SocialSensing - Social sensing application for Smartphone

Roberto José Gomes Silva

Thesis to obtain the Master of Science Degree in
Communications Networks Engineering

Examination Committee
Chairperson: Prof. Paulo Jorge Pires Ferreira
Supervisor: Prof. Rui Manuel Rodrigues Rocha
Member of the Committee: Prof. Ricardo Jorge Feliciano Lopes Pereira

November 2013
Acknowledgments

First and foremost, I would like to thank my family for the constant support they gave to me even though I could not spend much time with them because of my academic life. I enjoy the brief moments of their company. I would also like to show my gratitude to my supervisor Prof. Rui Rocha, a respectable, knowledgeable and strict scholar, who has provided me with valuable guidance in every stage of the writing of this thesis.

I shall extend my thanks to my wise and patient friend, João Fernandes. He has been acting like an older brother since I first arrived in Lisbon. Also, I would like to thank my friend Vitor Mansur for offering me valuable advice. And to my girlfriend, Yao Yien who has been supporting me with her understanding and love throughout all this work.

Last but not least, I’d like to thank my friends João Prioste, João Rosa, João Cardoso, Gonçalo Pereira, Nadia Gonçalves and Michael Martins for their encouragement and support.

To each and every one of you - thank you.
Resumo

Os telemóveis e as redes sociais tornaram-se parte do quotidiano das pessoas. Estes telemóveis contêm um conjunto de sensores que fornecem informações úteis (localização, inclinação, etc), permitindo o surgimento de aplicações de monitorização pessoais e de grupo. No entanto, as aplicações existentes não tem em conta que ciclistas andam em grupo. Estas actividades em grupo introduzem uma oportunidade para partilha de recursos entre os elementos do grupo. Esta dissertação tem por objectivo desenvolver uma plataforma que forneça vários benefícios para os ciclistas. Usando uma aplicação móvel os ciclistas conseguiram medir dados do seu desempenho e o ambiente do seu percurso. Quando o ciclista anda em grupo, a aplicação partilhará recursos com o resto da equipa. Esta também fornece informação da posição relativa do utilizador em relação ao resto da equipa. Caso algum membro do grupo tenha posição absoluta, será usado como ancora para transladar a posição do resto da equipa para absoluta. Caso algum dos ciclistas perca o rasto da equipa, a aplicação é capaz de fornecer em tempo real a localização dos elementos da equipa. O utilizador também pode usar uma aplicação web para analisar o seu histórico, ou planear em avanço um percurso. A avaliação do sistema foi baseada na veracidade dos dados medidos, no consumo de energia e opinião dos utilizadores.

Palavras-chave: Sensorização social, PhoneSensing, Ciclismo, Aplicação Móvel, Aplicação Web, Android, Serviços Web, P2P.
Abstract

The smartphones and the social networks become part of everyday life. Modern smartphones having a set of embedded sensors inside, are able to provide useful information such as location, slope, etc, which are enabling the emergence of personal and group sensing applications. However, most of the existing sensing platforms for cyclists only gather data in a personal level. Quiet often do cyclist perform their activities in group. These groups activities present an opportunity to explore resources sharing among the group elements. This dissertation aims to develop a platform that provides several benefits to the cyclist users on a personal, social and public level. Using a mobile phone application, cyclists will be able to gather fitness and environmental data about their rides. When they cycle in group, the application will share resources with the rest of the group. It will also provide position estimation providing the localization of every cyclist relatively to others in the team. In case absolute location is available in the group, they are used as anchors to translate the relative location to absolute. In addition, if one of the cyclists loses track of the rest of the team, the mobile application provides information about the location of his team mates in real time. Furthermore the user can use a web-based application to analyse his previous rides, or plan in advance his rides. We evaluated the system based on the device sensors/algorithm accuracy, power consume and feedback from cyclists.

Keywords: Social Sensing, PhoneSensing, Bicycling, Mobile application, Web application, Android, Web Services, P2P.
Contents

Acknowledgments .................................................. iii
Resumo ................................................................. v
Abstract ............................................................... vii
List of Tables ......................................................... xiii
List of Figures ......................................................... xvi
Nomenclature ........................................................ xvii
Glossary ................................................................. 1

1 Introduction ......................................................... 1
  1.1 Motivation and Goals ........................................... 2
  1.2 Challenges ..................................................... 2
  1.3 Contributions ................................................ 3
  1.4 Organization .................................................. 3

2 User requirements and typical solutions ......................... 5
  2.1 Type of Service ............................................... 5
  2.2 Architecture .................................................. 6
  2.3 Mobile Sensing Networks ...................................... 7
  2.4 Social Networks APIs .......................................... 9
  2.5 Mobile Application Interaction ................................. 11
    2.5.1 User interface ........................................... 11
    2.5.2 User profile ............................................. 11
  2.6 Privacy and Security .......................................... 11
  2.7 Functionality ................................................ 12
  2.8 Related work ................................................ 14
    2.8.1 Bikely .................................................... 14
    2.8.2 Cyclopath ............................................... 14
    2.8.3 Bikemap .................................................. 14
    2.8.4 Nike + sportwatch GPS .................................. 15
    2.8.5 Endomondo ............................................... 15
    2.8.6 Biketastic ............................................... 16
List of Tables

2.1 Comparison between the general approaches employed by each cycling application. . . . 18

4.1 Average values measured during the accuracy tests . . . . . . . . . . . . . . . . . . . . . 48

5.1 Average values measured during the tests and their respective errors. . . . . . . . . 68
List of Figures

2.1 Validation of "track the team members on a map" .......................... 13
2.2 Validation of "Route's noise and steepness in advance" ......................... 13

3.1 Social Sensing system overview ................................................. 25
3.2 General view of the client-server architecture ................................. 26
3.3 Site Map ................................................................. 26
3.4 General view of the PhoneSensing framework .................................. 28
3.5 General view of the SSMA .................................................. 29
3.6 Team formation process ....................................................... 33
3.7 Measure distance algorithm ..................................................... 34
3.8 Reply from remote device ...................................................... 35

4.1 Diagram representing the registry process ....................................... 39
4.2 Map showing the ride of a user .................................................. 41
4.3 User requests a team from the back-end ......................................... 51
4.4 RSSI measure before and after correction ....................................... 55
4.5 Translate the relative location to latitude and longitude coordinates ......... 56
4.6 Authentication screens ......................................................... 59

5.1 Data gathered along the route: a) Noise data; b) Steepness data .............. 62
5.2 Noise data gathered along the route. a) Silva, b) Mansur and c) Yao .......... 63
5.3 Steepness data gathered along the route. From the left to the right: Silva, Mansur and Yao 64
5.4 CPU consumption ............................................................ 65
5.5 Battery consumption ............................................................ 65
5.6 Time in seconds to form a team ............................................... 66
5.7 Time in seconds to form a team using distributed approach ................. 66
5.8 Power consumption for centralized and distributed approach. The distributed approach is divided in two: Team leader, team member .................................................. 67
5.9 CPU and battery consumptions .................................................. 68
5.10 Browser's usage ............................................................... 69
5.11 Average time spent for each task ............................................. 70
5.12 Average number of mistakes for each task .................................... 70
Acronyms

AOS  Android Operating System
API  Application Programming Interface
CSS  Cascading Style Sheets
GPS  Global Positioning System
HTTP  Hypertext Transfer Protocol
HTML  HyperText Markup Language
JSON  JavaScript Object Notation
KLV  Key-lenght-value
MDS  Multidimensional scaling
MS  Mobile Sensor
PCM  Pulse-code modulation
PHP  Hypertext Preprocessor
PSF  PhoneSensing Framework
RSSI  Received signal strength indication
SAP  Sensor Access Point
SN  Social Network
SSMA  Social Sensing Mobile Application
SSP  Social Sensing Platform
SSWA  Social Sensing Web Application
UI  User Interface
URI  Uniform Resource Identifier
URL  Uniform Resource Locator
XML  Extensible Markup Language
Chapter 1

Introduction

There is a continuous increase of cyclists all over the world. This is in part related with the increasing fuel costs, Europe declining economy and an higher environmental awareness.

The cyclist community has been demonstrating an increasing interest in discovering and sharing new and better routes. They take factors like security and enjoyability in consideration to choose an appropriate route. Cyclists also display an increasing interest in gathering and collecting metrics about their rides, in order to monitor their own performance.

Until recently, cyclists had to not only acquire expensive equipment to collect their metrics, they also had to acquire several devices, since each was specialized in collecting a specific metric. As an example, they would have to acquire an odometer to measure the traveled distance, Global Positioning System (GPS) to keep track of the traveled route, and an inclinometer to measure the slope during the ride. Furthermore, these devices did not communicate among themselves, so it was necessary for the user to develop a pipeline to clean and integrate the data, in order to get a clear overview of his performance. However, smartphones have been slowly replacing these specialized devices, due to the fact that nowadays smartphones already have a set of embedded sensors and interesting computing resources, but also due to their proliferation.

Another problem faced by cyclists, is that it was difficult to collect and share metrics and routes, allowing the creation of communities that encourage sharing this knowledge. But due to the rising popularity of Social Networks (SNs), sharing information with other users has become increasingly easier. This means that cyclists now have access to platforms and communities sharing and exploring new routes, organizing cycling events, and sharing metrics, therefore fostering the competition among members of the community.

Several existing platforms already address some of the problems introduced. Consider the following as example:

- **Bikely**: a route mapping and sharing website for cyclists\(^1\);

- **MapMyRide**: allows monitoring and sharing the cyclist performance, using social networks\(^2\).


Biketastic: also allows monitoring and sharing the performance, but also monitors the environment surrounding the cyclist, such as slope and the noise pollution of the route travelled[28].

1.1 Motivation and Goals

Existing platforms still do not take into account that cycling can be seen as a social activity: cyclists often perform their activities in groups. As such, these group activities present an opportunity to gather the collective data, using the sensors available in the rider’s smartphone.

There is already research work that explores opportunistic connections between devices in order to share resources, exchange information, and execute tasks remotely. It is possible to use this work to examine the possibilities of using opportunistic connections within cycling groups to improve the capabilities of the smartphones and improve the accuracy of the measurements of the sensors. As an example, one could improve the capabilities of a smartphone, or even improve the accuracy of the metrics collected, given that the cycling group has several smartphones available.

Furthermore, given that smartphones have the capability to be connected to the internet, it is possible to upload the raw data collected from the sensors to the cloud and use the almost unlimited resources of the cloud to process and analyse that data. Other interesting aspects of exploring this problem are:

- **Allow users to analyse performance metrics**: there is an increasing interest in the cyclist community to collect and analyse cycling activity data, such as maximum speed, average speed, distance travelled, calories burned, among others;

- **Allow users to discover cycling routes**: provide more information to the cyclist community regarding routes, so that cyclists can be able to choose a route that better fits their requirements: route with less noise pollution, less slope, or even routes recommended by other users;

- **Promote environment friendly lifestyle**: social networks are a good way to promote an active lifestyle while at the same time alerting society to minimize the environmental footprint;

- **New market segment**: cycling in group is not taken into account by most applications, creating a market opportunity for addressing an unfulfilled. Thus, this research allows to address real problems of the cycling community while presenting an interesting technical challenge.

