the last two days, while the first group lowered the number of stairsteps made without the game in the last two days with respect to the first two days of the experiment. This means that the *Team vs. Team* game mode is not always accepted by the users due to the initial setup phase, but, if used, is able to acquire good results in persuading people to change their behavior, and this result also remains in absence of the game. On the other side, singleplayer games were able to engage both groups, showing how and easy entry setup of the game makes it more engaging.

As we can see from these results, it is clear that *ClimbTheWorld* is really effective in incentivizing people in taking stairs, and both singleplayer and multiplayer modes are engaging and appreciated by users.

# 8. *PlayWithEyes* for visual acuity assessment

In the previous chapter, we presented *ClimbTheWorld*, an example of an application of the *serious games* paradigm in the medical field, in particular for behavior change. Since the medical field is extremely wide, it offers the possibility to introduce *serious games* not only for behavior change but even for other purposes, i.e., patient monitoring, medical diagnosis, rehabilitation, etc. In this chapter, we will focus our attention on diagnosis and rehabilitation of particular diseases.

In particular, let us consider, for example, some diseases like dyslexia or amblyopia where an early diagnosis of these diseases can increase the possibility to revert the condition and reduce its influence in people' life. The earlier some particular diseases are identified and diagnosed, the earlier it is possible to start a treatment and try to solve the problem. Since an early diagnosis is the best approach for this kind of diseases, screening is very important for children, and this new set of target users require a completely different approach to reach the best results.

When considering children as the target of exercises for a medical diagnosis, several problems arise. This comes from the fact we cannot consider children simply as young adults, and for this reason, the same test exercises do not provide the same accurate answers that we would get from adults. This is because one of the biggest problems with children is to maintain their attention during all the duration of the test. This is extremely important, especially for ophthalmologists, since the accurateness of the diagnosis of the answers provided by the child. If he/she stops to provide the real answer or to pay attention to questions only because he/she is tired or bored by exercises, the diagnosis will be inaccurate.

Another important aspect of medical treatment is related to the rehabilitation program. Even here, if an early diagnosis is necessary to reduce the influence of a particular disease in children' life, event the rehabilitation program has to face the problem that the target users are children and not adults, and the rehabilitation exercises cannot be the same. In particular, one particular problem it is necessary to avoid is the *drop out from therapy* phenomena, that would mean that children stop their therapy because they do not like the exercises they have to perform. With *serious games* we can make these exercises more engaging, more appreciated by children and letting the rehabilitation last longer with better results. An example of this approach is our *serious games* for the rehabilitation of children affected by dyslexia [49], where we build a set of *serious games* in collaboration with a psychologist to engage kindergarten children in rehabilitative exercises for this particular disease.

For this reason, *serious games* can be used to transform a normal medical test into a funny moment for the child and at the same time to perform a medical test and provide a diagnosis. In this way, using a game, it is possible to maintain the attention of the child for a longer time, having his/her attention and having from him/her accurate answers to the test questions.

From this observation, we developed several *serious games* both for medical diagnosis. For example, we developed a set of portable games, e.g., that can be used with a smartphone, tablets or even a PC, for the rehabilitation of children affected by dyslexia, that can play with them without knowing that in reality they are performing a rehabilitation exercise.

Moreover, we developed, in collaboration with a group of psychologist, a system for the diagnosis and rehabilitation of children affected by *Cerebral Visual Impairment* (CVI) [32]. This system is based on two *serious games*, called *HelpMe!* and *CatchMe!* and an eye-tracker able to understand where children look at the screen while playing with the game. In this way, the doctor can obtain objective data about the performances of the child and if he/she is improving among the therapy. Moreover, since the child is engaged with this games, the tests can last longer, and the doctor can obtain a higher amount of data with respect to the standard exercises.

In this chapter we present another example of *serious games* applied to the medical field, *PlayWithEyes*, for the early identification of amblyopia and daltonism in preschool children. Differently from already existing system, the children interact with the system using a touch interface, i.e., an Apple® iPod Touch, which displays several possible answers to the test. During all the development we payed much attention in minimizing system requirements and encumbrance to increase portability: it only requires a tablet, a smartphone, and a monitor.

A simple test for each child is based on several exercises, where we project "questions" on the wall and we use an iPod Touch to present the possible answers to the child. For the acuity test, the child has to choose the right letter or symbol showed on the wall. The system provides different acuity tests, using letters, Lea symbols or the "E" chart, also called tumbling "E", i.e., different rotation of the letter "E". Daltonism tests require children to distinguish a path inside an Ishihara plate.

One of the main goals is to provide a very friendly interaction, even for younger patients. For this reason, we create two interaction modalities, a very simple one for younger children, and touch interaction more similar to games to involve older children. The use of the *serious game* paradigm helps to improve the diagnosis of very young children, because we are able to gain the attention with the game, since the child has fun, while we perform our medical tests, allowing the doctor to better observe the little patient.

We tested *PlayWithEyes* with 65 children from a kindergarten, with ages ranging between 3 and 5 years old. The evaluation of our tests showed that we reach positive results both for the *serious* part of the system, i.e., test and evaluate visual acuity and daltonism in children with an efficient use of the time, and from the *game* part of the system, i.e., children were engaged by our games and positively accept the exercises. The combination of this two evaluations shows that our system can be a valid substitute to the normal tests used by ophthalmologists to measure visual acuity in children, especially for screening in kindergarten.

## 8.1 Related Works

Since our system is specifically intended for preschool children, we must pay particular attention to the interface and how to engage children in this particular way of testing their visual acuity. Many papers address the problem of interfaces for children. One possibility is to use what are called *tangible interfaces*, that use physical artifacts for accessing and manipulating information, as shown in [116]. In this paper, preschool children navigate and access images stored in a database through physical and magic objects of wizard Zurlino.

On the other side, Forlines et al. [59] investigated the differences between mouse and direct touch input, both in terms of quantitative performance and subjective preference. They conclude that touch interfaces, even if they may not lead to greater performance, especially for speed and accuracy, are preferable for other considerations like fatigue, spatial memory and simplicity. This is particularly true for children, that are even called "Digital Native Speaker", where touch interaction seems to be much more natural for them. Therefore, not much training is necessary to teach them how to interact with touch applications.

Other papers in literature present the design of *serious games* to deal with children affected by particular diseases. Kato explored the positive aspects of using existing commercial video games for health improvements or surgical training, and tailor-made game for particular disease group to improve recovery and rehabilitation of patients [83]. She defined Re-Mission, a game to help young patients to understand and deal with cancer: game characters represent the drug which destroys cells with cancer. Moreover, the game also provides patients a forum to discuss together.

*DYSL-X* [9] integrates Dyslexia predictors in a tablet game, to capture children attention to obtain a more accurate measurement. The authors evaluate several existing games for preschoolers to derive a set of guidelines to design an optimal tablet game for the 5 years children, then these guidelines are used to develop Diesel-X, a game about a robot dog, Diesel, which has to fight against a gang of criminal cats.

*HelpMe!* [32] is a serious game to help the rehabilitation process for children affected by CVI (Cerebral Visual Impairment). The game is able to adapt the rehabilitative exercises to each particular child, and can follow the improvements of the patient, to reduce the influence of this disability in future life. Moreover, the system can help doctors to perform a good assessment of a patient and to create a rehabilitation program.

Gaggi et al. [63] developed a set of *serious games* to identify Developmental Dyslexia to detect this disability at early stages and immediately start a therapy. These *serious games*, with the characters from the *Nemo* cartoon, were designed to be accessible from any devices, both tablets using a touch interaction or a PC using a keyboard and a mouse, to give the possibility to the doctor to choose the best interaction method depending on the age and the ability of the child.

### 8.1.1 Existing Software

A certain number of software products that provide vision tests exist, both for professional and personal use. At the state of the art the main professional solutions that can be adopted by ophthalmologists are listed below:

- 20/20 Vision (Canela Software) [24]
- PVVAT$^{\text{TM}}$ (Precision Vision) [118]

Both products are desktop applications for professional use. The software shows the tools used by ophthalmologists (eye charts, Snellen tables, etc.) on the desired screen to test patients. The screen can be either a computer monitor or an external display. Both products do not focus on testing children patients specifically, even though Lea optotypes can be used. Moreover, these solutions do not use a game paradigm to improve users' attention.

Given the proliferation of mobile phones, another set of interesting software solutions are applications that allow users to self-test their visual acuity with a cell phone or a portable device. The relevant applications are:

- Acuity (Intellicore) [78]
- EyeChart HD (Dok LLC) [46]
- Vision Test (3 sides cube) [1]
- FastAcuity Lite (KyberVision Consulting, RD) [88]
- Vision (AppZap) [8]
- iC Pro: The EyeTest (Konrad Feiler) [86]

All the listed applications for iPhone, iPod Touch or iPad, behave almost in the same way with nonperceptible differences. They propose daltonism tests based on Snellen tables and acuity tests based on eye charts and Landolt rings. Applications like these, often developed without strong specifications, provide nothing more than simple self-made tests with no real implications.

Comparing our solution with these state-of-the-art applications, it is clear that our improvements are both in the *serious* and in the *game* part of the system. For the *serious* part, as we will see in Section 8.5.1, our system allows to correctly evaluate visual acuity of children as normal tests used by doctors. For the *game* part, differently from other applications, we consider and tackle the problem of children engagement, showing how our system is able to engage children and so to perform eye tests without annoying them, with the risk of having wrong answers due to a reduction in their concentration. Moreover, our system can work in any situation, in terms of distance from the projected optotypes, size of the screen, availability of a wireless connection and age of the children.

## 8.2 Description of the game *PlayWithEyes*

*PlayWithEyes* is a serious game which can be used both in kindergartens and doctor's surgeries with different implications. In the first case, the game uses a set of pre-configured tests, studied by a group of ophthalmologists, so the teacher can easily test children during a normal school day, simply inserting their names. Data about children can be added in the database and classroom can be created before the test, in order to lower the waiting time between a child and the next one. This is particularly important because we do not want the children to get impatient while waiting, since bored patients can provide casual answers, during tests, only because they are tired. Each child is tested individually and the report, obtained while playing, indicates if the child needs a specific visual examination by an ophthalmologist. In this way, our system helps a quick and widespread screening of children.

If used by a doctor, our system is just a facility for the ophthalmologist to obtain a better cooperation from the children, because as long as they have fun, they continue playing, thus testing their visual acuity. This helps to obtain a more precise sight assessment, since the eye test may last longer, and the child maintains the attention, thus improving the possibility to perform a correct diagnosis for very young children.

Finally, the system has to store and retrieve data acquired during tests. In particular, for each tested child the system has to store the proposed test and the answer of the child, eventually reporting children with evidence of visual problems.

### 8.2.1 Visual acuity tests

Visual acuity tests were initially designed using the standard Lea symbols [92]. Lea symbols, shown in Figure 8.1, are specifically defined for preschool children because of their simplicity: children are able to refer to them without ambiguity even if they cannot read letters. The original[1] symbols are four: a house, an apple, a square and a circle.

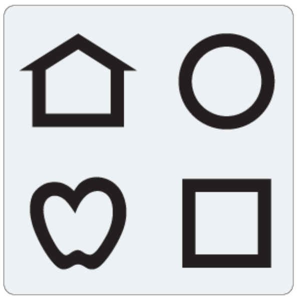


Figure 8.1: Lea symbols used to test visual acuity for children

Another possibility is the use of the tumbling "E". The child has to select the correct orientation of the projected letter.

The most common optotypes used for letters are the Snellen optotypes (Figure 8.2) but, they do not include all the different letters, so we need to draw our optotypes for the remaining letters. Each letter is drawn in a 10x10, resizable, grid, where each cell can be colored in black or white. To write letters which need curves, e.g., letters "B" and "C", some cell can be half colored. The final list of optotypes is shown in Figure 8.3.

**C D E F L O P T Z**

Figure 8.2: Snellen optotypes with alphabetical letters

Symbols and letters can be displayed one at a time, or inside a row or a matrix. The first type of test is simpler for the children, the other two, also called "crowding test", increase the difficulty for the children, who have to recognize a symbol inside crowded

[1]Some minor variants with additional optotypes are also used in ophthalmology.

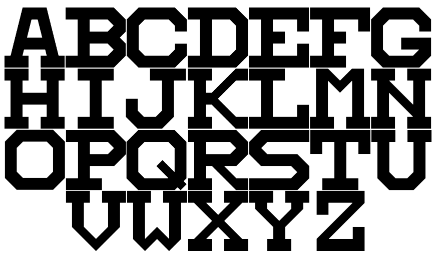


Figure 8.3: Our letters optotypes

interface. In this case, the symbol to recognize is indicated by an arrow, when symbols are displayed on a row, or by a surrounding square, when symbols are displayed in a matrix.

### 8.2.2 Test for color blindness

As mentioned before, *PlayWithEyes* has been developed even to test Daltonism. The first version of the Daltonism tests asked the child to recognize the right color of a chameleon that is projected, choosing between four different possibilities.

During tests, several problems arose. Despite the initial decision to use only 6 different colors proposed by Lea, the difficulties related to the representation of these colors in an external monitor (like brightness, contrast and gamma correction of the monitor or the brightness of the environment) negatively affected the effectiveness of the tests, since it is not possible to guarantee that a given color is represented in exactly in the same way in any monitor (a description of this evidence is provided in Section 8.5.1). For these reasons, we decided to introduce a new kind of test. Despite recognizing the color showed on an external monitor, the child has simply to recognize a path that is unrecognizable for children affected by daltonism. We use the Ishihara plates shown in Figure 8.4, that represents a path inside a circle, using specific colors not distinguishable for persons with daltonism[2]. Images projected on a wall or showed on a screen to the child are colored, but the interface for the child shows the black and white version of the images, where the background is black while the internal path is white. Figure 8.5 shows the black and white conversion of the Ishihara plates.

### 8.2.3 Interaction methods

The system lets the possibility to the child to interact with the game in two different ways, depending on his/her capabilities to deal with touch interfaces: touch or drag interaction.

Let us consider the touch interaction. This is the easiest way for the child to provide an answer, since it requires the lowest level of interaction (and complexity) for the child. He/she has simply to touch the image that represents his/her answer as depicted in Figure

[2]We note here that paths represented in Figure 8.4 cannot be identified when the figure is printed in black and white since Ishihara plates aim to evaluate color blindness.

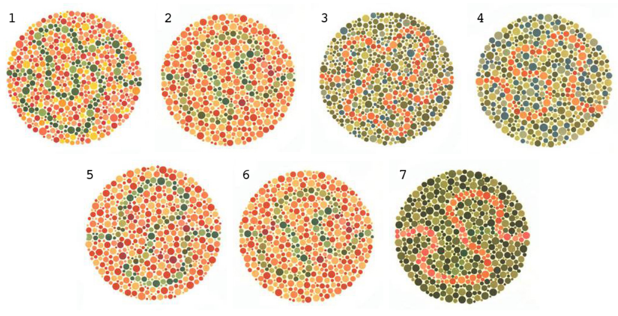


Figure 8.4: Ishihara plates

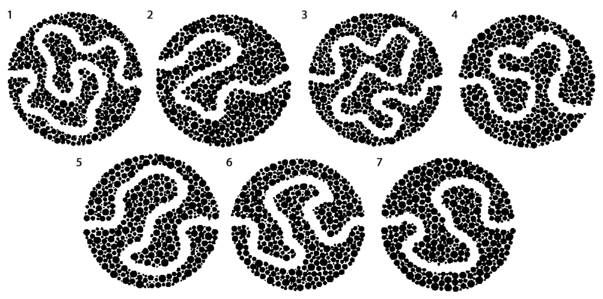


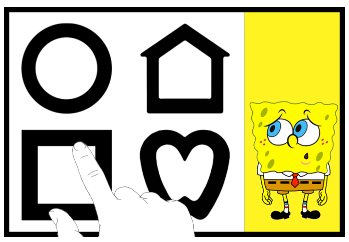
Figure 8.5: Black and white version of the Ishihara plates

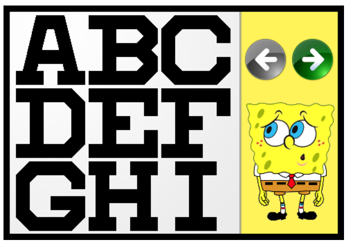
8.6.

To increase the engagement of child, which is extremely important if we want to maximize the concentration of the child and the correctness of his/her answers, several solutions have been adopted. First of all, the test and its possible answers (symbols or letters) are decorated with some cartoon characters on the background, in a number that does not create negative interference with the examination during the gameplay. Secondly, some non-misleading animations and sound are used every time he/she provides the answer to the test, e.g., an applause in case of correct answer, to enhance the game experience.

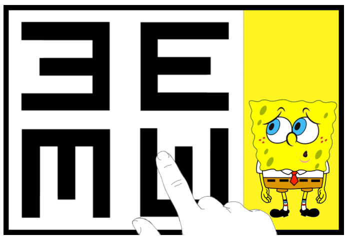
The second interaction mode uses the dragging or panning operations, moving an image on the surface of the touch screen: in Figures 8.7(a) and (b) the user has to move the cartoon character (in particular, an image of the *SpongeBob SquarePants*® world) over the right answer. This kind of interaction is more difficult than the first one, and is more suitable for older children. In case of the rotation test, which uses "E" chart, the child has to rotate the letter until the position on the screen corresponds to the projected one, and submit the answer clicking on another *SpongeBob*® image (see Figure 8.7(c)).

The touch interaction is the only one available when dealing with all the letters since the total number of choices for the question is too high to be able to show all of them with the iPod/iPhone screen in a way that it is possible to move a cartoon character over one of them, since the images of the letters become too small. So, the set of possible answer is divided into two different subsets, and the child uses two arrows and the touch interaction


(a) Crowding test with Lea Symbols



(b) Crowding test with letters


(c) Rotation test

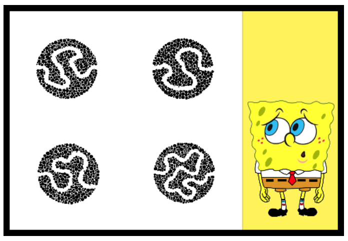
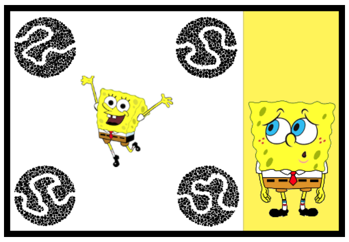
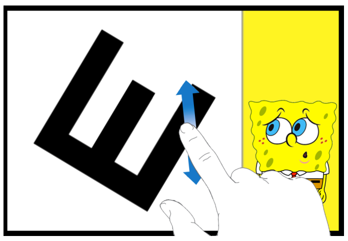

(d) Daltonism test

Figure 8.6: Child interface with touch interaction


(a) Daltonism test


(b) Rotation test

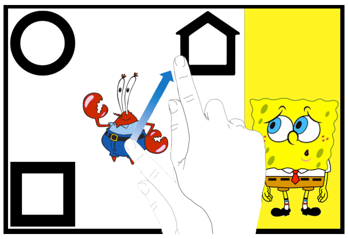

(c) Crowding test with Lea Symbols

Figure 8.7: Child interface with drag interaction

to move between the two subsets as depicted in Figure 8.6 (b).

## 8.3 System Requirements and Architecture

Users of the system may have essentially three different roles:

1. **Administrator:** can configure the set of games proposed to the children. Moreover, it sends data about patients and their game performances to the doctor. The teacher or the doctor may have this role;

2. **Player:** children who play with the *serious game*;
3. **Consultant:** data collected during the game play can be stored in a Microsoft Excel file and sent by email to a consultant. Otherwise, he/she has the possibility to read this data and evaluate performances of each child using the system through the doctor's interface.

*PlayWithEyes* has a classic server-client behavior, based on the following components:

- **Server:** used by the administrator to configure the vision tests. It is responsible to create, manage and show the exercises, but also to provide data to consultant for further analysis;
- **Client:** used by the children to play the game, it is responsible to show the possible answer and to collect the users choices.

In addition to these components, an external display is necessary to show the acuity or daltonism tests. In this case, we can use either an external monitor or a projector that shows the images on a wall. Figure 8.8 shows the architecture of the system. As we can see, the architecture follows a common client-server architecture. The server is responsible for managing the communication with the client that is used by the child to submit the answers. Moreover, it is connected to the projector or the external display and manages the images to show. Finally, it must save, in a privacy-preserving mode, all the data acquired during tests, and to forward this data only to consultants for the following analysis.

The system has been designed to perform the game in 5-10 minutes per child, a time sufficient to test visual acuity and the ability to distinguish colors. Every game session is organized as follows:

1. the test symbols are projected on the display, with background images that show cartoons characters, to engage the child during all the time of the test; no music or audio comments are played in this phase to allow the child to focus on the answer;
2. the child chooses the answer on his/her device, depending on what he/she is able to see;
3. the server receives data and stores them in a persistent way. Doctor can analyze them in a second moment.

As already discussed in Section 8.2, our system can be used in two different environments, kindergarten and doctor's surgeries. In both environments, two different problems must be considered. The first one is related to space. Since the available space is not usually extremely high, the system should minimize the required space and encumbrance. This means that, for example, wireless connectivity should be preferred to the wired one, and even the usage of a PC should be replaced with something smaller and more portable.

The second problem is related to the infrastructure. In particular, it is not possible to rely on the hypothesis that a (wireless) Internet connectivity will be available during tests.

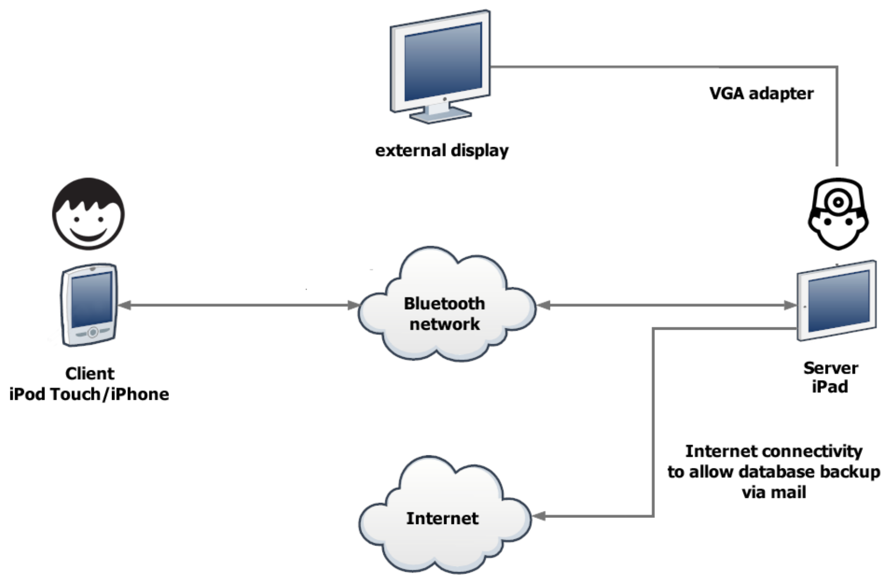


Figure 8.8: The architecture of the system, with a client-server behavior.

This is particularly true when tests are performed in a kindergarten, since not all buildings are equipped with this infrastructure. This means that the need to connect to an Internet application or to rely on online service reduces the portability of the system. Therefore, a Bluetooth® connectivity is used between client and server instead of WiFi connection.

As already mentioned, the actors that interact with the system are kindergarten teachers, oculists and children. None of them are supposed to have any special knowledge about the usage of computers, therefore during all development phase, we consider as an important target to achieve the creation of a simple and easy-to-use product with a friendly interface for final users. Devices which require minimal configurations and provide touch interfaces are preferred among traditional computers, since, according to discussion in Section 8.1, touch interfaces are best suited for children, that are naturally inclined to interact with the outside world through touch.

Mobile solutions also allow us to reduce the total amount of space required for the entire system, so the possible choices are the Apple® iPad/iPod or the Android devices. We also investigated the possibility to develop a cross-platform application, and let the users choose the best hardware setup depending on preferences, budget, etc. With this kind of approach, it is possible to develop only one single application that is further converted into two different native applications that can be used on the target devices, one in Objective-C for Apple devices or Java for Android ones.