1.2 Challenges

To tackle and solve the problems this dissertation entails, several topics were analysed, so that an appropriate solution could be developed. Some of the critical challenges that were faced while developing and implementing the solution, can be enumerated as follows:

- **Sampling Context**: is the set of conditions required for sampling to take place, including location and orientation. This is a challenge since it can be difficult to know what are exactly the conditions in which sensing is possible, that is, if high-fidelity is to be reached;
– Mobility - An interesting particularity of the architecture of this system, is that the nodes of the cyclists group are not frozen in a determined position, which makes it difficult to collect the metrics properly. Node mobility can also affect the opportunistic approach. In situations where a node has been commissioned to execute a certain task in behalf of another, if the distance between the two nodes is such that does not allow the two nodes to effectively communicate, the node that requested the task to be executed, might never get its result back, or at least not receive it in an acceptable time frame;

– Concurrence in the Bluetooth discovery process - If a user is using the “discovery team” functionality using the Bluetooth, other team member smartphones should wait for the device that initiated the request to finish the discovery process and all steps included in that process;

– Added value - To provide an added value service while at the same time trying to keep the battery consumption low;

– The PhoneSensing is a framework with several modules. Understanding the architecture and introducing the changes in a system of this dimension is by itself a challenge;

1.3 Contributions

This work provides several benefits to the cyclist community that can be categorized as: Personal, Social, and Public. At the personal level, single cyclists (those who are not cycling in group) can use the Social Sensing platform to monitor their individual performance, such as fitness metrics, or get their real time location. The platform also allows to permanently store their metrics, providing means for the cyclist to analyse their metrics and routes over time.

At the social level, i.e. when cyclists ride in teams or groups, the Social Sensing platform provides all the aforementioned services. Furthermore, it allows cyclists to see where each team member is in almost real time using a map, either using a mobile phone or even when they are not participating in the activity but want to track the team using the web. For each cyclist, information about the current location and route conditions are displayed (e.g. such as slope and noise pollution). Using the Social Sensing platform, cyclists from the same team are also able to share device resources such as GPS or internet connectivity. This means that even though one of the team members might not be able to access the internet, the other members from the group can provide such functionality.

At the public level, each cyclist can share the data collected by the Social Sensing platform with the community. This way, others can search and discover routes that fit their criteria. This platform also allows cyclists to share their fitness metrics in a social network such as Facebook.

1.4 Organization

This dissertation is composed of 6 chapters which are structured as follows: chapter 2 presents the user requirements and typical solutions which is organized by the preliminary studies done to identify cyclists
problems and the key features presented from the related work; chapter 3 represents the architecture of the Social Sensing Platform, including the requirements and a detailed specification; chapter 4 describes the technologies, equipment and the methods used to implement the planned system; chapter 5 is responsible for the describing evaluation and validation of the Social Sensing features; finally, chapter 6 describes the conclusions and the future work.
Chapter 2

User requirements and typical solutions

This dissertation addresses a problem faced by the cycling community, with an emphasis on the group cycling experience. This chapter analyzes implemented solutions that address the gathering of performance data, the location of riders, the discover of cycling routes and generally the features associated with the social, personal and public planes that can be compared with those provided by the SocialSensing application.

2.1 Type of Service

Many services are offered to gather data concerning cycling performance, cyclist health, fitness and the environment surrounding the cyclists routes. The existing services can be divided into mashups, geowikis, ride loggers, and participatory/opportunistic sensing.

The mashups\(^1\) are web pages or applications that use and combine data from two or more sources to create new services. Usually mashups specified for cycling activities are applications built on top of mapping Application Programming Interfaces (APIs) that enable the user to draw and share routes [28].

The geowiki\(^2\) introduces rich geo-editing features to the map where anyone can share notes about roads and trails, enter tags about special locations, and fix map problems.

The ride loggers use the smartphone sensors to record time, speed, average speed, distance and acceleration data during the cycling [28]. Additionally, the user can upload location traces to a website for visualization and ride analyses.

Participatory and opportunistic sensing are relatively new approaches to data collection and interpretation. In these approaches, participants use their mobile phones to collect data automatically (e.g., location logging) or manually (e.g., taking pictures)[13]. Afterwards, the data will be uploaded to the cloud, where a participatory or an opportunistic sensing system summarizes the results and turns them

\(^1\)http://blog.sherifmansour.com/?p=187 - accessed in September of 2013
\(^2\)http://cyclopath.org/ - accessed in September of 2013
into simple visualizations that can be shared.

The participatory sensing is useful to solve context problems, since the device counts with the user assistance. For instance, let’s suppose a user is going to cycle and the device will collect noise pollution data during the ride. Since the device does not know where it is located (e.g. on the cyclist pocket or backpack), a message can be shown to the user, asking for device location.

A drawback of this approach is that it gives a higher emphasis on the user side relying heavily on the enthusiasm of the participant, which is different from the opportunistic sensing [22]. Not having to deal with the user participation for the sensing tasks, the opportunistic sensing is able to cover more users.

The downside of the sensing paradigms systems is that the data collected through the sensors for the communal project may not be trustful. Because of privacy preservation, the data collected cannot be tracked to the source, reducing the fidelity of the data.

2.2 Architecture

The emphasized architectures in this document are divided into two parts: the centralized and the decentralized. This section will briefly explain the advantages and disadvantages of the both. The last point should not be noted as a full listing, but as slightly comparison.

The centralized architecture is the most cited example in the cycling systems and is usually based on a client-server model. This model may be horizontal where all clients only connect with a single server (that could be replicated to improve reliability on the service and the data), or it may be hierarchical for higher scalability [18]. In this case, the servers of one level act as clients of higher levels servers.

The server (e.g., powerful dedicated computers) has higher performance than the clients (e.g., mobile device, personal computer), which are mostly low performance systems. Consequently, clients count on servers for storage (e.g., files, save data), to execute tasks, to update data, to share information, etc.

Some advantages of this scheme are as follows: firstly, since the files are centralized, data management is less painful, therefore it is easier and quicker to perform backups and error management; secondly, the server’s hardware, which is more powerful than the clients’, can serve the requests faster; finally, after being collected, all the data is processed by the server and only the outcome will be seen by the client, thus reducing the network traffic between the server and the client.

A serious drawback of this approach is that it leads to an undesired single point of failure at the core, as it disrupts the whole interconnected system when the core fails to fulfill its tasks successfully. Also this architecture is very expensive and requires a great deal of maintenance, resulting in a poor scalability.

The decentralized architecture has two main subdivisions: Hybrid and Pure[12][18]. While the pure architecture does not require a central server to work, the hybrid needs to contact a central server to retrieve meta-information, as for the identity of the node or the security authentication credentials.

In the pure decentralized scheme, the nodes involved play similar roles and work in a cooperative way as peers without any distinction between clients and server. Hence these peers have to self-organize themselves and collaborate with the locally accessible peers (neighbours) using the available information which is usually dispersed across several nodes.
The hybrid scheme, as the name implies, combines the client-server architecture with the pure distributed architecture together. This means that the central server only acts as intermediary in the system. Generally, it performs two primary functions in the network: (1) it works as central directory where connected users and indexed data can be traced to the current IP address or location (e.g., device coordinates); (2) the server drives traffic among the nodes. Therefore, a node usually starts a connection with the central server (e.g., to require the identity/location of the node), succeeded by a direct communication with that node.

The advantages of this distributed architecture are: in the case of pure distribution, there is no single point of failure. That is to say, if one node stops working, the rest of the network is still able to work. This architecture provides opportunity to benefit from unused resources such as processing power for calculations and storage space. Contrary to centralized systems, where the central servers carry the most of the cost of the network, the decentralized scheme disseminates the cost among the nodes (e.g., the files are saved in several nodes, and not in one server). Since peers communicate with each other, it is less costly to scale and leads to less bottlenecks, such as traffic overload, due to the absence of a centralized control.

The biggest pitfall of this architecture, especially in the opportunistic connections case, is the volatile connectivity. Although this problem is mitigated by the hybrid approach, there is still a variability from connection to disconnection between the devices. What’s more, the quality tends to be unstable as the connections between the nodes usually are not planned for high data rates. The distributed architecture introduces various difficulties at different levels of implementation than the centralized approach. For example, if all the information is residing locally and does not need to communicate with any resources outside, security is not as complex as in the distributed architecture, where the client handles applications coming from an unknown source.

2.3 Mobile Sensing Networks

Smarphones can be used to collaboratively accomplish sensing tasks, thus offering more services, saving energy and reduce data redundancy or higher data accuracy, depending on the importance of the data fidelity. The smartphones can explore the resources from other available devices using a centralized solution or a distributed solution.

This section of the document will focus briefly on a couple of mobile sensing networks platforms alternatives, mainly Halo, Quintet and the PhoneSensing.

The Halo scheme use a centralized approach to this problem [9][8]. The architecture is composed by the Sensor Access Points (SAPs) (e.g., a mobile or static sensor) and Mobile Sensors (MSs).

All requests must go through these SAPs so, for example, every time a sensor wishes to send a request it has to move within the sphere of interaction\(^3\) of a SAPs. The SAP will then task an appropriate mobile sensor within its range for the requested task. This mobile needs to remain within the SAP sphere until the task ends. If such does not happen, the SAP tier and involved mobile sensors keep the state

---

\(^3\)radio range.
of the active incomplete tasking operation. Thus, the architecture is able to ‘pause’ and ‘resume’ the tasking as a mobile sensor moves in and out of SAP coverage areas.

The sphere of interaction is implemented using transmission power control and multi-hop signalling between SAPs and MSs near the respective SAPs. The range will vary conform the requests arrive to the SAP, in order to reduce power consumption. For example, when the application demands increase, the range of the sphere will also increase.

The Halo also takes into account delay-aware applications, and permits MSs to stipulate a degree of time sensitivity along with the tasking request in an effort to lessen the tasking prediction problem and offers an extra effort to produce the tasking results within the desired time frame.

A serious drawback of this approach is that it requires extra infrastructural deployment, otherwise the system won’t work properly.

The Quintet system uses a decentralized architecture to the opportunistic sensing approach, so all devices must run an instance of the developed code. The protocol used includes a distributed context analysis engine that determines sensors in the neighborhood that best match the query. The queries indicate a sensor category to be sampled and the rate and duration of the sampling. Furthermore, to improve the nature of the data gathered, queries may also indicate a set of qualities of the sampling context (e.g., the orientation of the sensor on the body or the clothing).

In order to find a better-suited solution from the surrounding nodes, Quintet suggests a three stage process: resource discovery, resource selection and data transfer. In the discovery stage the mobile device broadcasts a sharing solicitation message that contains information about the sensory type and context requirements. The devices that meet the query conditions reply, and after receiving the reply message, the primary node enters in a selection stage. In this stage, the node contrast all the replies and selects the best responding solicited nodes. Upon selection, sends an unicast message to the node that fits closely with the query requirements. The association with the selected nodes is complete when each of the nodes replies with an acknowledgement.

Although the sampling context is important, in a small network of devices, such as a group of cyclist, the strict interpretation will discard several nodes if not all, that could provide the required information, although with not the best sampling context desired. Also, this strict interpretation most likely will lead to excessive delay in acquiring the necessary samples, which is not plausible in an environment of uncontrolled mobility and short rendezvous situations that the devices will most probably be confronted with.

The PhoneSensing framework (PSF) is a sensor sharing Opportunistic Sensor Network (OSN)[31]. It uses a decentralized architecture, and consequently all mobile devices have to run an instance of the developed code. This decentralized architecture allows high scalability and availability since is not dependent of cycling environment (does not require any extra infrastructural deployment). This framework uses a protocol that takes into account the following requirements:

- Message exchange can not be high in order for it to scale in high node density scenarios;
- Conversion times have to be low so that in low rendez-vous times the protocol still allows for
resources to be shared;

- Exchanged messages have to be small so that transfer times aren't long;

The protocol, named Multiple Active Request (MAR), works as follow:

When a device requires a sensor it does not have, he sends a Capability Request Message\(^4\) to all neighboring devices. It then waits for a Capability Response Message\(^5\). Upon receiving the resources available it checks if any device has the desired sensor. If this is the case, then a Sensor Request Message (SREQM) send a RDL describing the requested sensor, and will not stop sending SREQM even if one has already been sent. It assumes that the message might fail, or the remote node might not comply with the request. Hence, it sends as many requests as possible, and then waits for the first one to reply. With this protocol, after sending a Sensor Reply Message\(^6\) all other reply messages are ignored. Messages are only resent if they failed to reach the destination. They all have a Time To Live (TTL) value associated, this is so if a request reaches the destination and the destination goes out of range for a long time, the reply is not sent when / if they connect in the future. Although this protocol may send a lot of messages, flooding the network with messages, it has the advantage of converging faster, since it does not depend on just the remote device.