As presented in Section 2.1, several approaches and frameworks are available for this kind of mobile application. However, as shown by our analysis both from a developer point of view (see Section 3.4) and from an energy consumption point of view (see Section

4.3), these frameworks are currently not suitable for being used to develop *PlayWithEyes*.

Giving the fact that a cross-platform development is not possible, we decided to develop an iOS application instead of an Android one for two reasons. The first one is that the Android world has a high variety of devices, of different screen dimensions, operating system, etc., and so it would not be possible to guarantee the compatibility of the system with all the devices. The other reason is that the Android devices have a more complex design, in particular, they have much more physical buttons that could become misleading for children, attracting them to touch all the buttons, thus suspending the game in some cases, instead of focus on the game.

Therefore, we use an Apple® iPad for the device (we call the server application *iPadSight*) used by the teacher and one iPod Touch or an iPhone for the child (we call the client application *iPodSight*). Moreover, an external visual display unit is also needed to execute the vision screening tests.

*PlayWithEyes* has been developed on iOS operating system, the Objective-C language and the Cocoa framework [73]. The communication is supplied by the *GameKit* framework [74], whose main focus is to wrap the well-known *Bonjour* implementation of the *Zeroconf* protocol to establish a communication channel between two devices without asking users any configuration parameters. The *UIScreen* class provided by Cocoa in the *UIKit* framework [75] enables the possibility to handle an external screen and to decide what to display on it.

Given the described architecture, we defined the following requirements:

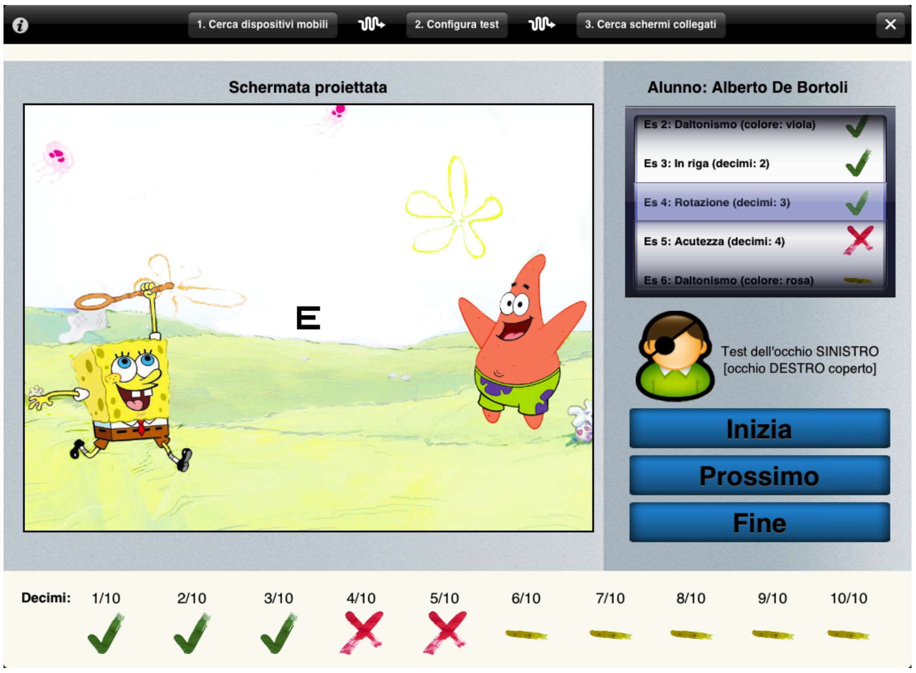
1. *iPadSight* must manage users and tests;
2. *iPadSight* and *iPodSight* must adopt the Bluetooth® wireless protocol to communicate with each other;
3. *iPadSight* must connect to an external screen and manage what to display on;
4. *iPodSight* is the interface of the serious games for kids that must be easy-to-use and funny.

The first requirement allows to create, delete and edit users, classroom and tests, but it is also necessary to keep track of the screening results in order to transmit data to oculist for diagnosis, in the case of screening in a kindergarten, and to allow statistic analysis. The Bluetooth® protocol (requirement #2) allows the system to work even in absence of a wireless network: once more, we minimize the system requirements in term of hardware components[3]. Moreover, the development of the system has shown that Apple devices allow Bluetooth® communications in a very simple way.

During eye tests, children have to respond interacting with the *iPodSight*, in accordance

[3]We must note here that an Internet connection is required only to sent data to the specialist. This operation can be done later.

to what they are able to see displayed on the screen at a given distance. At the same time, the doctor uses the *iPadSight* to watch what the system is showing to the child and to control the test progress. In this way, the doctor/teacher can control the answers already given by the child (correct/wrong) and get an idea of the remaining exercises necessary to complete the test. Figure 8.9 shows the user interface for the doctor/teacher. The system reports the child responses on the bottom section of the interface: for each size of the optotypes, the green check marks indicate a correct answer, the red “X” depicts a wrong answer, and the yellow dash means that the child does not give an answer. If more exercises use the same optotypes’ size, an “X” is displayed if the child gives, at least, one incorrect answer.



Alunno: Alberto De Bortoli

| | |
|---|---|
| Es 2: Daltonismo (colore: viola) | ✓ |
| Es 3: In riga (decimi: 2) | ✓ |
| Es 4: Rotazione (decimi: 3) | ✓ |
| Es 5: Acutezza (decimi: 4) | ✗ |
| Es 6: Daltonismo (colore: rosa) | – |

Test dell'occhio SINISTRO
[occhio DESTRO coperto]

| Decimi: | 1/10 | 2/10 | 3/10 | 4/10 | 5/10 | 6/10 | 7/10 | 8/10 | 9/10 | 10/10 |
|---|---|---|---|---|---|---|---|---|---|---|
| | ✓ | ✓ | ✓ | ✗ | ✗ | – | – | – | – | – |

Figure 8.9: User interface for the *iPadSight*

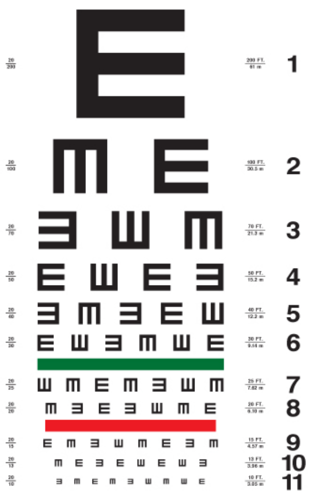
Both server and client make use of the typical gesture of touch and tablet devices. Using an iPad as a server is not a common choice, but it represents the best solution in our scenario to reduce space encumbrance. In fact, the server is used both to display the eye test on a visual display, and to interact with the doctor or teacher to select tests or create new ones (in the case of doctor), or to input children data (name, surname, age, etc.). Since the entire server resides on the iPad, its development has taken into account efficiency issues, since it should be used in a scenario of low resources in term of CPU, network bandwidth and disk space.

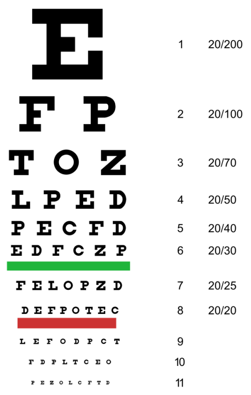
## 8.4 Resizing Optotypes

To support portability, our system is able to adapt itself to any configuration of the environment, e.g., distance of the external display, fixed monitor resolution, etc., to let the user define the best settings depending on the environment constraints and hardware settings. To allow this system customization and get the right results from our tests, it is, therefore, necessary to calculate the right size for optotypes and images shown by the monitor. In the next section, we provide the theoretical basis and the implementation steps necessary to make this customization possible.

### 8.4.1 Theoretical basis

The visual acuity is defined as the amplitude of the angle between two lines connecting the eye to two different points that are sufficiently far from each other to be able to distinguish them. The Snellen optotypes, the first optotypes used to test visual acuity, as well as the "E" chart, used for very young children, are letters that can be inserted into a 5x5 grid (see Figure 8.10c), where each grid element composes the different letters; Figures 8.10b and 8.10a provide examples of the "E" and the Snellen charts used to test visual acuity[4].


(a) Example of E chart used to test visual acuity



(b) Example of Snellen chart used to test visual acuity

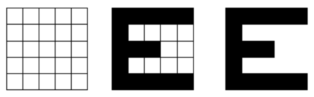

(c) Snellen optotype build with a 5x5 grid

Figure 8.10: Examples of the E chart and the Snellen optotypes commonly used by physicians to test visual acuity.

[4]Snellen optotypes which use all the letters require a 10x10 grid, but the calculation is quite the same.

According to Snellen definition, “standard vision” is the ability to recognize one of the optotypes when it subtends 5 minutes of arc, and thus, the person is able to discriminate a single stretch of size 1 minute of arc.

The **visual acuity (VA)** is defined as

$$VA = \frac{D_{effective}}{D_{1'}} \tag{8.1}$$

where $D_{effective}$ is the distance at which the test is made and $D_{1'}$ is the distance at which the smallest optotype identified subtends an angle of 1 minute of arc [34].

Considering (8.1), theoretically, it would be sufficient to use only one single letter and change the distance of the test to get different vision angles, thus keeping fixed the denominator and change only the numerator. This procedure, practically speaking, is difficult since it requires big spaces and distances to get different values of vision acuity. Moreover, the patient should continuously change the position during the test. Therefore, it is much more convenient to change the size of the letters and keep fixed the distance of the test. Nevertheless, since the formula to calculate the visual acuity with respect to the size of the optotype remains the same, it is necessary to calculate for every optotype at which distance it subtends 5 minutes of arc.

To calculate the right size of each optotype we need to define the following notation:

- $H1$: height of the optotype;
- $H2$: height of the single element of the grid;
- $D$: distance between the patient and the wall;
- $\alpha$: angle subtended by a single element of the grid;

Using this notation, the following formulas hold:

$$VA = \frac{1}{\alpha} \tag{8.2}$$

$$\tan(\alpha) = \frac{H2}{D} \tag{8.3}$$

therefore

$$\alpha = \arctan\frac{H2}{D} \tag{8.4}$$

One optotype is 5 times an element of the grid, i.e.:

$$5H2 = H1 \tag{8.5}$$

Since a single optotype subtend a 5 minute of arc, according to the Snellen definition of *Visual Acuity*, combining (8.2), (8.4) and (8.5), we obtain:

$$VA = \frac{1}{\arctan \frac{H1}{5D}} \tag{8.6}$$

In (8.6) we consider a minute of arc (see (8.1)), but since we need to work with degrees and $1' = 60°$, we get:

$$VA = \frac{1}{60 \arctan \frac{H1}{5D}} \tag{8.7}$$

and

$$H1 = 5D \tan \frac{1}{60\ VA} \tag{8.8}$$

Using (8.7), it is possible to calculate the visual acuity of an optotype of any dimension $H1$ and at a given distance $D$. In the same way, with the inverse formula (8.8) it is possible to calculate the height of the optotype to create the optotypes charts for every distance. (8.8) can be rewritten also in the following way:

$$\frac{H1}{5 \tan \frac{1}{60 VA}} = D \tag{8.9}$$

We can simplify (8.9), considering that we are calculating the "normal visual acuity", i. e, when $VA = 10/10 = 1$, and

$$\frac{1}{5 \tan \frac{1}{60}} = 687,5 \tag{8.10}$$

When $VA \neq 1$, since $\frac{1}{60}$ is a constant value, our formula can be rewritten in the following way:

$$\frac{H1}{5 \tan \frac{1}{60} \tan \frac{1}{VA}} = D \tag{8.11}$$

Since $VA \in [0.1, 1]$, $tan\frac{1}{VA} \approx \frac{1}{VA}$, from (8.11) we can approximate in the following way:

$$\frac{H1}{5\tan(\frac{1}{60})\frac{1}{VA}} = D \tag{8.12}$$

Rearranging factors in (8.12) and substituting the constant value calculated in (8.10), we obtain:

$$H1 = \frac{D}{687,5VA} \tag{8.13}$$

Using (8.13), we can calculate the size of an optotype depending on the distance used during the test ($D$) and the visual acuity that we want to measure $VA$.

Let us provide some examples. Considering a distance of 3 meters, the height of an optotype to test a normal vision acuity (10/10) is $4,36mm$, according to:

$$\frac{3000mm}{687,5} = 4,36mm \tag{8.14}$$

while at a distance of 5 meters the height is:

$$\frac{5000mm}{687,5} = 7,27mm \tag{8.15}$$

If we want to calculate the height of an optotype for a visual acuity of $3/10$ at a distance of 3 meters, what we get is:

$$\frac{3000m}{687,5 * \frac{3}{10}} = \frac{3000mm * 10}{687,5 * 3} = 14,54mm \tag{8.16}$$

### 8.4.2 Implementation

To get the final height of the optotype to show on the external display ($H$), we can use the (8.13) presented before to calculate the height of each optotype:

$$H = \frac{D}{687{,}5\,VA}$$

where $D$ is the distance of the test and $VA$ is the vision acuity. With this formula, it is possible to get the height of the image of an optotype for every vision acuity and determine the correct height in pixel to use with the screen.

Consider the following problem: we want to draw the optotype to test the vision acuity of 2/10 at a distance of 4 meters; lets calculate the final number of pixels $p$ necessary to draw the optotype knowing the size and the resolution of the display (15' and 1024$x$768).

Lets use the following notation:

- $A$: height of the display
- $R$: number of pixels of the height of the resolution
- $D$: distance of the test
- $H$: height of the optotype
- $p$: number of pixels for the height of the optotype
- $f$: number of optotypes that can be drawn together on a screen

In our case, $R$ is equal to 768, while $A$ values 270$mm$ (considering a 15' monitor).

Given (8.13), the height of the optotype is

$$H = \frac{4000mm}{687{,}5 * \frac{2}{10}} = \frac{4000mm*10}{687{,}5*2} = 29,1mm$$

The number of times it is possible to draw the optotype in height is:

$$f = \frac{A}{H} = \frac{270}{29,1} = 9,28 \tag{8.17}$$

Therefore, the number of pixels used by an optotype is:

$$p = \frac{R}{f} = \frac{768}{9,28} = 82,7 \tag{8.18}$$

then 83.

Both $R$,$A$,$D$ and $VA$ are variables that are managed via software. The setup of this values gives the possibility to use different displays of different sizes and resolution, and test distance can change, making the system adaptable to the environment settings. To have a practical test and confirmation about the calculus made by the system, before the beginning of the first test the external monitor show a ruler that, in case of right resolution detected by the system, is comparable and equal to a physical one.

## 8.5 Tests

To evaluate the system, we perform several tests involving teachers and children between 3 and 6 years old. We aim at evaluating both the engagement of the children with the game and if the system is able to test children eyes. The tests have involved 65 children, 40 females and 25 males, from kindergarten “Gianni Rodari” in Mogliano Veneto, Italy. Children are divided into 3 classes, according to age. In the first class (3 years old) there were 9 males and 10 females, in the second class (4 years old) there were 8 males and 13 females and in the third class (5 years old) there were 8 males and 17 females. As we can see, we tested our system with children of different ages to see if and how it is able to adapt, i.e. different interaction methods, and if and how children are engaged while playing the game/test.

We initially presented the game in each classroom, presenting the cartoon characters and the operators who assisted the children during the game. The tests lasted for three days, every day we tested, on average, the eyes of 21 children. We presented the system to the children as a game, but we asked them to pay attention in order to “win” the game, giving correct answers. Two children, 1 male of the second year class and a female of the first year class (out of 63, i.e., only 3%) refuse to play the game.

After the initial presentation, every child played with the game in a separated room. At the end of the game, we asked the child to evaluate the game. During the game, the children wore particular glasses with occlusion in one eye. The children played the game initially using the left eye and then the right eye. The exercises performed by the children contains all type of exercises presented before. Moreover, we proposed the touch interaction mode to the youngest children, i.e., children belonging to the first class (3-4 years old), and the drag interaction mode to the other two classes.

The test phase provided suggestions for several necessary corrections to improve our system, in particular for the *serious* part of the system. Despite these necessary corrections, discussed in Section 8.5.1, our system was able to identify visual acuity reduction in two children, as confirmed by their parents. Moreover, this test phase also highlighted how our system is able to engage children in game, and so in the visual acuity tests.

### 8.5.1 The *serious* part of the system

Each game session per child was organized with 10 exercises to test visual acuity plus other two tests for daltonism. For each test, we recorded the answer given by each child, plus the total time required to complete an exercise session. The order of the exercises was the following: acuity (1/10)[5], crowding (2/10), rotation (3/10), acuity (4/10), rotation (5/10),

[5]With (*x*/10) we want to indicate the visual acuity that we want to test, i.e. the image shown to test 1/10 is bigger that the one to test 10/10.

acuity (6/10), rotation (7/10), acuity (8/10), crowding (9/10), acuity (10/10), daltonism (path ♯1), daltonism (path ♯2). Starting from the answers provided by the children, we analyze the trend of correct answers over the different tests. This data is presented in Figure 8.11, and lead us to several interesting analysis.

First of all, the tests have shown that the averages of mistakes made by children were higher in the crowding exercises (♯2 and ♯9 in Figure 8.11), even with big optotypes, while the number of mistakes was lower for other kinds of exercise even with the same (or smaller) size of the optotypes. This means that the problem was in the exercise and not in the children sight. We found that the distance between optotypes in crowding exercises was too small. Following tests with more widely spaced optotypes do not present anymore this anomaly of the average number of mistakes.

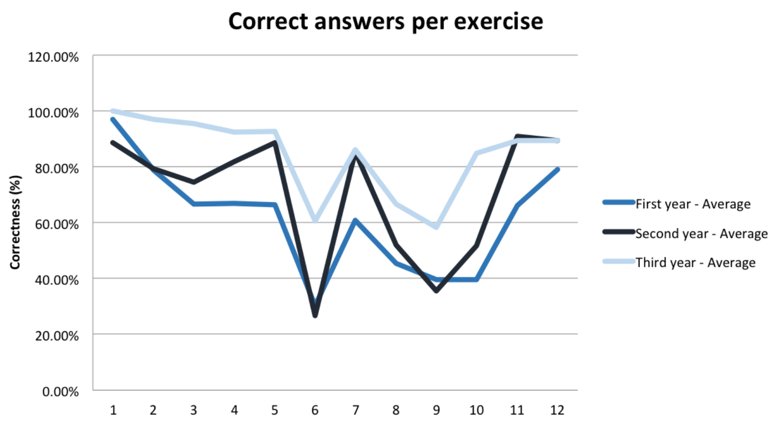


Figure 8.11: Percentage of correct answers for each exercise, according to children age.

Another interesting analysis can be made for exercise number 6. In this acuity test for 6/10, where we used Lea optotypes, we asked the child to distinguish between an apple, house, a circle and a square, and children gave more wrong answers for this exercise with respect to other exercises which ask to recognize smaller optotypes for higher value of *VA*, e. g., 8/10 or 10/10. Since the optotypes were too small, many children made confusion, mixing, in particular, the apple with the circle and the house with the square. The children do not make the same mistakes using the “E” or the Snellen chart for the same value of *VA*. Besides this, the high number of errors was also generated from the low definition of the screen, i.e. 1024*x*768 pixels. Therefore, a higher resolution monitor could decrease problems in misunderstanding. More in general, what we found is that the Lea optotypes are not suitable to assess children visual acuity equal or superior to 6/10, since when the optotypes are too small, the “E” or the Snellen charts should be preferred.

We also notice that some types of exercise have an insufficient value of contrast between the color of the optotype (black) and the color of the background (sky blue). We then modify the background color to white, increasing brightness and contrast, according to the suggestion of ophthalmologist involved in the project.

Moreover, although the Lea symbols and the tumbling "E" are extremely easy and suitable for preschool children, the tests highlight a big limitation. In both cases, the total number of possible answers are limited to four (the total number of Lea symbols, up, down, right and left for the "E"), so the probability for a child to correctly answer to a test randomly choosing between one of the four symbols is 25%, which rises to 33% after the first, wrong, answer. Therefore, we decided to introduce, in addition to the Lea symbols, also the alphabetical letters. Moving from 4 different choices to 26 (the complete English alphabet), the probability of a correct randomly chosen answer decreases to 3,85%. However, the introduction of a big number of alternatives requires to redesign the interface for the children, as discussed in Section 8.2.3.

To evaluate if our system is able to recognize visual acuity reduction in children, we asked an ophthalmologist to use the data provided by our system to evaluate visual acuity of the children and to identify particular diseases. For example, a child who shows a significant difference between visual acuity of each single eye has, potentially, an amblyopic eye. Using our data, the ophthalmologist was able to identify visual acuity reduction in four of the tested children, amblyopia was identified in two of them. We ask the parents of these children to have a test with an ophthalmologist or to provide medical records from other doctors: three of them agree, and the results obtain with our system were confirmed in two cases from external medical records (one of them confirm amblyopia) where other doctors reach the same conclusion using actual and standard visual acuity test. In the third case, the test was repeated with the white background and the child was able to answer to all the exercise. This can be clearly considered a big result, since it shows that it is possible to reach the same conclusion about visual acuity of children using our system instead of the standard exercises. The differences with respect to the standard tests are that now children are engaged (for a deeper analysis of this aspect see Section 8.5.2), tests can be longer and so provide more accurate and precise data.

As already discussed in Section 8.2.2, these tests also showed that the initial exercise for daltonism was not correct. We showed a chameleon on the monitor, and we asked the children to choose between four chameleons on the iPod screen. Unfortunately, colors used to fill the chameleon were rendered in a different way in the monitor and on the iPod screen, and two children were not able to answer since they perceived all the offered possibilities as different from the color rendered on the monitor. Using printed Ishihara plates, both the children gave correct answer. Therefore, we change the exercise for daltonism using colored Ishihara plates on the external monitor, and a black and white representation of the paths on the iPod monitor.

### 8.5.2 The *game* part of the system

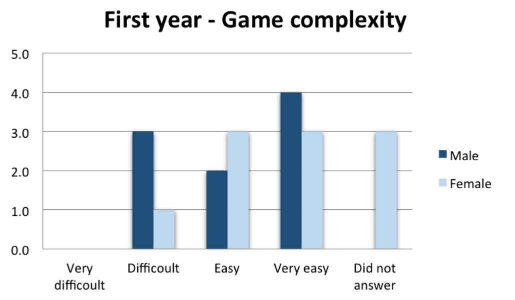
To evaluate if the children liked the game, after each test, i.e. a set of twelve exercises for visual acuity and daltonism, we asked each child to provide us an evaluation of the game and about their experience. Children of the second and third year (aged between 4 and 6) were able to use the dragging interaction method. Children of the first year instead were able to use only the touch interaction method.

The first question we provided to the children was about the complexity of the game. This evaluation is important since if the game is perceived as too much complex by a child, he/she could decide to stop to perform the exercises because the game is requiring too much strain from him/her or decide to give random answers only to quickly complete the game. The results of this question are reported in Figure 8.12, divided per year (a-c) and considering all the children (d). As we can see, the data provided follows a good distribution over all the years. In general, Figure 8.12d shows that the most preferred answers were “Easy” and “Very Easy”, meaning that the games were correctly designed and were suitable for all the ages.

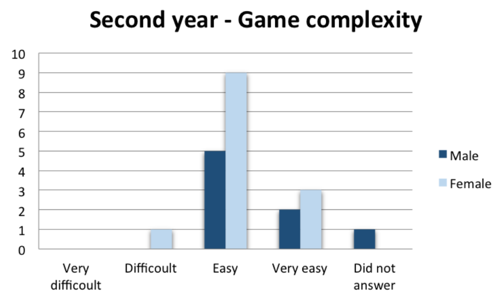
Considering the data acquired per each year, we can see how data follows an expected trend. Since the same game were used for all the children that belong to three different classrooms and ages, it is clear that the perceived difficulty of the game cannot be the same. This difference in the perceived difficulty was expected and, therefore, correct. Considering the older children (third year, 5-6 years old, see Figure 8.12c) the most commons answer provided are “Very Easy” and “Easy” (with no ”Difficult” answers). Only one child gave ”Very Difficult” as answer, but we want to pointy out that he was the first child to play with the game, and he was agitated and clearly in anxiety for this test.