### 2.4 Social Networks APIs

The social networks are used on a regular basis by one-billion-plus people [29]. The major players in this arena are the Facebook\(^7\), Twitter\(^8\) and Google+\(^9\). Social Networks (SNs) facilitates sharing information with other users, thus incorporating social component into an application allows the cyclists to compete with riders from all over the world, and share their scores with their peers. Another important aspect of incorporating the social component into an application is virality. Thus a social application can have incredible viral growth. Usually the SNs open their Application Programming Interface (API) to allow developers to integrate the SN with their applications.

The API\(^10\) is the interface that allows other applications to communicate with the main application. They uncover chosen functionality and data while preserving other parts of the application. When a company develops an API, more applications will interact with it, and the more popular becomes the application which provides the interface. So, an API simplify communication enabling distinct applications to connect with each other, popularising the application. When one or more application incorporate data and capabilities to create another application, we have a mashup (see section 2.1).

All the three major SN API use a RESTful API design\(^11\) 12 13. The Representational State Transfer

---

\(^4\)This message is sent to a device when the sender wants to query the remote device’s available resources.
\(^5\)Carries a list with all the available sensors, represented by a byte array. This byte array is the Resource Description Language (RDL) describing the available resources in Key-length-value (KLV) format.
\(^6\)Send a message represented simply by the character array, CAP_REQ.
\(^13\)[https://dev.twitter.com/docs/api](https://dev.twitter.com/docs/api) - accessed in September of 2013
(REST) is an architectural style by which Web services that target on system’s resources can be design. The architectural design includes how resources states are addressed and transferred over Hypertext Transfer Protocol (HTTP) by a vast group of clients written in different languages. The REST web service uses four key design principles:

- Use HTTP methods explicitly;
- Be stateless;
- Expose directory structure-like Uniform Resource Identifiers (URIs);
- Transfer Extensible Markup Language (XML), JavaScript Object Notation (JSON), or both.

The REST is a simpler alternative to Simple Object Access Protocol (SOAP) and Web Services Description Language (WSDL), since is easier to use and is resource oriented, as the shift in interface design attest. One of the key points of this API design is the explicit use of HTTP methods. It is using this low-level HTTP-based API that the developer can use to query data, post data, between other tasks that an application might require to do to the SN.

The Facebook, Google + and Twitter use the OAuth authentication protocol to allow users to approve application to act on their behalf without the need to share their password. The functionalities offered by the APIs, are very similar, although with different names. For example, all of them allow authenticating using their services, or share data on the user space. The difference is the API calls quota to third-party services.

The Facebook API call limit are: no more than 5 million monthly users or than 100 million API calls per day or no more than 50 million impressions per day. The Google+ has a limited number of API calls of 1000 requests / day, which can be expanded upon request. The Twitter offers two types of API calls: (1) Per User; (2) Per Application. If Per User, it has a limit of 15 request per rate limit window, when using Per Application rate limits are determined globally for the entire application. If a method allows for 15 requests per rate limit window, then it allows you to make 15 requests per window –on behalf of your application. The limit window is 15 minutes.

The Facebook is the only SN, in this group, that has an API to create multiple accounts for testing purposes, named Test User API. The Test User API, which is created on behalf of an application for the purpose of testing that app’s Facebook integration, allows creating hidden user accounts associated

---

15http://www.w3.org/TR/soap/ - accessed in September of 2013
16http://www.w3.org/TR/wsdl - accessed in September of 2013
17http://www.w3.org/Protocols/ - accessed in September of 2013
19https://dev.twitter.com/docs/api/1.1 - accessed in September of 2013
20https://developers.facebook.com/docs/api/1.1 - accessed in September of 2013
23https://dev.twitter.com/docs/auth/using-oauth - accessed in September of 2013
24http://oauth.net/ - accessed in September of 2013
26https://dev.twitter.com/docs/rate-limiting/1.1 - accessed in September of 2013
with an application for the intent of testing the functionality of the application. In conclusion, most of the
APIs available share the same design principles, and the functionalities offered, such as post data to
the user wall or retrieve data, share very similarities. The main differences are in the API calls and the
number of users each platform has.

2.5 Mobile Application Interaction

Creating mobile user experiences that satisfies users is not easy. It is difficult, in particularly because of
mobile-specific considerations like the small screens and the ever changing mobile context.

2.5.1 User interface

The User Interface\(^{29}\) (UI) should be simple and efficient in terms of accomplishing user goals. The
graphical user interface (GUI) integrated with a touch screen display is the most widely spread way of
allowing users to interact with the smartphones. Nonetheless, when it comes to the user safety, it may
not be quite good for cyclists to take his hands off the wheel or distract his attention away from the
surroundings.

None of the analysed systems address this problem. One workaround to this problem is to use voice
control to interact with smartphone. The system described by Zdena Dobesova in [7] allows users to
to control the map zoom and navigation using voice commands. This system splits the commands for
controlling maps into two sets: the first set allows changing the scale of the map, and the second makes
use of search commands to look for places and routes.

This technology needs to be improved, since it still presents challenges in interpreting the commands
of the user and struggles with the ambient noise. Given that every interface system has its own pitfalls
the better approach would be to combine several technologies into a multi-modal interface for the user.

2.5.2 User profile

A user profile is a collection of settings that make an equipment look and work the way the user specifies.
This allows the system to change its customizations accordingly to the user. Usually the user profile
is used to store information about the user (e.g. age, weight, height, etc), but can also be used to
contain more than static rules, they can also store dynamic information to serve as input to data-centric
applications, aiding it in its decision-making process when one or more events occur.

2.6 Privacy and Security

The social sensing applications provide numerous challenges to guarantee the privacy of the data
shared, since the information provided by the user can be used for criminal purposes. One solution

\(^{29}\)http://themes.jdsans.net/zend/?p=102 - accessed in September of 2013
to privacy is to offer different privacy levels to the user, or to allow the user to "lie" (perturbed data) about their information without corrupting application quality [5].

To solve security issues, the data can be processed locally and encrypted before sending it to the cloud. However performing those techniques on the mobile device requires additional processing, consumes additional energy and it does not assure that the problem of security and privacy is solved [15]. Even if the user is willing to share the information on the SNs or anonymously on a web site (e.g. for a communal project) he may not be willing to share phone resources with another person.

An interesting perspective to address this problem is to explore the human social structure [22]. For example, although CenceMe share important data (e.g., location and activity), they do so within groups in which the user have an trusted relationship based on friendship or a shared common interest such as reducing their carbon footprint[10].

Concluding this section, the users should always be able to control the privacy policies. In this way users can determinate how much of their presence or sensitive data is shared and with who they are willing to share it. Additionally we can say that privacy and anonymity will persist as an important issue in mobile sensing systems for the foreseeable future.

2.7 Functionality

The Social Sensing project (SSP) addresses real problems faced by the cyclist community. Thus, the initial design of the SSP involved guidance from a small group of local cyclists. We asked this small group of cyclists what they would be interested in to improve their cycling and group experience, beyond the usual services offered by the current market applications (e.g., speed, calories, distance, etc).

The most wanted features obtained from the input of this local group were:

- track the team members on a map - when cycling in large groups cyclists often fell apart, and usually to find the rest of the team was necessary to perform a phone call, which would kill the cycling mood and experience;

- share the phone resources (GPS, microphone, internet) - Smartphones do not have all the same resources, thus a cyclist without, for example, GPS may not know his current position. When cycling in group, this should not be a problem, since it is possible to share sensor data with those who are missing sensors. Thus, giving the same cycling experience to all the group;

- Route’s noise and steepness in advance - Due to the lack of a platform where bikers can share routes and experiences, the better routes are often discovered through an informal process of trial and error combined with knowledge sharing in the cycling community. With the amount of resources available on the web, it should be possible for cyclists to find and share good routes, where the condition of a route is based on safety, steepness and enjoyment;

After finalizing the interviews with the local cyclists group, we create a more extensive survey to inquiry the utility and validity of the results obtained with these interviews. The survey was conducted
using the google forms\textsuperscript{30}. Several online pages and groups related with cycling and sport were used to spread the survey. At the end of the survey, 133 answers were collected. The survey can be consulted in the section A.1, figure A.21. One of them was discarded for having invalid information.

Before attesting the validation of the services offered to improve the cycling experience, it is relevant to analyze other questions/answers provided by the survey. The percentage of cyclists that cycle often with friends/team are 80\%, and 77\% of those who cycle with friends, said the size of the group was 2 or up to 4 members. From those who possess a smartphone, 73\% uses the Android Operating System (AOS). Consequently developing a mobile application for Android first, is a good option to captivate the maximum number of users as possible.

The number of users that use a fitness application to track their results is about 23\%, when asked why, 55\% says is too expensive or the UI is too confuse, while the rest said mainly because they do not need, or their smartphone is very limited. When asked if the application they use to track their rides improves the group cycling experience, only 53\% said yes, a very low number.

Returning to the central questions, when asked about the importance of tracking the team members on a map, 76\% said it would be useful. Truly a high rate of acceptance (see figure 2.1).

![Figure 2.1: Validation of "track the team members on a map".](image)

When asked if they would share their phone resources (GPS, microphone, internet) with other team members, 41\% said yes and 38\% said maybe. The importance of privacy is demonstrated clearly when asked the reason behind the rejection, 83\% said privacy concerns. If this issue is well address, it is possible to achieve, at least, 79\% of the users. The third idea, achieved the higher validation (see figure 2.2), 82\% validate the usefulness of knowing the route’s noise and steepness in advance.

![Figure 2.2: Validation of "Route’s noise and steepness in advance".](image)

Also when asked, if they would collect noise samples from a cycling route, 53\% said yes and 32\% said maybe. The results of this study validate the ideas suggested by the initial group of cyclists. Given these facts, these ideas will be part of the functional requirements addressed when comparing the existing applications.

\textsuperscript{30}https://support.google.com/drive/answer/87809?hl=en - accessed in September of 2013
2.8 Related work

In this chapter we introduce an analysis of the most used platforms for each type of service described in section 2.1. The chosen platforms are Bikely\textsuperscript{31}, Cyclopath\textsuperscript{32}, Bikemap\textsuperscript{33}, Nike + sportwatch GPS\textsuperscript{34}, Endomondo\textsuperscript{35}, Biketastic\textsuperscript{36} and BikeNet\textsuperscript{37}. The analysis will be present in a top-down way, with the simplest platforms being explained first and moving forward to the more completes.

2.8.1 Bikely

The Bikely\textsuperscript{38} system is a route bookmarking site, a place to find and share cycling routes around the world. The routes are drawn or uploaded from the GPS by the community and the user can choose any route and outlook submitted biked routes nearby on a Google Map, with explanatory notes about the route. These routes are available for download. However it is not required for the cyclist to ride the routes they share\cite{28}, which could lead to misleading information.

2.8.2 Cyclopath

The Cyclopath\textsuperscript{40} is a geowiki, which lets users enter personal bikeability ratings for roads and trails. They hope with their unique rating system provide better information to help the users find the best routes for their requirements.

The user can also use their Android application to find routes or track their ride. Being a geowiki, the application fails to give good information about the route, and depend of the user enthusiasm to give good descriptions about the routes. The application does not provide any more functions beyond map information.