Considering the first year classroom (3 years old, see Figure 8.12a), some children rated our games as “Difficult”. In the middle, rates from children of the second year contain a higher number of “Easy” answers with respect with the “Very Easy” ones (see Figure 8.12b). Again, since the games were the same for each child of all the ages, this data trend is encouraging and shows how our games were correctly designed from a difficulty point of view.

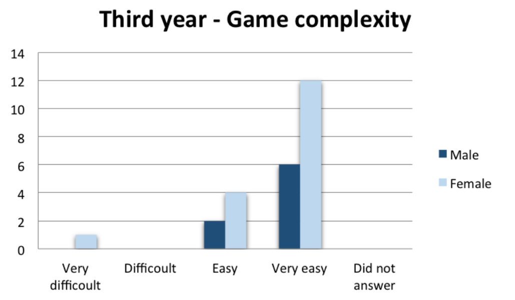
At the end of the test, we also ask children to rate the game appreciation. Each child had to choose between the answer “Very Much”, “Much”, “Neutral”, “Not so much” and “No”, using a scaled build using five different smiling (or sad) faces. This evaluation was necessary to understand if the games were sufficiently engaging and if they could be used to keep the attention of the child during the exercises, thus having more precise and useful answers. During the submission of the questionnaire, we notice that children are able to distinguish between a sad and a happy face, but their degree of attendance decrease when they have to discriminate between an happy and an happier face. For this reason, and to



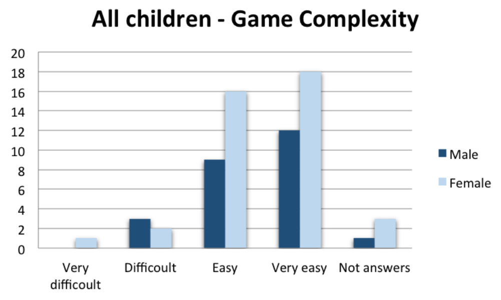
(a) Game complexity evaluation, first year children.



(b) Game complexity evaluation, second year children.



(c) Game complexity evaluation, third year children.



(d) Game complexity evaluation, all children.

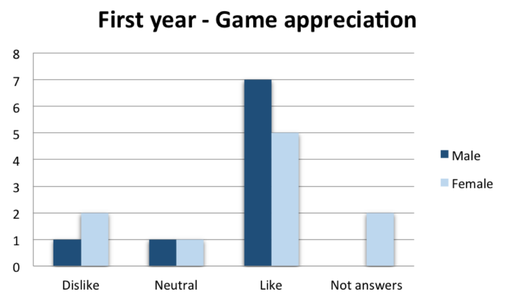
Figure 8.12: Evaluation of the complexity of the games.

present more statistically significant data, we grouped the initial 5 possibilities in 3 for clarity sake: ‘Very Much” and “Much” were grouped into “Like”, “Not so much” and “No” into “Dislike” and “Neutral” remains the same. The results of this question are provided in Figure 8.13d, and divided per classroom in Figures 8.13a, 8.13b 8.13c.

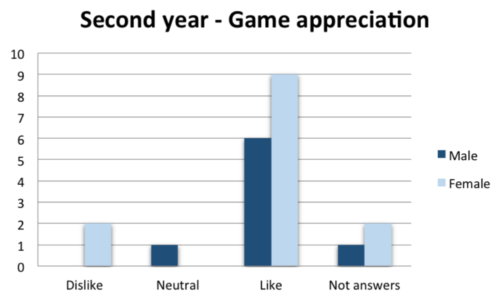
As we can see, the collected data are clearly positive and encouraging. In general, 75% (49 over 65) of the children chose “Like” as answer, while only about 7% chose the “Neutral” one and only another 7% did not like the game. About 9% decided to not answer. As we can see, the game was well accepted by the child, meaning that our system could be used to engage them and to perform an eye test while playing. In this case, we do not observe many modifications in the acquired data between the three different classrooms, meaning that the engagement was good with all the tested children.

### 8.5.3 Tests execution time

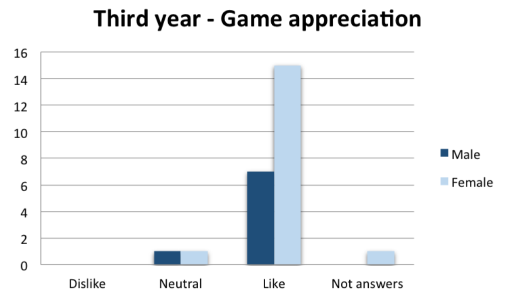
For every child we have even measured the execution time; the averages are reported in Table 8.1. Since the test begun with the left eye, the time necessary to complete the exercises with this eye was higher with respect to the right eye, meaning that children spent more time understanding how to play. Moreover, children of the first year used the touch interaction which allows quicker answers.



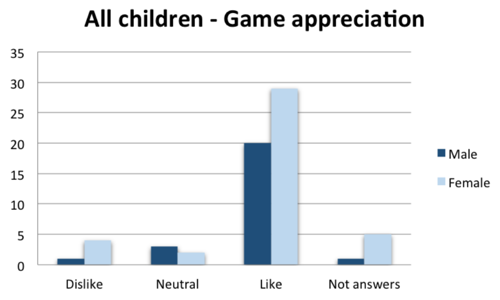
(a) Game appreciation evaluation, first year children.



(b) Game appreciation evaluation, second year children.



(c) Game appreciation evaluation, third year children.



(d) Game appreciation evaluation, all children.

Figure 8.13: Games appreciation evaluation.

On average, we need only 8 minutes to complete the assessment of each child. Usually, a normal visual acuity test requires between 15 and 20 minutes, with the child's parents that constantly help the doctor to maintain the child's attention. The low execution time has been positively evaluated even by teachers, that reported how an audiometric test made in the same school took several months, interfering with the didactic work.

| | **3 years old** | **4 years old** | **5 years old** |
|---|---|---|---|
| **First eye (Left)** | 4'27 | 5'13 | 3'42 |
| **Second eye (Right)** | 3'02 | 3'06 | 3'15 |
| **Total time** | **7'29** | **8'19** | **6'57** |

Table 8.1: Execution times for the test

V

# Conclusions and Future Works

# 9. Conclusions and future works

In this thesis we analyzed several important aspects of *ubiquitous computing* applied to healthcare and well-being, considering both technological aspects (see Part II) and their applications (see Part III and Part IV). In this chapter, for each considered issue, we will summarize the most important results we obtained with our work and our future directions.

## 9.1 Conclusions

### Cross-platform frameworks

First of all, in Chapter 2, Chapter 3 and Chapter 4 we considered the problem of market fragmentation, evaluating cross-platform frameworks for the development of a single application later deployed to different mobile operating systems. Since different cross-platform frameworks can follow different approaches in how they build the final application, we evaluated these frameworks from two different points of view. The first one, the developer point of view, considered different criteria like available APIs, documentation, etc. The second one, the user perspective, analyzed differences regarding energy consumption, depending on the considered framework.

The first analysis provided several insights. The implementation of MoSync, although promising, is stopped to old versions of all mobile platforms. jQuery provides a more widespread support to animations, but it suffer from performances degradation when used in a mobile environment. Therefore, it can be used for simple applications, but not for games. At the moment, the best framework for the development of dynamic mobile applications is Titanium, because it natively supports, with some limitations, animations and transition effects, and its performances are good and promising, even for applications more complex than our case study (a simple game with some animations, sounds and user interaction).

From the user experience point of view, our results showed how the adoption of cross-platform frameworks always implies an increase in energy consumption, even if the framework allows to deploy a real native application, as with the *Cross-Compiled Approach*.

Moreover, we compared together frameworks with the aim of finding a final rank to the current best approach to follow. Actually, the *Cross-Compiled Approach*, i.e., MoSync in our test, in particular using the Javascript implementation, if available, is the one that lowers the increase of energy required to acquire data from sensors and show them, i.e., to update the User Interface.

Unfortunately, the same approach using another programming language, i.e., MoSync using C++, has the worst performances when retrieving data from sensors (except for the compass, for which the performance are comparable to the *Hybrid Approach*). Even if, as already discussed, we can guess that this performance may improve using an appropriate library for events, these data shows that results are profoundly affected by the current state-of-art of the implementation. Therefore, the *Cross-Compiled Approach* is the more promising for what relates energy consumption, but developers should pay attention to this issue during implementation.

Our experiments also showed that is not possible to have a final and absolute rank for all the considered framework, since the set of supported features and sensors is not the same, and some frameworks are better optimized for a platform instead of the other one. This is clearly against the idea of cross-platform development. Therefore, for those applications that consider energy consumption as a big issue, at the time of writing, the native solution is better, or the framework to use should be carefully considered. Again we do not want to say that the implementation of an efficient cross-platform framework is not possible, but we solicit the developers of these frameworks to consider also the energy consumption issue since it is crucial and its current state-of-art is only at the beginning.

Finally, we highlighted another important aspect with our experiments, which is that the update of the User Interface represents the most expensive task and the primary cause of the increase in energy consumption. Moreover, the updating frequency of data retrieved by sensors and their visualization heavily affects energy consumption. Therefore, the developer has to take into consideration very carefully these two aspects when dealing with mobile applications and, in particular, with data from sensors.

### Emotion assessment using smartphones

Despite the fact that actually market fragmentation increases the problems of application development to reach the highest number of possible users, smartphones remains the best instrument for ubiquitously monitor users.

The analysis of the current state-of-the-art of cross-platform frameworks brings us to choose a native approach for the assessment of stress condition in users without using privacy-related information. Indeed, previous works on stress assessment mainly focused their attention on how to determine differences in user behavior using smartphone sensors, acquiring data about location, SMSes, calls, etc.

Differently from this strongly intrusive approach, we decided to focus our attention to human-smartphone interaction, analyzing data about the common gestures used to interact with the smartphone, e.g., ‘tap’, ‘scroll’, ‘swipe’ and text writing. About this interactions it is possible to natively and transparently, without affecting people life, extract information like pressure applied on the screen, touch size, space length of the interaction, etc. Using this data for ‘scroll’ and ‘swipe’ interactions, we show how it is possible to assess stress condition of users with an F-measure between $79\% \pm 0,02$ and $85\% \pm 0,03$ when training a separate classifier for each user, or an F-measure of about $73\% \pm 0,04$ (‘scroll’) and $81\% \pm 0,06$ (‘swipe’) with the global model.

Moreover, we investigated statistical differences of ‘text input’ features between no-stress and stress state, identifying information as the ‘number of errors made during writing’ and the ‘digits (tap interactions over a virtual keyboard) frequency’ as features with statistical significance for stress assessment. To the best of our knowledge, this is a unique approach focusing on human-smartphone interaction, and especially on ‘swipe’, ‘scroll’ and ‘text input’, proving to accurately enabling to assess stress state in smartphone users.

### *Serious games* for medical purposes

Another case study in which we use mobile technologies to help doctors and patients, is the possibility of using *serious games* into different healthcare contexts: we studied behavior change and medical diagnosis.

*ClimbTheWorld* is a *serious game* that aims at incentivizing people to take stairs instead of elevators or escalators. We evaluated our solution both considering the precision in the stairstep recognition (92%) and user experience, with a particular interest to behavior change. The obtained results showed that the game is able to engage users and to persuade them to change their behavior. In particular, we noted that the *Team vs. Team* game mode, which provides the possibility to create a challenge between two teams of different users, is capable of achieving results that seem to be maintained along time. Even in this case, we used a native approach, and we investigated energy consumption issues. We looked the best trade-off between the precision of the system and battery usage, to avoid wasting energy and ruin user experience. For this reason, in our application we introduced a new technique for data segmentation, based not on time but on data pattern, that lets us save much more energy (about 15%) with respect to the standard approach of (overlapping) sliding windows.

*PlayWithEyes*, our system for early test of visual acuity and daltonism, first of all, thanks to the use of mobile technologies, we designed a *portable* system which is able to adapt itself to any situation, in terms of availability of the Internet connection, size of the external monitor and distance of the children from the monitor. Moreover, the system provides a touch interface both for children and doctors (or teachers) since the use of this technology improves the accessibility and usability of interfaces for non-expert users. The user interface provides two interaction mode for adapting its level of difficulty to different ages of children.

We tested our system with 65 children, and the results of the tests and the following interviews with children showed great results. We were able to identify two children with visual acuity problems which were confirmed by medical records. Therefore, we can argue that the precision of the results obtained using our system are comparable to the ones obtained with standard tests. Another important result is that the use of a *serious game* can dramatically reduce the length of the test (8 minutes against 15-20 minutes) and easily engage the children thus improving the precision of their answers.

Finally, the results of the interviews with children showed that we were able to engage 97% of them, without distinction of age and sex. Therefore, interaction methods are suitable for all the ages. Moreover, the use of touch interface greatly helped usability of the system, since the perceived difficulty followed the correct trend, being easier for the older children and a bit more difficult (but not too much difficult) for the younger one.

## 9.2 Future Works

In this Section, we will highlight some future directions and works that will naturally follow the results of this thesis.

### Cross-platform frameworks

Cross-platform frameworks for mobile development are actually at the beginning of their development, meaning that some work needs to be addressed in the future.

First of all, from a developer point of view, frameworks have to increase their support for native APIs and mobile OS, i.e., increase the number of supported features/sensors, support the latest version of the OS, etc. If we aim at increasing the adoption of this approach, it is clear that it is necessary that these frameworks provide the widest range of

APIs that are natively available when using the native approach. In this way, we reduce the chance that features that the developer would need for the application would be available only using the native approach.

On the other side, as pointed out in Section 4.4, the second future work for cross-platform frameworks is related to energy consumption. In particular, since different frameworks follow a different approach on how they deploy the final application, battery usage optimization is the next step. In particular, we showed how only the *Interpreted Approach* and the *Cross-Compiled Approach* are the ones that could have an improvement, since the other two approaches, the *Web* and the *Hybrid Approach*, rely on external engines and applications and so their improvement rely only on the improvement of these external components. In particular, the *Interpreted Approach* should focus its attention on the interpretation step, to reduce the number of CPU operations that are performed, thus decreasing the final energy consumption. On the other side, the *Cross-Compiled Approach* has to focus on the compiling step, to produce a final native code that is optimized for each platform and does not waste energy, in particular when dealing with sensor events and data acquisition.

In the future, we plan to constantly monitor evolution of cross-platform frameworks, using our testbed suite, to understand if they will reach a point where they will become a good alternative to the native development, in particular considering the availability of APIs (that let developers have access to the highest number of native features) and energy consumption issues, especially when updating the User Interface is important for the developed application.

### Emotion assessment using smartphones

Since our work for stress assessment using human-smartphone interaction analysis is one of the first in this direction, it is only the starting point for different future works. The first one is to implement, instead of an application that stress users and acquires data about their interaction, a background service that constantly monitors the interaction between the user and his/her smartphone, and in real time performs three different and fundamentals steps. The first one is to automatically understand which kind of interaction is currently taking place between the user and his/her smartphone, e.g., ‘tap’, ‘scroll’, ‘swipe’, etc., since it is not possible to get this information directly from the operating system. Once we are able to understand and extract information about this interaction, we can build a classifier to assess stress conditions of users in real time without the usage of any external device. Finally, once we are able to determine if the user is stressed or not, we can use this information to apply a strategy to reduce the stress of people.

Considering the last task, two problems need to be addressed: the first one is to understand which one is the best strategy to reduce stress in people. Several solutions have been already presented in literature, but they were not able to reach good results, since in some cases the proposed solution had the drawback to stress even more the user, i.e., to extinguish a fire breathing on the microphone. The second problem instead is "timing", since it is fundamental to decide when is the best moment to propose the relaxation strategy to the user.

Another interesting approach to stress assessment is to move from the laboratory to a wild environment, where the user is free to use the smartphone as he/she prefers and in his/her natural settings. In this condition, it is possible to make a step further considering a different level of interaction.

Differently from our proposed approach that considers interactions like 'tap', 'scroll', 'swipe', etc., we refer to the possibility of considering higher level information like how many times the user turns the screen on and off, for how much time he/she uses the smartphone, the kind of applications that he/she uses (communication, sport, news, etc.), for how much time, etc. All this kind of information could be recorded in background without being too intrusive in people life and without acquiring privacy related information. With our analysis, we want to understand if, and in which measure, the way we use smartphones changes depending on if we are stressed or not, i.e., time spent on particular applications, number of times the screen is turned on and off without unlocking it, etc. Actually, we already acquired data from a 4 weeks experiment with 29 participants conducted in collaboration with the "Quality of Life" group of the University of Geneva, that developed a software for information recording in background with the usage of the Experience Sampling Method (ESM) to provide surveys to the participants during the day [138]. Preliminary results showed that particular features extracted from the acquired data have statistical difference between stress and no-stress condition of users.

Finally, another important aspect we will consider in the future is to analyze changes in the interaction not only with smartphones but even with other devices, i.e., smartwatch. Since the diffusion of these devices is extremely increasing in the last years, even how we use these devices among the days could be influenced by their mood and stress level.

### *Serious games* for medical purposes

*Serious games* used in the medical field are still under investigation. In particular, one of the biggest problems is that it is hard to provide evidence that shows that, in all the possible application areas, thanks to *serious game* there is an improvement of people' behavior.

For behavior change and lifestyle suggestion, it is clear that long user tests are necessary to understand if the *serious game* is effective in changing people' behavior. It is clear that learning (and teaching) with fun can reach better results, but objective data are required to demonstrate this statement. In the future, we plan to start a longer study with our application *ClimbTheWorld* with a wider range of tester, to be able to better understand how and in which measure our game can help people in changing their behavior. Moreover, we will investigate the possibility of using smartwatches to acquire data from the accelerometer, thus increasing the precision of our system, even combining data from the accelerometer of the smartphone and data from the smartwatch. Finally, we will investigate how to deal when the smartphone is in the trousers' pocket, since in this particular case data from the accelerometer is completely different, even if we transform it to our fixed coordinate system. In this case, it is even necessary to deal with the problem of recognizing when the smartphone is in the pocket, since simply the usage of the proximity sensor is not sufficient (data from this user's sensor is the same if the smartphone is in the bag or in the pocket).

From a doctor perspective, the usage of *serious games* for diagnosis and rehabilitation has to be carefully analyzed. Our analysis with children showed that this kind of approach is particularly promising since it can capture and maintain the attention of the patient, thus giving more time to the system to acquire data and to the doctor to get answers and formulate a diagnosis. Even if the results are encouraging, a deeper investigation and a bigger test case are required to demonstrate the effectiveness of this approach. In particular, collaboration with doctors will be necessary to have a comparison between the results provided by the system and the diagnosis made by the doctor. In the future, we plan to investigate how *serious games* can be useful in the diagnosis and the following rehabilitation of other diseases where early identification is fundamental to reduce their impact in people' life.

## 9.3 Final remarks

Summing up the whole work presented in this thesis, the conclusions we can summarize are:

- At the time of writing and of our analysis, cross-platform frameworks are still not mature for a strong adoption in mobile applications development, since actually they do not support all the native features available from smartphones. Moreover, they are still affected by energy consumption issues that negatively affect user experience;
- Native smartphone applications give the possibility to continuously monitor people behavior and mood in a non-intrusive and transparently way, letting doctors monitor their patients in their normal environment without being intrusive, violating their privacy or negatively affecting their life;
- *Serious games* developed for smartphone are a good and interesting strategy to teach people a different and healthier behavior in a non invasive and funny way, that is much more engaging and so more effective;
- *Serious games* can be used even when doctors have to deal with children to perform a diagnosis or suggest a particular rehabilitation, since they are extremely engaging

and, in conjunction with touch interfaces that work extremely good in providing an easy and immediate interaction method to them, can maintain child's attention for a longer time with respect to the standard tests and exercises provided to adults.

# Bibliography

[1] 3 sides cube. Vision Test.
http://3sidedcube.com/#work/vision-test-2.

[2] Fabio Aiolli, Matteo Ciman, Michele Donini, and Ombretta Gaggi. Climbtheworld: Real-time stairstep counting to increase physical activity. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (Mobiquitous 14)*, pages 218–227, London, UK, December 2014.

[3] Fabio Aiolli, Matteo Ciman, Michele Donini, and Ombretta Gaggi. A seriuos game to persuade people to use stairs. In *Proceedings of the 9th International Conference on Persuasive Technology (Persuasive 2014)*, 2014.

[4] Fabio Aiolli, Giovanni Da San Martino, and Alessandro Sperduti. A kernel method for the optimization of the margin distribution. In *ICANN*, pages 305–314, 2008.

[5] Al Danial. CLOC - Count Lines of Code,
http://cloc.sourceforge.net/.

[6] A. Anjum and M.U. Ilyas. Activity recognition using smartphone sensors. In *IEEE Consumer Communications and Networking Conference (CCNC 2013)*, pages 914–919, Jan 2013.

[7] Apple Inc. Health. http://www.apple.com/ios/whats-new/health/, 2015.

[8] AppZap. Vision.
https://itunes.apple.com/it/app/vision/id295144131?mt=8.

[9] Lieven Audenaeren, Veronique Celis, Vero Abeele, Luc Geurts, Jelle Husson, Pol Ghesquiere, Jan Wouters, Leen Loyez, and Ann Goeleven. Dysl-x: Design of a tablet game for early risk detection of dyslexia in preschoolers. *Games for Health*, pages 257–266, 2013.

[10] Makoto Ayabe, Junichiro Aoki, Kojiro Ishii, Kohsaku Takayama, and Hiroaki Tanaka. Pedometer accuracy during stair climbing and bench stepping exercises. *Journal of sports science & medicine*, 7(2):249, 2008.

[11] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy consumption in mobile phones: A measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '09, pages 280–293, 2009.

[12] Tom Baranowski, Richard Buday, Debbe I. Thompson, and Janice Baranowski. Playing for real: video games and stories for health-related behavior change. *American Journal of Preventive Medicine*, 34(1):74–82.e10, 2015/10/19.

[13] Jakob E. Bardram, Mads Frost, Károly Szántó, Maria Faurholt-Jepsen, Maj Vinberg, and Lars Vedel Kessing. Designing mobile health technology for bipolar disorder: A field trial of the monarca system. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 2627–2636, 2013.

[14] G. Bauer and P. Lukowicz. Can smartphones detect stress-related changes in the behaviour of individuals? In *Proceedings of IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 423–426, 2012.

[15] L. Beale, K. Field, D. Briggs, P. Picton, and H. Matthews. Mapping for wheelchair users: Route navigation in urban spaces. *Cartograp. J.*, 43(1):66–81, 2006.

[16] Luciano Bernardi, Joanna Wdowczyk-Szulc, Cinzia Valenti, Stefano Castoldi, Claudio Passino, Giammario Spadacini, and Peter Sleight. Effects of controlled breathing, mental activity and mental stress with or without verbalization on heart rate variability. *Journal of the American College of Cardiology*, pages 1462–1469, 2000.

[17] Pavol Bielik, Michal Tomlein, Peter Krátky, Štefan Mitrík, Michal Barla, and Mária Bieliková. Move2play: An innovative approach to encouraging people to be more

physically active. In *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*, IHI '12, pages 61–70, 2012.

[18] Lance Bloom, Rachel Eardley, Erik Geelhoed, Meera Manahan, and Parthasarathy Ranganathan. Investigating the relationship between battery life and user acceptance of dynamic, energy-aware interfaces on handhelds. In *Proceedings of the International Conference on Human Computer Interaction with Mobile Devices & Services*, pages 13–24, 2004.

[19] Ian Bogost. *Persuasive Games: The Expressive Power of Videogames*. The MIT Press, 2007.

[20] James Bornholt, Todd Mytkowicz, and Kathryn S. McKinley. The model is not enough-understanding energy consumption in mobile devices. Hot Chips, August 2012.

[21] Agata Brajdic and Robert Harle. Walk detection and step counting on unconstrained smartphones. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '13)*, pages 225–234, 2013.

[22] A. Bujari, B. Licar, and C. E. Palazzi. Road crossing recognition through smartphone's accelerometer. In *Proceedings of the IFIP/IEEE Wireless Days 2011*, 2011.