2.8.3 Bikemap

Bikemap\textsuperscript{41} is also a geowiki, like Cyclopath (see section: 2.8.2). Their goal is to help users find new bike routes. It is worldwide used with more than 800 000 routes curated by their community. They also have a mobile application which helps the user with navigation. The main functionalities offered by their application are:

- Search for a route;
- View routes on a map with distance, description and rating;

\textsuperscript{31}http://www.bikely.com/ - accessed in September of 2013
\textsuperscript{33}http://www.bikemap.net/ - accessed in September of 2013
\textsuperscript{34}http://nikelife.nike.com/plus/products/sport_watch/ - accessed in September of 2013
\textsuperscript{35}http://www.endomondo.com/ - accessed in September of 2013
\textsuperscript{37}http://metrosense.cs.dartmouth.edu/projects.html - accessed in September of 2013
\textsuperscript{38}http://www.bikely.com/ - accessed in September of 2013
\textsuperscript{39}http://lifehacker.com/191120/find-your-cycling-route-at-bikely - accessed in September of 2013
\textsuperscript{40}http://cyclopath.org/ - accessed in September of 2013
\textsuperscript{42}www.bikemap.net - accessed in September of 2013
– Interactive elevation profile;
– Current location while riding;
– Share routes with your friends, using either Facebook or Google+;
– Record routes (for later uploading to their back-end);

The Bikemap is very focused on maps and roads, but suffers from the same problem of all participatory systems, the user needs to be engaged to provide good routes description.

2.8.4 Nike + sportwatch GPS

Nike + sportwatch GPS\textsuperscript{43} is specialized fitness hardware that provides athletes with dynamic and motivational running experience. This fitness hardware fits in the ride loggers category and uses GPS data and a sensor on the shoe to track the user pace, elapsed time, distance, heart rate (require another sensor) and calories burned.

The system uses a hybrid architecture approach. The connection between the watch and sensors are done in a P2P fashion, while the connection with the back-end is done using a centralized approach. The user can use the website of Nike\textsuperscript{44} to see and share routes, and add tags of interest.

Nike+ sportwatch GPS is also social. It allows the user to share his achievements with friends or challenge other users to perform the same tracks in less time. The Nike+ does not make use of the potential made present when the user is running/cycling in group in order to expand the user experience and provide more fitness data.

2.8.5 Endomondo

The Endomondo\textsuperscript{45} system is a good example of a ride analytics, and as such it is a sport community based on real-time GPS tracking of running, cycling and other sports. This system has an hybrid architecture, where the smartphone can interact\textsuperscript{46} with a wide range of sensors to provide extra information (e.g. measure heart rate), or with the back-end for the purpose of storing, ride analysis and future visualization. The main services offered by the mobile application are:

– current speed;
– average speed;
– total calories burned;
– distance travelled;
– real time location;

\textsuperscript{44}http://nikeplus.nike.com/plus/places/ - accessed in September of 2013
\textsuperscript{45}http://www.endomondo.com/ - accessed in September of 2013
\textsuperscript{46}http://www.endomondo.com/features - accessed in September of 2013
The application also displays other functions, such as “Beat myself” or “Beat a friend”. For instance, in the “Beat a friend” feature, the user chooses a friend from his list, and the pick the “Personal Best” of that friend. To motivate the user, the application offer an “Audio Coach”. This coach offer different functionalities depending on the workout (e.g., audio feedback for every mile or kilometer during the workout).

The Endomondo web site offers the same social functionalities as the application, plus training statistics and history, information about the routes (e.g. best time achieved, number of times tracked, score and comments). Beyond challenging and sharing data with friends, the Endomondo ecosystem does not leverage the possibilities made available when a cyclist ride in flocks (e.g., “borrow” sensors not available in the local smartphone). Also instead of using the sensors available on the smartphone to provide rich information about the ride environment conditions, the user is asked to manually add descriptions to the route. A serious drawback of the Endomondo is that most of the services described before, needs to acquired separately (e.g. Beat Yourself: USD 1.99, Pro: 4.99).

2.8.6 Biketastic

The Biketastic is a platform that helps users finds good routes. By using a mobile application and online map visualization, cyclists are able to document and share routes, ride statistics (e.g., calories burned, average pace, and route speed), sensed information to infer route roughness and noisiness, and media that documents ride experience [28].

This platform uses a centralized scheme, with the mobile gathering route data and the back-end as the sensor storage database. For instance, as the user is cycling, the data is being collected by the smartphone, and as it is gathered it is uploaded to the back-end. Once the cyclist announces the ride has ended, successions of operations are performed to acquire extra inferences, to collect accumulated metrics about the ride, and for visualization purposes. The author of [28] does not mention what happen if the user does not have connectivity to the back-end, and so, we can only extrapolate that probably the data is saved in the smartphone, until connectivity to the back-end is available. The BikeTastic uses the GPS on the mobile phone to provide average pace, route speed, calories burned and the accelerometer and microphone to infer the road roughness and noise level along the ride. To provide more complete information about the route, they use a participatory sensing approach, by asking the user to capture geo-tagged media (e.g., images, videos clips) of the route while they ride. With this data, the web portal of BikeTastic is able to provide route information, were the cyclist can observe the health of a route (e.g., noise pollution, steepness).

The document of [28] contains a number of new and important insights about the cyclist experience, when faced with participatory sensing: According to their survey, the users did not enjoy dialogues to indicate the position of the phone every time, and that the device should remember past configurations

---

47The friend must also use Endomondo. - accessed in September of 2013
and have a simple “start” button. Also the study says the users disapproved the media capture, and suggested that “easy-to-use” tags would be a better approach.

This survey is particularly relevant, since it shows how we should address the interaction with the user. To put it in another way, the cyclists want a system that remember previous settings and the application should be the more autonomous possible and count as less as possible with the user assistance to solve context problems. Although this system provides rich information about the user environment, such as road conditions and noisiness, and do not regard that there is a strong social element to cycling, with many cyclists riding in groups.

### 2.8.7 BikeNet

The BikeNet system is a versatile sensing system for cyclist experience mapping that takes advantage of opportunistic networking principles and techniques[26]. Their system present an hybrid architecture with a combination of Halo and Quintet systems (see section 2.3) for support of their sensing, tasking and data collection. They explore inter-bicycles networking for resources sharing, data sharing, data mulling, real-time and delay-tolerant uploading of data using Sensor Access Points to a back-end repository.

The system use several sensing devices (e.g., GPS, speedmeter, tilt, stress monitor, between others) installed in different positions of the bicycle that are networked over a common IEEE 802.15.4\textsuperscript{51} short range radio channel. To each sensing device is assigned a specific role. This assignment is done via a resource discovery protocol based on the capabilities of a Moteiv Tmote Invent\textsuperscript{52} mote to provide the required sensor readings. The user can also set different profiles in the mote to include a list of preferences in order to specify desirable additional sensing roles, namely which sensor to use and sampling parameterizations.

Three forms of data transfer happen in the BikeNet system: (1) tasking exchange, (2) uploading exchange, (3) mulling exchange. The tasking and uploading data exchange can be done using the Quintet or Halo approach, this is, it occurs between the mobile sensing platforms and SAPs. For instance, a SAP tasks an available mobile sensing platform (e.g., Tmote Invents). Upon receiving the request, the Tmote, will see his preference profile, and assembles the specific device in the platform to perform the desired sensing task, as described in section 2.3 (e.g., Halo). Another example, of data tasking is, if a mobile sensing platform does not hold all the preferred sensing roles as desired by the user, it uses an opportunistic sensing approach to request sensed data from another sensing platform nearby. In order to discover which platform provides the better requirements, the mote uses the same protocol explained in section 2.3 (e.g., Quintet). Beyond the profile list of preferences where the user can choose which sensor to use, there is little evidence of how privacy is assured, since any BikeNet system can require a sensing task from another system.

The third form uses only the Quintet approach. When mulling exchange occurs, sensed data is transferred between the mobile sensing devices (native to the same platform), in an attempt to increase

\textsuperscript{51}http://www.ieee802.org/15/pub/TG4.html - accessed in September of 2013

\textsuperscript{52}http://www.tempsensornews.com/generic-temp-sensors/tmote-inventtm-wireless-sensing-system/ - accessed in September of 2013
the probability of near real-time delivered data. Since the devices are probably attached to the bicycle they may be within the range of a SAP for uploading. The device that receives this data cannot replicate it. The authors’ offers insufficient information of how the mobile sensing device assures that the mulling exchange occurs in the same platform, or data is not replicated more than once.

The services offered by the BikeNet mobile sensing platform to quantify the cyclist experience are: current speed, average speed, total calories burned, distance travelled, path incline, heart rate and galvanic skin response. The galvanic skin response is used as an indicator of emotional excitement or stress level.

The system also uses opportunistic sensing to provide information about routes in terms of pollutions, allergen, noise and terrain roughness levels. These measures are then uploaded to the back-end, for data archiving, retrieval and/or visualization. The user can also use their web portal for real time sharing and archived cycling related data.

To sum up the foregoing, the BikeNet is a system that uses several approach to better quantify the user ride and experience, but sensing occurs in many devices, becoming expensive and impractical for cyclists. Also it does not take into account that users like to challenge other cyclists or share information about their performance in the SNs.

2.9 Summary

After presented the state of the art, the table 2.1 offers a global view of the cycling application organizing them by the general approaches presented previously.
<table>
<thead>
<tr>
<th><strong>Key features</strong></th>
<th>BikeNet</th>
<th>BikeTastic</th>
<th>Endomondo</th>
<th>Nike+ sportwatch GPS</th>
<th>Bikemap</th>
<th>Cyclopath</th>
<th>Bikely</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type of Service</strong></td>
<td>Opport. sensing and ride logger</td>
<td>Both Sensing paradigms and ride logger</td>
<td>Ride logger</td>
<td>Ride Logger</td>
<td>Geo-wiki</td>
<td>Geo-wiki</td>
<td>Mashups</td>
</tr>
<tr>
<td><strong>Architecture</strong></td>
<td>Hybrid</td>
<td>Central. solution</td>
<td>Hybrid</td>
<td>Hybrid</td>
<td>Central. solution</td>
<td>Central. solution</td>
<td>Central. solution</td>
</tr>
<tr>
<td><strong>Mobile Sensing Networks</strong></td>
<td>Halo and Quintet</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Social Network APIs</strong></td>
<td>-</td>
<td>-</td>
<td>All major SNs</td>
<td>All major SNs</td>
<td>Facebook</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>User Interface</strong></td>
<td>GUI</td>
<td>GUI</td>
<td>GUI</td>
<td>GUI</td>
<td>GUI</td>
<td>GUI</td>
<td>-</td>
</tr>
<tr>
<td><strong>User profile</strong></td>
<td>Charact. of a person + Triggers</td>
<td>Charact. of a person</td>
<td>Charact. of a person</td>
<td>Charact. of a person</td>
<td>Charact. of a person</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Privacy and Security</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td><strong>Survey Functionality</strong></td>
<td>Share the phone resources and route environment mapping</td>
<td>Route environment mapping</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### 2.10 Discussion

The Endomondo and Nike+ sport watch are designed primarily as exercise and recreational bike planning/tracking tools. They do not capture the information about the road roughness and noise levels using the sensors embedded in the smartphone (in the case of Endomondo) and expect the user to tag the information. This participatory sensing approach has the problem of relying heavily on the enthusiasm of the participant. The Cyclopath, Bikemap and Bikely do not require the users to ride the routes they...
share, and also rely heavily on the interest of the participant. The Biketastic and Bikenet are the only systems that try to provide all the type of services refereed on section 2.1. These two systems does not use SNs to provide a better service for the cyclists and only Bikenet address the problem that cyclist often perform their activities in groups. As said before when cyclists cycle together presents an opportunity to provide added values services. The Bikenet researchers analyze the correlation amongst the metric readings of the group of cyclists and address the sharing sensor data with those who are missing sensors [26]. While the sensor data is done locally, the correlation of the data is done at the cloud. But the Bikenet system requires the cyclist to buy expensive specialized equipment. Furthermore the user has to manage several sensors in various positions on the bike.
Chapter 3

Architecture

This chapter will focus on explaining the project’s requirements and on reviewing the chosen architectural approach. All layers and modules will be introduced in a top-down way. By doing so, it will help to understand the next chapter, where we will detail the project’s implementation. Some choices will be backed by results presented in the Tests and Validation chapter. Others will be explained in the Implementation chapter.