[23] J. Burrell, T. Brooke, and R. Beckwith. Vineyard computing: sensor networks in agricultural production. *IEEE Pervasive Computing*, 3(1):38–45, 2004.

[24] Canela Software. 20/20 Vision. http://canelasoftware.com.

[25] Aaron Carroll and Gernot Heiser. An analysis of power consumption in a smartphone. In *USENIX annual technical conference*, pages 271–285, 2010.

[26] Andre Charland and Brian Leroux. Mobile application development: web vs. native. *Communications of ACM*, 54(5):49–53, May 2011.

[27] Xiang Chen, Yiran Chen, Zhan Ma, and Felix C. A. Fernandes. How is energy consumed in smartphone display applications? In *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, HotMobile '13, 2013.

[28] AdrianDavid Cheok, KokHwee Goh, Wei Liu, Farzam Farbiz, SiewWan Fong, SzeLee Teo, Yu Li, and Xubo Yang. Human pacman: a mobile, wide-area entertainment system based on physical, social, and ubiquitous computing. *Personal and Ubiquitous Computing*, 8(2):71–81, 2004.

[29] Luca Chittaro. Passengers' safety in aircraft evacuations: Employing serious games to educate and persuade. In *Persuasive Technology. Design for Health and Safety*, pages 215–226. Springer Berlin Heidelberg, 2012.

[30] M. Ciman and O. Gaggi. Evaluating impact of cross-platform frameworks in energy consumption of mobile applications. In *Proceedings of the 10th International Conference on Web Information Systems and Technologies (WEBIST 2014)*, pages 423–431, 2014.

[31] Matteo Ciman, Ombretta Gaggi, and Nicola Gonzo. Cross-platform mobile development: a study on apps with animations. In *Symposium on Applied Computing, SAC 2014, Gyeongju, Republic of Korea - March 24 - 28, 2014*, pages 757–759, 2014.

[32] Matteo Ciman, Ombretta Gaggi, Laura Nota, Luisa Pinello, Nicola Riparelli, and Teresa Maria Sgaramella. HelpMe!: A serious game for rehabilitation of children affected by CVI. In *Proceedings of the 9th International Conference on Web Information Systems and Technologies (WEBIST), Aachen, Germany, 8-10 May, 2013*, pages 257–262, 2013.

[33] S. Cohen, R.C. Kessler, and L.U. Gordon. *Measuring Stress: A Guide for Health and Social Scientists*. Guide for Health and Social Scientists. 1997.

[34] August Colenbrander. The historical evolution of visual acuity measurement. *Visual Impairment Research*, 10(2-3):57–66, 2008.

[35] Sunny Consolvo, David W. McDonald, Tammy Toscos, Mike Y. Chen, Jon Froehlich, Beverly Harrison, Predrag Klasnja, Anthony LaMarca, Louis LeGrand, Ryan Libby, Ian Smith, and James A. Landay. Activity sensing in the wild: A field trial of ubifit garden. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1797–1806, 2008.

[36] Corusen LLC. Accupedo. http://www.accupedo.com/, 2015.

[37] Mihaly Csikszentmihalyi. *Beyond boredom and anxiety*. Jossey-Bass, 2000.

[38] I. Dalmasso, S.K. Datta, C. Bonnet, and N. Nikaein. Survey, comparison and evaluation of cross platform mobile application development tools. In *Proceedings of the 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 323–328, July 2013.

[39] EliseS. Dan-Glauser and KlausR. Scherer. The geneva affective picture database (gaped): a new 730-picture database focusing on valence and normative significance. *Behavior Research Methods*, 2011.

[40] D. Deponti, D. Maggiorini, and C. E. Palazzi. DroidGlove: An android-based application for wrist rehabilitation. In *Proceedings of the IEEE International Conference on Ultramodern Telecommunications and Workshops (ICUMT 2009)*, 2009.

[41] D. Deponti, D. Maggiorini, and C. E. Palazzi. Smartphone's physiatric serious game. In *Proceedings of the IEEE International Conference on Serious Games and Applications for Health (SeGAH 2011)*, 2011.

[42] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From game design elements to gamefulness: Defining "gamification". In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, MindTrek '11, pages 9–15, 2011.

[43] I. Di Loreto and A. Gouaich. Mixed reality serious games: The therapist perspective. In *Proceedings of the IEEE International Conference on on Serious Games and Applicationsm for Health (SeGAH 2011)*, pages 1–10, Braga, Portugal, 2011.

[44] Sally S Dickerson and Margaret E Kemeny. Acute stressors and cortisol responses: a theoretical integration and synthesis of laboratory research. *Psychological bulletin*, pages 355–391, 2004.

[45] Thang M Do, Seng W Loke, and Fei Liu. Healthylife: An activity recognition system with smartphone using logic-based stream reasoning. pages 188–199, 2013.

[46] Dok LLC. EyeChart HD.
https://itunes.apple.com/it/app/eye-chart-hd/id382019572?mt=8.

[47] McGraw-Hill Education. Government in action. Available online:http://www.mhpractice.com/, 2012.

[48] G. Esteban, C. Fernandez, V. Matellan, and J. Gonzalo. Computer surgery 3d simulations for a new teaching-learning model. In *Proceedings of the IEEE International Conference on on Serious Games and Applicationsm for Health (SeGAH 2011)*, pages 1–4, Braga, Portugal, 2011.

[49] A. Facoetti, S. Franceschini, O. Gaggi, G. Galiazzo, S. Gori, C.E. Palazzi, and M. Ruffino. Multiplatform games for dyslexia identification in preschoolers. In *Proceedings of the IEEE 11th Consumer Communications and Networking Conference (CCNC)*, pages 1152–1153, 2014.

[50] S. Ferretti, M. Furini, C. E. Palazzi, M. Roccetti, and P. Salomoni. WWW recycling for a better world. *Communications of the ACM*, 53(4):139–143, 2010.

[51] Fitbit Inc. Fitbit. http://www.fitbit.com/, 2015.

[52] Jason Flinn and M. Satyanarayanan. Powerscope: a tool for profiling the energy usage of mobile applications. In *Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '99)*, Feb 1999.

[53] B. J. Fogg. A behavior model for persuasive design. In *Proceedings of the 4th International Conference on Persuasive Technology*, Persuasive '09, pages 1–7, 2009.

[54] B. J. Fogg and Jason Hreha. Behavior wizard: A method for matching target behaviors with solutions. In *Proceedings of the 5th International Conference on Persuasive Technology*, Persuasive '10, pages 117–131, 2010.

[55] BJ Fogg. The behavior grid: 35 ways behavior can change. In *Proceedings of the 4th International Conference on Persuasive Technology*, Persuasive '09, pages 1–5, 2009.

[56] Foog, B. J. BJ Fogg's Behavior Model. http://www.behaviormodel.org/, 2015.

[57] Alain Forget, Sonia Chiasson, P.C. van Oorschot, and Robert Biddle. Persuasion for stronger passwords: Motivation and pilot study. In *Persuasive Technology*, pages 140–150. 2008.

[58] C. Forlines, D. Wigdor, C. Shen, and R. Balakrishnan. Direct-touch vs. mouse input for tabletop displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 647–656, 2007.

[59] C. Forlines, D. Wigdor, C. Shen, and R. Balakrishnan. Direct-Touch vs. Mouse Input for Tabletop Displays. In *Proceedings of ACM CHI 2007 Conference on Human Factors in Computing Systems*, pages 647–656, May 2007.

[60] R. Friedman, A. Kogan, and Y. Krivolapov. On power and throughput tradeoffs of wifi and bluetooth in smartphones. *IEEE Transactions on Mobile Computing*, 12(7):1363–1376, 2013.

[61] O. Gaggi and M. Ciman. The use of games to help children eyes testing. *Multimedia Tools and Applications*, pages 1–26, 2015.

[62] O. Gaggi, G. Galiazzo, C. E. Palazzi, A. Facoetti, and S. Franceschini. A serious game for predicting the risk of developmental dyslexia in pre-readers children. In *Proceedings of the 21st International Conference on Computer Communications and Networking (ICCCN 2012)*, 2012.

[63] Ombretta Gaggi, Claudio Enrico Palazzi, Matteo Ciman, Giorgia Galiazzo, Sandro Franceschini, Milena Ruffino, Simone Gori, and Andrea Facoetti. Serious games for early identification of developmental dyslexia. *ACM Computers in Entertainment*, 2014.

[64] Yuan Gao, Nadia Bianchi-Berthouze, and Hongying Meng. What does touch tell us about emotions in touchscreen-based gameplay? *ACM Transactions on Computer-Human Interaction*, pages 1–30, 2012.

[65] Google, Inc. Google APIs for Android, Available at: https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognitionApi.

[66] Gridvision Engineering GmbH. Gleeo Time Tracker, http://gleeo.com.

[67] S.D.W. Gunawardhane, P.M. De Silva, D.S.B. Kulathunga, and S.M.K.D. Arunatileka. Non invasive human stress detection using key stroke dynamics and pattern variations. In *Proceedings of the International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 240–247, Dec 2013.

[68] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 2009.

[69] Shuai Hao, Ding Li, William G. J. Halfond, and Ramesh Govindan. Estimating mobile application energy consumption using program analysis. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 92–101, 2013.

[70] Robert T Hays, John W Jacobs, Carolyn Prince, and Eduardo Salas. Flight simulator training effectiveness: A meta-analysis. *Military Psychology*, 4(2):63, 1992.

[71] Henning Heitkötter, S. Hanschke, and Tim A. Majchrzak. Comparing cross-platform development approaches for mobile applications. In *Proceedings of the 8th International Conference of Web Information Systems and Technologies*, WEBIST '12, pages 299–311, 2012.

[72] Javier Hernandez, Pablo Paredes, Asta Roseway, and Mary Czerwinski. Under pressure: Sensing stress of computer users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 51–60, 2014.

[73] Apple Inc. Cocoa framework.
https://developer.apple.com/technologies/mac/cocoa.html.

[74] Apple Inc. Gamekit framework.
https://developer.apple.com/library/ios/documentation/gamekit/reference/GameKit_Collection/index.html.

[75] Apple Inc. Uikit framework.
https://developer.apple.com/library/ios/documentation/uikit/reference/uikit_framework/index.html.

[76] Google Inc. Panoramio. Available online: http://www.panoramio.com/, 2006.

[77] Google Inc. Google image labeler. Available online: http://images.google.com/imagelabeler/, 2007.

[78] Intellicore. Acuity.
https://itunes.apple.com/us/app/acuity/id307916140?mt=8&ign-mpt=uo%3D4.

[79] Jawbone. Jawbone Up. https://jawbone.com/, 2015.

[80] Maurits Kaptein and Dean Eckles. Heterogeneity in the effects of online persuasion. *Journal of Interactive Marketing*, pages 176 – 188, 2012.

[81] P. Kasemsuppakorn and H. Karimi. Personalized routing for wheelchair navigation. *J. of Location Based Services*, 3(1):24–54, 2009.

[82] P. M. Kato. Video games in health care: closing the gap. *Review of General Psychology*, 14(2):113–121, 2010.

[83] Pamela M. Kato, Steve W. Cole, Andrew S. Bradlyn, and Brad H. Pollock. A video game improves behavioral outcomes in adolescents and young adults with cancer: A randomized trial. *Pediatrics*, 122(2):e305–e317, 2008.

[84] Saskia M. Kelders, Robin N. Kok, Hans C. Ossebaard, and Julia EWC Van Gemert-Pijnen. Persuasive system design does matter: A systematic review of adherence to web-based interventions. *Communications of the Association for Information Systems*, 2012.

[85] Jesper Kjeldskov and Mikael B. Skov. Exploring context-awareness for ubiquitous computing in the healthcare domain. *Personal Ubiquitous Computing*, 11(7):549–562, 2007.

[86] Konrad Feiler. iC Pro: The EyeTest. https://itunes.apple.com/us/app/ic-pro-the-eyetest/id405950873?mt=8.

[87] J. Krumm. Ubiquitous advertising: The killer application for the 21st century. *IEEE Pervasive Computing*, 10(1):66–73, 2011.