3.1 Requirements Analysis

The project requirements took into account the project goals and that the SSP is a network made of mobile nodes, with characteristics of uncontrolled mobility and presumably infrequent radio network connectivity. It was also designed in accordance with the results of the survey in section A.1, figure A.21.

3.1.1 Functional Requirements

Cyclist Performance/ Fitness Measurements The mobile system should gather and save data about the following cycling performance metrics: current speed, average speed, max speed, distance covered, calories burned and duration. These are the metrics commonly available in the existing commercial products.

Environment Mapping The Social Sensing Mobile Application (SSMA) provides information to the cyclist about the healthiness of a given route in terms of noise pollution and steepness. This information shall be tagged with location and time to provide contextual information. The environmental information should be presented in the web site to the cyclist community.

Advanced services Cycling has a strong social element attached. Often, when cyclists ride in group, they may not start all at the same time or they split into smaller groups as the ride goes on. This may become a hassle when they try to reunite. In order to solve this problem the application will provide an easy, self-acting, near real time and intuitive way of tracking the team members, be on the SSMA or using the Social Sensing Web Application (SSWA). Also, the cyclist may want to know the environment
condition of distant elements of the team. This information should be showed in an intuitive way together with the location of that element.

Cycling in group also allows to leverage the capacities of the less modern smartphones. To do so, the mobile application will share phone resources, such as Global Positioning System (GPS) or internet, on request. Thus, giving a homogeneous cycling experience to all the group.

In addition, the cycling team should not be dependent on the GPS embedded in the smartphones. If no GPS is available in the group, the system should be able to provide in self-acting fashion, relative positioning\(^1\). In case one or more GPS sensors are available in the group, they should be used to estimate the cyclists’ position in the earth’s reference frame.

**Team formation** The cyclist often cycle in environments where Telecom services (e.g., internet connectivity) cannot be reached. This should not be an obstacle to create a cycling team. Using wireless radio technology available in the smartphone, which is not infrastructure dependent, the SSMA should be able to create a cycling team with the same information, such as users names, and respective capacities of their smartphones, as if it was done using the web site.

**Data Query and Remote presentation** The SSP has a web portal on the back-end. The user can insert queries into the back-end to demand information of interest, such as: team location, individual (or group) fitness metrics, route environment conditions, create/delete teams and add users to teams.

**Social Networks** Social networks have “exploded” in number of users, being millions added every year. In order to facilitate sharing information with friends, the SSMA should allow the cyclist to share his performance metrics, using at least one social network. Furthermore, in order to facilitate the registration/login should be given to the user the possibility of performing those actions using his network account.

**Long Term Performance Trend Analysis** The data gathered on the ride should prevail long after the ride from it is collected. The SSMA shall upload the information into a personal repository. Some of this data, will only be available to the user, such as his performance, other will be available to his team-mates (if allowed by the user), such as his last recorded position, tagged with date to provide more rich context. Furthermore, part of the data, as the steepness and noise pollution will be shared with the rest of the community.

**User Preferences** The mobile application should allow the user to modify the application settings in accordance with his preferences. For example, which data is collected and displayed during the ride, which sensors are used, etc. This should be remembered by the application.

### 3.1.2 Non-Functional Requirements

**Disconnected Operation** If connectivity is available, and the user preferences allows it, the mobile application shall provide near real time information to the back-end, in support of real-time sensing and real time location, in case of environment mapping and team location, respectively. Nevertheless, the

\(^1\)Definition of relative position accordingly with [6]: “Instead of determining the position of one point on the earth with respect to the satellites (as done with GPS positioning), the position of one point on the earth is determined with respect to another “known” point.”.
SSMA will operate, by default, in a delay tolerant sensing mode. The cyclists go on ride, collecting data, and uploading it upon arriving home.

**Concurrency** The SSMA should allow resources sharing with other mobile devices in the team. As such, the SSMA should be able to handle concurrent request, for example requests for the microphone or accelerometer sensed data, from two or more mobile devices. Therefore the SSMA must provide a scheme of concurrency control.

**Robustness** The SSMA will be confronted with unrestrained mobility of the nodes. As such it is important for the suggested solution to consider the short rendezvous situations that devices will most probably be confronted with.

**Automatic** The user will use the application to cycle, and not to worry if a service is being done or not. Consequently the application will have to be the most autonomous and self-acting as possible. For instance, the user does not need to worry if there is connectivity to upload the data to the back-end or not. The application should verify the best time to upload the data by itself. Another example is the team tracking, if a cyclist is cycling with a team, the application should provide this functionality periodically without the user interaction.

**Security and Privacy** The system should guarantee a secure and authorized access to it, in order to assure the integrity of the proper functioning of the system. When the user logout the mobile application, the data in the smartphone should be deleted in order to provide security and privacy of the data gathered. The data shared with the rest of the community must preserve the anonymity of the person who shares it.

**Operation Environment** As an Android application, it will be designed to interface with the hardware present on Android phones. Theoretically the application can also be run by other devices which can emulate the Android, but this will not be a consideration during the design.

### 3.2 Use Cases

The main concept of Social Sensing project was to design a system that would be both functional and easy to use. These guidelines are put into use to all situations of interaction with the system and can be classified in three perspectives: (1) Single user; (2) Social user; (3) Web user. This section underlines the different users’ perspective of the system in order to better understand the next section.

**Single user perspective** It is a user with a mobile device and SSMA installed on it. The user can opt to register, login or simply skip directly to the application without registration. Although users are advised to log in if they have an active Social Sensing account, since some functionality won’t be available, such as save data to the cloud. This user is mainly interested in personal sensing, such as his fitness metrics and other custodial information about his cycle patterns. He is also interested in visualizing his real time location and past trace location on an intuitive map.

His main tasks are: Create a cycling profile (e.g., city ride, mountain ride, single ride, team ride, etc) in order to customize his settings and privacy policy, and start riding. He can also collect noise samples

---

2The choice will be explained in the Implementation.
from a cycling route or share his ride information with his friends with his social account.

Social user perspective The second user perspective is an extension of the single user perspective. Thus the social user has the same characteristic as the single user, but is also interested in the type of team information, such as average and median metric values, that is shared within the group.

The social user is divided into two views: team leader, which is the one who builds the team; and the rest of the team members. The team leader will be the one responsible for building the team and assuring synchronization in this process. Also the team leader is responsible for sharing the location of the team members with those nearby, allowing the team to have a common experience. The team members that are far will have to use their own Internet service to know the location of the team. Beyond this, all functionalities are the same.

If the elements of this social group are not registered they are only allowed to use team functionalities supported by Bluetooth, such as make a temporary team to share resources or track short range team location. In case the users are registered they are able to save the team created with Bluetooth or use a team that is already formed in the cloud. They are also able to track long range distance members and their environment conditions.

Web user perspective This user can only access web functionalities if he is already registered, otherwise he can only see some basic information about the mobile application functionalities, login/register options and project author personal page. In case he has logged in he can plan in advance his ride, thanks to the SSWA that provides crowd-sourced environmental conditions with intuitive colors to describe the intensity of the noise pollution or steepness.

The web user has the following functionalities: analyze personal/team metrics, edit profile, build/edit teams and track his team location in almost real time.

3.3 Architecture Design

The proposed architecture for this project requires different situations to take into account, which can be divided into three different main schemes: (1) Back-end, (2) Web Application and (3) Mobile Application. Figure 4.4 illustrates an overview of the project system: Cyclists gather performance and environmental data along the ride; Application tasking can take place when cyclists are within the Bluetooth range of the rest of the team; Location sharing when the user has an internet connectivity.

3.3.1 Back-end

Given the use cases described in the previous section, there is a need to keep the user data in a centralized way. This way is possible for the user to access his data from different places and devices. Implementing a back-end application that abstract the access to the database allows to develop a service oriented architecture were virtually any client can request information from the database. Clients will make requests using the Hyper Text Transfer Protocol\(^3\) (HTTP). We propose the back-end to use the

\(^3\)http://www.w3.org/Protocols/ - accessed in September of 2013
web service\(^4\) architecture. Web services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks. This architecture has three main benefits:

- It is possible to create an abstraction layer that simplifies the client application architecture, since these do not have to deal directly with the database.

- The client application is independent from the back-end, since they only interact through web services; the client application can be written in a different programming language from the back-end application.

- Only the client application can start an interaction by making a request to the back-end application; this makes the back-end application simpler since we only have to react to these requests.

The API provided to the clients applications will allow them to submit application requests/queries to authenticate the user, get team information, register new teams, upload fitness metrics to the back-end, among others.

The back-end has a module which is responsible for processing the database connection information and every time one of the clients modules want to connect to the database, will be trough this module. The figure 3.2 provides a representative scheme of the back-end and is an overview of this client-server architecture.

### 3.3.2 Web Application

The Social Sensing project provides a web-based application that uses a web browser as a client, improving the portability of the project. The SSWA has graphical presentation of a cyclist fitness ride and route environment data. The figure 3.3 illustrates the site map of the SSWA. As is possible to observe this application is divided in two sections as refereed in section 3.2 for the Web user perspective.

\(^4\)http://www.w3.org/TR/ws-arch/#id2260892 - accessed in September of 2013
The public area includes a presentation page which helps the guest users to understand the main characteristic of the mobile application, such as start screens, team edition, ride information displayed and track team functionality. It is also in this section the user is able to register or login to the Social Sensing project. To register the user simply needs an email and a password. We do not ask more information, since would be too much time consuming and susceptible to human errors that causes the user to fill out the wrong details, reset and refill again, or worse leave the application without trying it. We follow a “plug and play” approach, if the user successfully registers, is automatically accepted without administrator confirmation. The private section, or login section, shows the full functionalities of the SSWA. The user can navigate through his past workouts and perform a critical analyses to the results or edit his profile with more complete information.

It is also possible to plan in advance the ride trip using our crowd-sensed environmental data, which provide rich information about the health quality of a given route, in terms of noise pollution, road quality and route steepness, and in a next phase with user comments. If the user enjoys cycling with friends, he can initiate a team with one click, add users to the team or delete. If the team is a professional team and possess a coach, he can remotely track the team location in near real time using the web application. This functionality is also available for any type of team, in case the user leaves home late, for example, and desires to know where the team already is.
3.3.3 Mobile Application

The SSMA is the most important part of the project, since most of the user interaction will be done with it. One of the project goals for the system is for the smartphones explore resources from other devices using a distributed solution. Thus, the application will be introduced to an environment of sparse connectivity and uncontrolled mobility. Given this characteristics, the SSMA needs a decoupled architecture and since the Halo system, as seen in section 2.3, has a centralized scheme leaves it out of the possible solutions. Also the protocol to be used must be fast and light. Once the Quintet uses a very strict protocol that drops many nodes if not every condition is achieved, this leads to less amount of available nodes to perform the desired task, and incrementing the delay of the task. On the contrary, the PhoneSensing protocol is lighter and tries to minimize the delay, by flooding the network.

For the aforementioned reasons the PSF is the one that better fits the application requirements and will be the one used in the SSMA.

Another advantage of this framework, is the fact that was developed at Instituto Superior Técnico. This allows gathering the knowledge of the persons involved in the project in case of issues.

Before detailing the proposed architecture one needs to explain the PSF on all its components. This provides an understating on how this framework can be used to correctly address the requirements as well as its limitations.

PhoneSensing Framework

The PSF allows the devices to explore sensors from other available devices. This subsection will explain the architecture of the framework and the wireless technologies available to implement the protocol.

The figure 3.4 illustrate an overview of the PSF modules. The modules works as follow: A request arrives to the Sensor Request Manager module asking for a sensor, for instance the light sensor. This module will then contact the Sensors module to inquire if the device can handle the request, otherwise contacts the Opportunistic Sensor Manager that will contact the Radio Interface. The Radio interface then contacts other devices using the protocol described in section 2.3. When a reply arrives to the Radio Interface it will forward to the Opportunistic Sensor Manager, which will then send to the Sensor Request Manager to process the data. Next we will proceed to an explanation of each module.

Sensor Request Manager The Sensor Request Manager works in an asynchronous way and is very much alike the Sensor Module. It is responsible for handling arriving and outgoing requests from and to remote devices. As requests are taken they are issued and queued. The queue uses a First In First Out (FIFO) agenda. The outgoing requests are handled in parallel with the incoming requests. The only simultaneous resource they share is the Bluetooth device, it does not allow concurrent connections. When an outgoing request has been queued, no more outgoing requests are accepted. By doing this, the framework ensures that duplicate requests are not emitted, preventing flooding of the surroundings with protocol messages.

Sensor Module This module is a service that runs in background. The module registers the number of sensors available on the device. The module use handlers to manage sensor requests. Because
handlers are asynchronous the Sensor module is able to handle a sensor request from the Sensor Request Manager. The requests are taken through service messages.

The SSSMA will need to interact with several sensors in order to provide data about the Cyclist Performance and environment mapping. Since the PhoneSensing only has the light sensor registered this is a severe limitation to the application that will need to be corrected.

**Radio Interface** The communication between the neighboring devices can be achieved using the Bluetooth (also known as IEEE 802.15.1 standard\(^5\)) or with the Wi-Fi Direct\(^6\). These are the most used\(^7\) wireless technology in the smartphones for Peer-to-Peer communications. The Near Field Communication\(^8\) is not address in the project, due to its short range (typically requiring a distance of 4cm or less to initiate a connection).

The Wi-Fi Direct devices support the same performance profiles of regular Wi-Fi devices\(^1\). The data rates can go above the 250 Mbps, and for devices based on 802.11 a or g, data rates will be about 54 Mbps and a coverage range of about 100 meters. This network can act as a one-to-one or one-to-many topology.

The wireless IEEE 802.15.1 standard is an ad hoc, terrestrial, wireless standard for short range communication\(^3\). This technology has low power consumption, is widely use in the smartphone industry and operates with three different classes of devices: Class 1, 2, and 3 where the range is about 100 meters, 10 meters and 1 meter respectively. The operation band is 2.4GHz and the data rates can reach between 1–3 Mbit/s. The Bluetooth network usually is constituted by one master and 7 slaves (piconet).

Therefore the Wi-Fi direct provides higher range and data rates than Bluetooth, but has a higher power consumption and is only available in Android 4.0 or later devices with the appropriate hardware and support to older products will be up to operating systems and chipset makers.

In conclusion, we selected Bluetooth instead of Wi-Fi Direct for two very practical reasons: (1) we want to maximize the smartphone battery lifetime, (2) Bluetooth is present in (almost) every smartphone

---

\(^8\)http://www.nearfieldcommunication.org/ - accessed in September of 2013
and (3) the drivers of the devices used to implement the application did not support the Wi-Fi Direct, thus the Bluetooth was chosen as the wireless technology to be used. Once the system is working and the drivers support the Wi-Fi Direct technology, more powerful transmission techniques can be implemented and tested.

Therefore this module is responsible to handle the incoming and outgoing Bluetooth connections. It operates both as a client and server. It assures that only one connection is established at any given moment. In order to assure an effective scheduling process, the interface of this module guarantees that when a connection is established a lock is enabled. This is a limitation, since only one connection can be established. The locks are gained with a message lock request through service messages. Beyond handling the Bluetooth access scheme, the module also determines what application module to message the information after receiving an incoming connection from another device.

This Radio Interface is not able to detect the devices that are running the framework, thus it needs two phases to know the PhoneSensing devices. This solution offers a delay, when one tries to discover which devices are running the Social Sensing Application and it is desirable that the users wait as less time as possible. Another limitation of the PhoneSensing is the fact the data gathered is not saved, which does not comply with the "Disconnected Operation" requirement. The application should be able to save the user sensor data along the ride.

Mobile application Architecture

It is using the SSMA that the user will collect and upload fitness and environment data. This data can be later uploaded to the back-end to be summarized and visualized using either the mobile application or the web client application. The figure 3.5 shows the SSMA modules.

As mentioned before the application will use the PSF. This framework will be responsible for accomplishing the application requirements, such as resources sharing between the team members and provide a scheme of concurrency control for this requests. Since the framework was not programmed to save sensor data, one needs to implement a module responsible for that. Thus the characterization of the modules will start by this module and will end by describing the visualization modules.
Local storage This component stores the collected data after it has been processed. As the raw data is treated, it is discarded. This is imperative to address the incorruptibility of the data and the privacy of the cyclist. Also none of the raw information is uploaded to the back-end.

Application Core This module acts as the brain of the application. It is responsible for orchestrating the user input and configurations. For instance, the user desires to constitute a team using Bluetooth, the GUI sends a message to the core module, which will send a message to PhoneSensing to discover other devices running an instance of the same application. When the PhoneSensing finishes the Bluetooth discovery, sends a message back to the Application Core, which will take the appropriate decision, in this case will generate a pop up with the name of the devices discovered. This block is purely representative since implementation wise most of the code is divided up into independent classes so scalability is not an issue.

Audio Client As mentioned in the section 3.3.3, the PhoneSensing had very few sensors registered, this is a limitation to the mobile application, because it is a requirement to infer the cyclist environment ride conditions. Thus the application extends the PhoneSensing module by registering the microphone sensor to it.

The microphone is probably the most common sensor available in an Android device. This module works in background and is responsible for measuring the energy of the recorded signal. The low energy level reveal silence or unacceptable phone context (e.g., inside a backpack, pocket, etc), while the opposite may revel traffic or other sources of noise pollution.

Inclination Client The inclination module is an extension of the PSF. This module will be responsible for determining the inclination during the user ride. It operates in background and is responsible for retrieve and process information of sensors\(^9\), such as accelerometer, magnetic or orientation to provide information about the pitch, expressed in percentage. The pitch indicates the inclination of the measured surface to a horizontal plane.

Internet Manager This module is in charge of handling the connections to the back-end. It does so in an asynchronous fashion. It uploads data from the local storage, and other primitives, such as user current location, fitness metrics, and environment data (e.g., steepness and sound pressure level). As soon as the data is uploaded, the connection is tear down. It may also download data if required (e.g., team formation, team location, ect). It is using this module the user is able to register or login.

Social client The social module allows the user to share his information in the SNs. This module is responsible for allowing the user to register/login and share data using a SN API.

Location provider Knowing the location of the user, allows the application to provide superior information to the user. This location can be either absolute or relative. The absolute location refers to the positioning coordinates of a receiver at an “unidentified” point with respect to the earth’s reference frame by involving the consultation of one, or more, dependable beacons or servers. The most usual of these systems is the Global Position System (GPS) \[19\][6]. The relative location describes those systems that allow the location of a node to be obtained, relative to other surrounding nodes. Thus, the two main functions of this module are: (1) to obtain the absolute location of the user; (2) obtain the user relative

location. Furthermore, if some nodes may have absolute geographical references, they can be used as anchors, so that relative coordinates may be converted to absolute ones.

To obtain the absolute location, we can use GPS and the Android’s Network Location Provider. The location providers supply at fixed intervals information on the geographical location of the device. The three providers’ classes will be introduced by order of decreasing accuracy.

The most accurate provider requires GPS hardware. For the GPS receiver to work it needs to be in line of sight of at least three satellites to obtain a 2D position (latitude and longitude) and track movement. If more satellites are available, the GPS receiver can calculate the user’s 3D position (latitude, longitude and altitude). Since the GPS receiver needs a clear path to a GPS satellite, they may have issues working properly in places where the sky is not visible, also it may take a substantial amount of time to lock the signal of at least three satellites. Another huge drawback of the GPS provider is it can consume more battery power than the others location providers.

The Wi-Fi provider, requires the smartphone to support Wi-Fi technology. It provides location information using the strength of the receiving signal of active wireless network access points (APs), and then query’s Google Location service, which later provide the location info. This solution offers several limitations, such as: the device needs to be in range of Wi-Fi networks; the networks should possess a publicly broadcasted service set identifier (SSID) well configured, otherwise it may be ignored by android. In addition, if the AP changes location, it may originate errors in the location data provided, since the Google Location service needs to be updated with the new location.

The cellular network location mechanism is akin to the Wi-Fi, the mobile device needs to be in range of one or more base transceiver stations (BTSs). The device measures the received signal power, and knowing the unique identification of the BTS the device is connected to or the past BTSs it has used, he sends this information to the Google Location service. Upon receiving the request the Google service updated the user of his location. The cellular network suffers from the same limitations as the Wi-Fi networks, which is the Google Location service must be aware of the cell towers IDs and their updated location. Although the probability of a BTS to change its locations is very low.

**Map Client** This module will allow the user to explore his location embed into a map with rich information, such as tracked points plotted or customs markers. Although there are several map clients, such as Bing or OpenStreetMap, Google Maps is by far the most used map client by the Android developer community, also is the one that offers better API information. Also since it is widely used, is easier to find solution for possible issues. Given these facts, the project will use the Google Maps API V2.

**Quick Action** This module is a personalized Android action bar, which provides quick actions and

---

14 [https://support.google.com/maps/answer/1725632?hl=en](https://support.google.com/maps/answer/1725632?hl=en) - accessed in September of 2013
guiding modes to the user. It is common to all screens after a successful login. Here the user can perform quick actions such as:

- Navigating Up with the App Icon - clicking on the application icon, the user navigates in the application based on the hierarchical relationships between screens. For example, if screen A displays an index of selectable actions, that upon selection leads the user to a screen B. By clinking in the action bar icon, the user goes back to the screen A.

- Requests - the user can manually perform a set of requests, such as require the GPS, accelerometer, microphone data, team location, between others.

- Team - this action allows the user to easily set up a team using the Bluetooth, or to download the Team list that he is part of. Also allows the user to select a team, which is already stored in the smartphone.

- Log out - the “Log out” option allows the user to delete all information on the SSMA, providing more privacy and security to the data. For instance in case of theft, the robber cannot have access to the user sensitive data.

Ride Information The fitness metrics and other services such as location, needs to be shown in an intuitive way. Using the information gathered by the application, this module will display useful information to the user. The usefulness of the information is decided by the user, since he can select what is shown.

Team Information This module will display rich information about the user current team, such as the elements names. Also allows editing / deleting the team in the device.

Profile/Settings The mobile application allows the users to enable and disable the sensing modalities available on the phone, such as audio, accelerometer, Bluetooth, GPS etc. With these functionalities the user is able to decide what portion of their presence is shared with his team or the cycling community. Thus, the user is able to customize his cycling experience by choosing what is more important for him, and the application is able to remind each configuration for their respective profile. This is important, since the user can settle once the different profiles for his rides. For instance, if the user is going to cycle in the city he may want a different settings configuration, than when going to the country side, the same can be said about when cycling with friends or alone. And most important, the user only needs to configure the settings once, and every time he returns to the application, they will be there.

3.3.4 Important Algorithms

There is a strong social element attached to the cycling activity, with many bikers often riding in swarm. If all the cyclists in the swarm run the SSMA, some benefits can be exploited such as energy-saving and resource-sharing. In order to leverage the opportunities made available in this cooperative ambiance, the mobile application allows users to set up teams, which can be done either on the SSWA or on the mobile using the Bluetooth technology.

19Unless the user deletes the information.
The Android API does not have a method to perform Bluetooth Broadcast, for which it will take a lot of time to do a leader election algorithm in the team formation process. This is the principal reason why it is required that the user who has the most mobile resources of the team start the team formation process. This process will start when the application core module receives a request from the action bar module. The algorithm that execute this tasks is illustrated in the figure 3.6.