[88] kyberVision Consulting R&D. FastAcuity Lite. http://www.kybervision.com/iphone/fastacuity/index.php.

[89] Nicholas D. Lane, Mashfiqui Mohammod, Mu Lin, Xiaochao Yang, Hong Lu, Shahid Ali, Afsaneh Doryab, Ethan Berke, Tanzeem Choudhury, and Andrew T. Campbell. Bewell: A smartphone application to monitor, model and promote wellbeing. In *Proceedings of the International ICST Conference on Pervasive Computing Technologies for Healthcare*, pages 23–26, 2011.

[90] Sitwat Langrial, Tuomas Lehto, Harri Oinas-Kukkonen, Marja Harjumaa, and Pasi Karppinen. Native mobile applications for personal well-being: A persuasive systems design evaluation.

[91] Reed Larson and Csikszentmihalyi. The experience sampling method. *New Directions for Methodology of Social & Behavioral Science*, 1983.

[92] Lea Test Ltd. Lea Vision Test System. http://www.lea-test.fi/index.html.

[93] Hosub Lee, Young Sang Choi, Sunjae Lee, and I.P. Park. Towards unobtrusive emotion recognition for affective social communication. In *Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC)*, pages 260–264, Jan 2012.

[94] James R. Lewis. Ibm computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use. *International Journal on Human-Computer Interaction*, 7(1):57–78, January 1995.

[95] Rensis Likert. A technique for the measurements of attitudes. *Archives of psychology*, page 55, 1932.

[96] Mu Lin, Nicholas D. Lane, Mashfiqui Mohammod, Xiaochao Yang, Hong Lu, Giuseppe Cardone, Shahid Ali, Afsaneh Doryab, Ethan Berke, Andrew T. Campbell, and Tanzeem Choudhury. Bewell+: Multi-dimensional wellbeing monitoring with community-guided user feedback and energy optimization. In *Proceedings of the Conference on Wireless Health*, WH '12, pages 10:1–10:8, 2012.

[97] M. Ma and K. Bechkoum. Serious games for movement therapy after stroke. In *Proceedings of IEEE Conference On Systems, Man and Cybernetics*, pages 1872–1877, Singapore, 2008.

[98] Tim A. Majchrzak and Henning Heitkötter. Development of mobile applications in regional companies: Status quo and best practices. In *Proceedings of the 8th International Conference of Web Information Systems and Technologies*, WEBIST '13, pages 335–346, 2013.

[99] Misfit. Flash. http://misfit.com/products/flash, 2015.

[100] Radhika Mittal, Aman Kansal, and Ranveer Chandra. Empowering developers to estimate app energy consumption. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, Mobicom '12, pages 317–328, 2012.

[101] Radhika Mittal, Aman Kansal, and Ranveer Chandra. Empowering developers to estimate app energy consumption. In *Proceedings of the 18th annual International Conference on Mobile Computing and Networking (MobiCom '12)*, pages 317–328, 2012.

[102] David Mizell. Using gravity to estimate accelerometer orientation. In *Proceedings of the 7th IEEE International Symposium on Wearable Computers (ISWC'03)*, page 252, 2003.

[103] Patel MS, Asch DA, and Volpp KG. Wearable devices as facilitators, not drivers, of health behavior change. *JAMA*, 313(5):459–460, 2015.

[104] Harri Oinas-Kukkonen. Behavior change support systems: A research model and agenda. In *Proceedings of the 5th International Conference on Persuasive Technology*, PERSUASIVE'10, pages 4–14, 2010.

[105] Harri Oinas-Kukkonen and Marja Harjumaa. Persuasive systems design: Key issues, process model, and system features. *Communications of the Association for Information Systems*, 2009.

[106] C. E. Palazzi, M. Brunati, and M. Roccetti. Path 2.0: A participatory system for the generation of accessible routes. In *Proceedings of the IEEE Conference on Multimedia and Expo (ICME 2010)*, 2010.

[107] C. E. Palazzi, G. Marfia, and M. Roccetti. Combining web squared and serious games for crossroad accessibility. In *Proceedings of the IEEE International Conference on Serious Games and Applications for Health (SeGAH 2011)*, 2011.

[108] C. E. Palazzi, M. Roccetti, and G. Marfia. Realizing the unexpected potential of games on serious challenges. *ACM Computers in Entertainment*, 8(4), 2010.

[109] M. Palmieri, I. Singh, and A. Cicchetti. Comparison of cross-platform mobile development tools. In *Proceedings of the 16th International Conference on Intelligence in Next Generation Networks*, ICIN '12, pages 179–186, 2012.

[110] P. Paredes, D. Sun, and J. Canny. Sensor-less sensing for affective computing and stress management technology. In *Proceedings of 7th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*, pages 459–463, 2013.

[111] Abhinav Pathak, Y. Charlie Hu, and Ming Zhang. Where is the energy spent inside my app?: Fine grained energy accounting on smartphones with eprof. In *Proceedings of the 7th ACM European Conference on Computer Systems (EuroSys '12)*, pages 29–42, 2012.

[112] A. Perkins. Saving money by reducing stress. *Harvard Business Review*, 1994.

[113] G.P. Perrucci, F.H.P. Fitzek, and J. Widmer. Survey on energy consumption entities on the smartphone platform. In *Proceedings of the 73rd IEEE Conference of Vehicular Technology (VTC Spring)*, pages 1–6, May 2011.

[114] R. W. Picard. *Affective Computing*. Cambridge: MIT press, 1995.

[115] Susanna Pirttikangas, Kaori Fujinami, and Tatsuo Nakajima. Feature selection and activity recognition from wearable sensors. *Lecture Notes in Computer Science*, 4239:516–527, 2006.

[116] F. Pittarello and R. Stecca. Querying and navigating a database of images with the magical objects of the wizard zurlino. In *Proceedings of the 9th International Conference on Interaction Design and Children*, IDC '10, pages 250–253, 2010.

[117] A Popleteev, V. Osmani, R. Maimone, S. Gabrielli, and O. Mayora. Mobile stress treatment: The interstress approach. In *Proceedings 6th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*, pages 197–198, May 2012.

[118] Precision Vision. PVVAT.
http://www.precision-vision.com/index.cfm/feature/20/pvvat.cfm.

[119] Ahmad Rahmati, Angela Qian, and Lin Zhong. Understanding human-battery interaction on mobile phones. In *Proceedings of the 9th International Conference*

*on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '07, pages 265–272, 2007.

[120] R. Raj and S.B. Tolety. A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach. In *Proceedings of the Annual IEEE India Conference*, INDICON '12, pages 625–629, 2012.

[121] M. Roccetti, A. Casari, and G. Marfia. Inside chronic autoimmune disease communities: A social networks perspective to crohn's patient behavior and medical information. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2015)*, 2015.

[122] M. Roccetti, G. Marfia, and Palazzi C. E. Entertainment beyond divertissment: Using computer games for city road accessibility. *ACM Computers in Entertainment*, 9(2), 2011.

[123] Lizawati Salahuddin and Desok Kim. Detection of acute stress by heart rate variability using a prototype mobile ecg sensor. In *Proceedings of the 2006 International Conference on Hybrid Information Technology - Volume 02*, ICHIT '06, pages 453–459, 2006.

[124] P. Salomoni, C. Prandi, M. Roccetti, V. Nisi, and N. J. Nunes. Crowdsourcing urban accessibility: Some preliminary experiences with results. In *Proceedings of the ACM 2015 CHItaly, 11th Edition of the Biannual Conference of the Italian SIGCHI Chapter, (CHItaly 2015)*, 2015.

[125] Akane Sano and Rosalind W. Picard. Stress recognition using wearable sensors and mobile phones. In *Proceedings of ACII*, pages 671–676, 2013.

[126] M. Shoaib, H. Scholten, and P.J.M. Havinga. Towards physical activity recognition using smartphone sensors. In *Proceedings of the 10th IEEE International Conference on Ubiquitous Intelligence and Computing and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*, pages 80–87, Dec 2013.

[127] Pekka Siirtola and Juha Röning. Recognizing human activities user-independently on smartphones based on accelerometer data. *International Journal of Interactive Multimedia and Artificial Intelligence*, 1(5):38–45, 06/2012 2012.

[128] E. G. Steenbeek-Planting, M. Boot, J. C. de Boer, M. Van de Ven, N. M. Swart, and D. van der Hout. Evidence-based psycholinguistic principles to remediate reading problems applied in the playful app letterprins: A perspective of quality of healthcare on learning to read. In *Games for Health*, pages 281–291. 2013.

[129] F. Sufi, Q. Fang, and I. Cosic. Ecg r-r peak detection on mobile phones. In *Proceedings of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2007. EMBS 2007*, pages 3697–3700, 2007.

[130] David Sun, Pablo Paredes, and John Canny. Moustress: Detecting stress from mouse motion. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 61–70, 2014.

[131] Narendran Thiagarajan, Gaurav Aggarwal, Angela Nicoara, Dan Boneh, and Jatinder Pal Singh. Who killed my battery?: Analyzing mobile browser energy consumption. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 41–50, 2012.

[132] Under Armour ConnectedFitness. Endomondo. https://www.endomondo.com/, 2015.

[133] IDC Corporate USA. Smartphone OS Market Share, http://www.idc.com/prodserv/smartphone-os-market-share.jsp, 2014.

[134] L. Van den Audenaeren, V. Celis, V. Vanden Abeele, L. Geurts, J. Husson, P. Ghesquière, J. Wouters, L. Loyez, and A. Goeleven. DYSL-X: Design of a tablet game for early risk detection of dyslexia in preschoolers. In *Games for Health*, pages 257–266. Springer Fachmedien Wiesbaden, 2013.

[135] Rogier M. van Eijk. Requirements for relaxation coaching - a formalization of the fogg behavior model. In *Proceedings of ICT for Ageing Well and e-Health*, pages 31–36, Lisboa, Portugal, 2015.

[136] Volkswagen. The Fun Theory. http.//www.thefuntheory.com.

[137] Volkswagen. The Piano stairs,
http://www.youtube.com/watch?v=2lXh2n0aPyw.

[138] Katarzyna Wac, Mattia Gustarini, Jerome Marchanoff, Marios A Fanourakis, Christiana Tsiourti, Matteo Ciman, Jody Hausmann, and Gerardo Pinar Lorente. mqol: Experiences of the 'mobile communications and computing for quality of life' living lab. In *Proceedings of the 17th IEEE International Conference on e-Health Networking, Applications and Services (HealthCom): The Living Lab approach for successful design of services and systems in eHealth (The Living Lab)*, Boston, USA, oct 2015.

[139] Mark Weiser. The computer for the 21st century. *SIGMOBILE Mobile Computing and Communications Reviews*, 3(3):3–11, 1999.

[140] Mark Weiser and JohnSeely Brown. The coming age of calm technology. In *Beyond Calculation*, pages 75–85. 1997.

[141] Mark Weiser, Rich Gold, and John Seely Brown. The origins of ubiquitous computing research at parc in the late 1980s. *IBM systems journal*, 38(4):693–696, 1999.

[142] Kevin Werbach. (re)defining gamification: A process approach. In *Persuasive Technology*, pages 266–272. 2014.

[143] Kevin Werbach and Dan Hunter. *For the win: How game thinking can revolutionize your business*. Wharton Digital Press, 2012.

[144] World Health Organization. Physical Inactivity: A Global Public Health Problemhttp://www.who.int/dietphysicalactivity/factsheet_inactivity/en/.

[145] PawełWoźniak, Kristina Knaving, Staffan Björk, and Morten Fjeld. Untangling running: Designing for real-life runner experiences. *ACM Interactions*, 22:40–43, February 2015.

[146] Wanmin Wu, Sanjoy Dasgupta, Ernesto E Ramirez, Carlyn Peterson, and Gregory J Norman. Classification accuracies of physical activities using smartphone motion sensors. *Journal of medical Internet research*, 14(5):e130, 2012.

[147] M. Zapusek, S. Cerar, and J. Rugelj. Serious computer games as instructional technology. In *Proceedings of the 34th International Convention MIPRO*, pages 1056–1058, 2011.

[148] Gabe Zichermann and Christopher Cunningham. *Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps*. O'Reilly Media, Inc., 2011.