![Figure 3.6: Team formation process.](image)

As we see in the figure 3.6, the team formation process is divided in three stages: (1) Discovering, (2) Filtering, (3) Capability request.

When the process starts, it confirms if the discovery process has not already started, if not then the device begins to search all the other devices nearby. This is done to avoid duplicate messages.

After this discovering stage, it comes to the filtering stage. The device will automatically choose those that run an instance of the PSF, as described in 4.5. Then the outcome with the selected devices will be shown to the user, who will decide both his team members and a team name. After that, the leader device itself will make a capability request to the selected devices about the emails, sensors and Internet.

Having received this request, the selected devices will not be able to activate the Bluetooth functionality for a short period of time. And the users will get a notification about that immediately. Once a device is involved in a team formation process, it cannot start another one at the same time. Meanwhile, the leader device will save all the collected information of the team devices in the SQLite database. The leader device will then share this information with the rest of the team. Everyone then, can me aware of what resources are available in their team. In case the team leader does not have internet connection, the application will work on delay tolerant mode. This is, the swarm of cyclists goes on trip, gather sensed data, and the team leader uploads the new formed team when he returns home. When a device is leader of a team (and the team is selected), his Bluetooth name changes to “Leader” followed by the team name.

When he changes of team, or restart the application, the Bluetooth device name return to the original one. This is done to facilitate the discovery of the team leader, since the function to add new users is only available to the leader. For instance, if another user arrives after the team is already formed, the user can select the action “Join team”. This will send a request to the application core, and then proceeds to the step one and two of the team formation. But instead of filtering devices that run the application, will filter
devices which their Bluetooth name starts by: "Leader" followed by the team name. This information is then processed and showed in a more presentable way to the user, who will select the team he desires to be part of. After selecting the team, the device sends a request together with his capacities to the leader of this team.

Upon receiving the request, the device shows a message to the user indicating that the user "X" wishes to join the team. If the user rejects, the process dies. If he accepts the device will add the new member information to the database and contact all the devices over Bluetooth to update all the team members, including the newly joined. Assuring that everyone is aware of the new team update. The drawback of this approach is that if a member of the team is not in range to the team leader, he will not be aware of the newly joined member. Also since the back-end does not start a communication with the devices, there is no way to warn those devices. In the future we expect the team leader device to maintain a record of the devices that have not been updated of the team members/capacities table, so when he contacts a devices which is on the table, will proceed to an update. Also we expect the back-end to start connections when requested by the team leader, assuring consistency of the shared information.

Another important algorithm is the one responsible to run the distance estimation, and works as follow:

Figure 3.7: Measure distance algorithm.
To start discovering devices, one simply calls the method "startDiscovery()". This is an asynchronous process and will return a boolean indicating whether discovery has successfully started. The application core will use an asynchronous task to filter the devices and will contact the devices with the request. The numbers of devices contacted are saved, so the application knows the number of replies expected to receive in order to proceed to the next step.

Since the devices may go out of range from the requester device and the application is time sensitive, a thread in background is executed which will then run for a limited period of time (e.g. 2 minutes, since a bigger delay of time would make this process slow and when tested with 2 devices, both of them replied in less than 1 minute.) in order to assure the continuity of the process.

The algorithm than runs on the remote team devices when they receive a request to infer the distance they are from their peers, follows the same logic as the algorithm 3.7, with exception that they do not forward the request to other devices, but reply back with the distance values. This way is possible to prevent the devices that already received the request to receive the same request again, flooding the network. Also this allows shortening the time to execute the algorithm. The image 3.8 shows the second part of this algorithm, when the device receives the replies.

When the location provider finish running the algorithm responsible for the distance estimation, it proceed to the spatial placement algorithm to produce a 2D map of where peer devices are.
Chapter 4

Social Sensing Implementation

This chapter describes the implementation of the SSP. This system accomplishes the functional and non-functional requirements of this project, as for example, provide a service with added value to the group of cyclist.

The SSP is divided in three main parts, the SSMA, the SSWA and the back-end. The first section will talk about the resources used and the second will describe the implementation of the back-end. The third section talks about the SSWA, while the fourth is going to describe improvements done to the PSF. The last section defines the implementation of the SSMA functionalities.

For the time being, the AOS is the only operating system the mobile application will run on, due to two reasons: (1) most cycling users, use the Android system (see 2.7), and (2) only Android devices where available. Also the mobile application will be developed in Java\(^1\), since the Android Bluetooth API is only available for native applications, the PSF is also written in Java, and native applications\(^2\) \(^3\) have better performance than web applications[21][31]. The drawback of this choice is that the application will be restricted to the Android platform, although it is the most used one.

4.1 Hardware

As an Android application, it will be designed to interface with the hardware present on Android phones. Theoretically the application can also be run by other devices which can emulate the Android, but this will not be a consideration during the design. The devices used during the implementation of this thesis, was the smartphone Sapo A5\(^4\). This smartphone is based on the ZTE Blade\(^5\) and is equipped with a LCD 3.5" with 800x400 pixels of resolution, a capacitive touch screen, 512MB of RAM, GPS, accelerometer, proximity sensor, Wi-Fi 802.11b/g, Bluetooth (v2.1).

The Sapo A5 contains a modest 600MHz processor, which is enough to run the desired functionalities of the Social application. The original operating system was the Android Froyo 2.1.

\(^1\)http://www.oracle.com/technetwork/java/index.html - accessed in September of 2013
\(^3\)http://sealedabstract.com/rants/why-mobile-web-apps-are-slow/ - accessed in September of 2013
\(^4\)http://a5.sapo.pt/- accessed in September of 2013
For the PSF to work properly it was necessary to always be in Bluetooth discoverable mode, so that neighboring devices could discover them and try to establish a connection. This functionality is only available in the more recent Android Versions, thus we had to change the Android version installed on the devices. Flash the low-level firmware of the Android devices gives the user the ability to install another versions of this operating system. The process of flashing the firmware of the phone has some challenges, since we need to confirm the right firmware and the integrity of the files, otherwise the process may brick the devices. One needs to flash part of the low-level firmware on the devices. For our devices we located a firmware and the AOS from the xda-developers\textsuperscript{6} forum. After some attempts we did manage to install the necessary programs on the phone so that we may have a more updated Android version. The back-end are IEEE 802.3\textsuperscript{7} connected servers equipped with practically unlimited storage and computation power.

4.2 Development Environment

We chose the integrated development environment (IDE) Eclipse to build the application. Eclipse is an open-source IDE maintained by the Eclipse Foundation\textsuperscript{8} and is the IDE recommended by the Android Developer Guide\textsuperscript{9}. The Android Standard Development Kit (SDK) is available through an Eclipse plug-in named Android Development Tools (ADT). The Android SDK is a compilation of all the classes and components that are used by the Android Framework, and it is essential to write Android Application. Also provides essential tools, such as the Android Debug Bridge (ADB), which handles connections between the computer and the real Android device.

4.3 Back-end

One of the biggest problems in software development is the integration between the applications that were built over different platforms and/or different languages, where these applications should communicate between each other in order to exchange data. To tackle this problem, we proposed a technology that goes by the name of Web Service. This Web Service was developed using Hypertext Preprocessor (PHP)\textsuperscript{10}. The PHP\textsuperscript{11} is a simple and powerful language which provides high compatibility with leading operating systems and web servers, thus allowing it to be deployed through several different platforms\textsuperscript{23}. It is designed to perform well with the web, so methods like accessing the GET and POST and working with HyperText Markup Language (HTML) and Uniform Resource Locators (URLs) are built-ins in the PHP. The data exchange format chosen was JSON. This data format\textsuperscript{12, 13} is based on a subset of the

\textsuperscript{6}forum.xda-developers.com - accessed in September of 2013
\textsuperscript{7}http://www.ieee802.org/3/ - accessed in September of 2013
\textsuperscript{8}http://www.eclipse.org/ - accessed in September of 2013
\textsuperscript{9}http://developer.android.com/training/basics/firstapp/creating-project.html - accessed in September of 2013
\textsuperscript{10}http://php.net/ - accessed in September of 2013
\textsuperscript{12}http://www.linux-mag.com/id/7717/ - accessed in September of 2013
\textsuperscript{13}http://www.vogella.com/articles/JSON/article.html - accessed in September of 2013
JavaScript language and as XML, is a format designed to organize data. JSON is more lightweight than XML, allowing a faster parsing and reduce bandwidth necessary, which is useful for mobile applications. Also to work with JSON in PHP is very straightforward, to encode we have `json_encode()`, and decode: `json_decode()`. This section is intended to explain the implementation of the Web Service.

**4.3.1 Database connector**

This module is a PHP script that holds all database information. This file is included in every php script that interacts with the database, using the defined variables.

This file is responsible to assure a secure authentication between the Web Service and the MySQL database. The file uses PHP Data Objects\(^\text{14}\) (PDO) extension for accessing the database, this way is possible to provide a data-access abstraction layer, which means that, regardless of which database we are using, we use the same functions to issue queries and fetch data. This abstraction layer hides the interaction with the database, by supplying all functionality through an API. In another words, the connection to the database are through a separate layer that handles all the complexity and processing.

Another important aspect of PDO is the support for prepared statements. When we use prepared statements and parameterized queries is possible to prevent SQL injections, since the SQL statements are sent to and parsed by the database server separately from any parameters.

**4.3.2 Web Service APIs**

The Web Service will handle requests from two different clients, the SSMA and the SSWA. Although the PHP files that handle this requests are in different folders for code simplification and security, they share some similarities.

The Web Client is constituted with several PHP scripts that will handle all requests from and to the SSMA or SSWA and interaction with the Database connector. The back-end does not start a request to the mobile client, it only replies. As mentioned all communication between this module and the mobile application are done using JSON. The scripts present in this module are:

- **register** - this script is responsible for registering the user;
- **login** - will authenticate the user;
- **get teams** - retrieves the teams the user is registered and replies back to the user;
- **get team and capabilities** - retrieves the email of each user registered in the team, and their devices capacities. The reply is an JSON array with: (1) array of the users,(2) array with devices capabilities;
- **add team** - allows the user to add teams, for instance: teams created with the Bluetooth technology or update an already registered team. The team is added with their respective user's email;


38
- **get team location** - This script provides the user with the last known location of each team element. The information is sent over an array, with their location, stamped with the time to provide context, and environment conditions (slope and pollution noise);

- **insert location to share** - in case the user wants to share his information with his friends, the information is sent to this script. This information include his location, time of upload and environmental conditions;

- **fitness metrics** - is responsible for handling the user fitness metrics. Includes: average speed, maximum speed, distance, calories lost, duration and date of upload. It also includes a flag indicating the id of the user, and if the ride was done within a team, the team id;

- **environment** - this script will handle the data correspondent to the fitness metrics. In another words this script will handle all the location and environment conditions presented during the cyclist ride;

- **route pollution noise** - show the user all the routes travelled by the community, with the respective noise pollution;

- **route steepness** - show the user all the routes travelled by the community, with the respective steepness;

In case the user authentication, the process is handle with extra security. For instance the registry, before the user’s registration, the fist action performed is to check the existence of the inserted email, if already exist an error message is sent to the SSMA or SSWA indicating the problem. In case the email is does not exist, the process works as illustrated in the figure 4.1.

![Registration process diagram](image)

**Figure 4.1:** Diagram representing the registry process.

A unique and random word is generated then, using the PHP function uniqid(). This function gets a prefixed unique identifier based on the current time in microseconds. We hash this random word to create a random salt that is added to the hashed password and finally both are hashed again using SHA512 encryption.
This methodology avoids the use of rainbow tables, precomputed table for reversing cryptographic hash functions. In case an error occurs during this process, will cancel the registration, and send a message back to the user, warning about the error.

The web client authentication works as the client, see figure 4.1. But in this case the messages are forward to the web application and not to the mobile. The client also has scripts to handle some specifies of the front-end code, which will be described in more detail in section 4.4.

4.4 Web Application

The Social Sensing project provides a web application for the user to analyze and visualize fitness and environmental data. This front-end is a mix of different languages which together are able to provide important services to the cyclist. The design or structure of the website are done using HTML5 and Cascading Style Sheets 3 (CSS3). These are markup languages used to structure tags that describe the document content. There were other options beyond the HTML. One of the other possible options was the Flash from Adobe Systems, but Flash code requires a Flash player installed being a drawback for the aspirations of portability. If HTML specifies how the browsers should display a web page, the CSS is responsible for specifying the guideline of how a browser will present the HTML code. This is important since the HTML code supported by the CSS is read in the same way by the different browsers, protecting the portability and versatility of the system.

To make the front-end more dynamic we used jQuery. JQuery is a javascript library that greatly simplifies JavaScript programming. The jQuery is mostly used to handle things like HTML document traversal and manipulation, event handling and animation. For instance the button event handling to show image illustrating noise pollution is done using jQuery.

The SSWA provide rich map information for the user analyze and visualize his fitness metrics. To do this, one needs first to register to the google APIs Console, after registering for the new version of Google Maps API v3 service, the key provided is added to the Google Map Api application. To use the map service we do the following steps:

- include the Maps API JavaScript using a script tag;
- create a div element named "map-canvas" to hold the Map;
- create a JavaScript object literal to hold a number of map properties;
- write a JavaScript function to create a "map" object;

---

15http://www.w3.org/TR/html5/ - accessed in September of 2013
16http://www.w3.org/TR/css3-fonts/ - accessed in September of 2013
After finishing those steps, the map needs to display data on it. The MySQL is used to store the user data, for instance his geographical location. Although in this document the back-end and the front-end are introduced separately, a small piece of PHP code is added to the front-end. This PHP code is responsible for retrieving data from the MySQL database, and printing an output with the name and arguments of a JavaScript function inside the Google Maps service. This output will add markers to the map together with arguments (e.g., latitude; longitude; date and fitness metrics).

It is also possible to see a trace of the user’s previous locations and a line that joins the coordinate points can also be seen. To do so, the coordinate values are passed when instantiating new polygon geometry objects. Then the coordinate values is passed to the new polygon’s path and added to the map. The figure 4.2, illustrates an user ride with several markers. Each marker refers to a space in time where the user registered his geographical location.

![figure 4.2: Map showing the ride of a user.](image)

The Social Sensing project is also a Crowdsensing platform[24][32]. Beyond the personal sensing, where the cyclist focuses on personal monitoring, the project also has public sensing. Public sensing is the type of data that is shared with all users for a common good. Since the cyclists already use the smartphone to monitor their performance, we leverage these devices by building large-scale sensing application efficiently (cost and time). For instance, alternatively to installing roadside cameras to gather traffic data and detect congestion level, one can use the microphone on the cyclist smartphone to achieve the same goal. Such solution minimizes the cost of deployment of specific sensing foundation.

After the cyclist uploaded the environmental data stamped with time and location for context purposes, the front-end map system makes visualization, interpretation and sharing of the route information. To make data interpretation easy we added a layer to the map23. Layers are objects on the map that include reside in one or more separate items, but are handled as a single unit. Usually they reflect group of objects that are added on top of a map to designate a common organization. We opted for using a “Heatmap” layer. A “Heatmap” is a visualization used to characterize intensity of data at location points. When this layer is on, a colored overlay will be shown on top of the map, with areas with more intensity appearing red and less intensity appearing green.

The information requests are done using AJAX (acronym for Asynchronous JavaScript and JSON).

AJAJ is similar to AJAX\(^24\) (acronym for Asynchronous JavaScript and XML), is executed asynchronously, so it does not compromise the flow of the JavaScript interpreter as it gets read\(^[4]\). Every time the user scrolls the map, the front-end will asynchronously send a request to the back-end, which will encode using JSON. When the front-end receives the reply, the JavaScript will decode (ex: JSON.parse(request.responseText)) the response and update (refresh) the map.

In the SSWA the user can also create or edit his team. To add users, they must be logged in. In the future the user will be able to send social network invitations for his friends to join the project. The application also allows the user to track the last known location of his team mates, using the map method described above. This information is only available for the team members. Finally the user can also see the team metrics (e.g., max speed, average speed, calories, distance, duration) but for all the team. This information is shown in a table in the form of average and medium.

### 4.5 Extending PhoneSensing Framework

The first thing done, after flashing the smartphones was to deploy the PhoneSensing to the new version of OS. After correcting several bugs, and changing the deprecated\(^25\) elements (e.g., PreferenceActivity\(^26\), addPreferencesFromResource\(^27\)), several changes had to be done in the code and design structure, which will be describe in this section.

The main problems faced when integrating the Framework, was:

- The framework was very poorly commented;
- Bluetooth was not always enabled;
- PhoneSensing Discover Protocol was slow;
- The PhoneSensing was done to send sensor data over Bluetooth;

Therefore in order to use this framework we needed to extends and optimize his capabilities. We started by retrieving the modules responsible for handling the extension of the Bluetooth discoverable mode. After removing the modules we change the Bluetooth application request to extend the discoverable mode to “always”. With this change the user will not be prompted every 300s to allow the extension of the discoverable mode.

Next we proceed to correct the time it takes to discover devices running an instance of the PSF. When the Bluetooth discovery process happens, the devices nearby were not filtered, this is, it was not possible to know if they were devices running the framework or not. So the PhoneSensing discovery process was divided in two process: (1) Discovery, (2) Filter. The discovery process discover all devices in the neighborhood, while the filter process connects to a specific port, since all the devices running this framework listens on that same port. This type of connection is denominated InsecureRFCOMM

\(^{24}\)http://www.w3schools.com/ajax/ajax_intro.asp - accessed in September of 2013
connection and also allows the device to connect to another without pairing. This is quite useful since the device does not need to accept or refuse the connection.

Thus to improve this discovery process, we used a method available since the Android API 10. This method allows to discover services offered by the devices, based on their Universally Unique Identifier (UUID). This way is not necessary a filter process, because one would only discover the devices that run the PhoneSensing framework.

We proposed to used the method createInsecureRfcommSocketToServiceRecord, that has the same functionality as the InsecureRFCOMM, but allow to connect to a specific UUID. Unfortunately the method had compatible issues between the Android version 4.0 and upper and the Bluetooth firmware version (2.1) of the devices used. This was a hard bug to discover since the error shown was: "IOException: Connection Refused", giving the perception the error was in the application code and not an compatibility issue.

The solutions available to correct this compatible issue were: downgrade the current AOS version to below 4.0 or maintain the current solution used by PhoneSensing. To put in another words, the choice is between having the Bluetooth mode always discoverable or a slower discover of devices running the framework. Because the application intends to be the most autonomous possible from the user, and it is not feasible to have a cyclist interrupting his ride every 300 seconds to extends the discoverable time, we decided it was better to keep using the old method used in the PhoneSensing, InsecureRFCOMM.

The delay introduced by the filter process will not considerably affect the performance of the application, since most of the time the user will have his team already formed, so the device will already know the MAC address of the devices to interact.

Another issue, is that the Social sensing will send more than sensor data, it will initiate team requests, will exchange team capabilities and share other resources, such as internet. Although the protocol used was the same as described in the subsection 2.3, some changes needs to be done. For instance, when the user starts a team request, the device will need to be able to receive more information than just from one device. And since the information to be send is not a string, it needs to be serializable. Also, if the process dies, that is to say, if the device is not able to receive any information the user should be informed of the process state, or at least the process should not die, but should try to minimize the situation by giving other information in the same context. This information will be better described in section 4.6.

Another necessary information was to add new sensors to the library used to describe the capacities of the device. When the application was developed it only had the description of the light sensor and GPS, although the GPS method was not implemented. The library used is Resource Description Language (RDL)[31][27]. The RDL allows the devices to exchange objects in key-length-value (KLV) format between devices. The RDL fits very well into the sensor sharing subject because RDL gives attention to the description of their characteristics for interoperability, sharing and discovery. The RDL can be specified in two different formats, XML and KLV. The KLV format yields very low sizes which in a mobile

---


29The issue can be seen in the Android Open Source Project here:https://code.google.com/p/android/issues/detail?id=29039 - accessed in September of 2013
environment with short rendez-vous and low bandwidth connections can be very useful. The resources added to this library were: microphone sensor, accelerometer, and internet service.

4.5.1 Audio Client

It is secure to assume that all Android smartphones provide a microphone to the user. This microphone can also be used as a sensor to record audio and analyze the outcome of the recording. The SSMA uses the microphone to analyze the route noise pollution and security, since a high noise pollution in the route may indicate presence of cars nearby. This data, after processed is then shared with everyone for the public good.

As mentioned before, the Audio Client is implemented in the PhoneSensing module, namely the SensorService block, which is an Android Service class. This module runs in background, but since it runs in the main thread of the hosting process any high intensive CPU tasks will interfere in the application normal behavior, thus we record the audio asynchronously.

The android SDK provides two different components to collect audio data: (1) MediaRecorder and (2) AudioRecorder. The MediaRecorder is simpler to implement since it takes care of most of the recording details, such as obtain audio data and measuring the maximum amplitude. The AudioRecorder is much more complex to use, because the developer allows more flexibility, such as set the rate of audio quality, choose encoding format and channel configuration.

The SSMA wants to determine sustained high amplitude and not just the maximum amplitude, since it leads to a more robust analysis against individual periods of high amplitude. Because the AudioRecorder is more flexible, we can apply different methods of signal processing algorithms to the uncompressed audio bytes, such as to determine the energy of the frame. For the aforementioned reasons we choose to use the AudioRecorder component. The first think to analyze are the following questions:

- What is the best sampling and encoding?
- What should be the duration of the Audio clip recorded?
- How to avoid buffer overflows?

The audio formats allowed by the Android API are the encoding PCM of 8 and 16 bit. This specifies the length of each audio data byte. The Android documentation calls “frame” for each audio sample. The Android documentation advises the 44100Hz sample rate, since is that is guaranteed to work on all devices, but other rates such as 22050Hz, 16000Hz, and 11025Hz may work on some devices. The lower the sampling the lower the recording quality.

The audio channels accessible to record are Mono and Stereo. The Mono uses a single frame, while the stereo uses two, which doubles the audio data gathered. But only Mono is is guaranteed to work...

33 The format in which the audio data is represented.
34 Pulse-code modulation is the simplest form of digital source representation/coding where each sample is independently represented with the same number of bits. [11]
on all device. The mobile application uses a PCM of 16bit since it offers a bigger frame window that the 8bit. The sample rate choose was the 44100Hz and the audio channel was Mono, since these are assured to work in any Android device.

To avoid buffer overflows, the buffer size should not be small. The buffer size is in bytes, and is where the data is written to during recording. The Android SDK provides a method that calculates the minimum buffer size for these configurations. The application starts recording audio when the SensorService receives a requesting to record data. There are three types of requests: (1) remote devices requires; (2) explicit request; (3) calibration.

- remote devices requires - operation stands for one chunk, which after processed is then forward to the device that requested this service. Since the Bluetooth signal range is short, the pollution noise measured in the cyclist device that requests and the cyclist device that receives the requests, will be probably similar, we did not fund much utility in this functionality. Thus is not used in the SSMA. When a better context or more powerful wireless technology is supported, this functionality can be enabled;

- started ride request - The application will request periodically the Audio Client to collect data, this data is stamped with geographical location and time. Thus providing context to the data gathered;

- Calibration - Due to the fact each device microphone has different sensibility and the phone context may be different from situation to situation, the user is able to calibrate this sensor, assuring his has accurate measures.

Upon creation the AudionRecord object begins its associated buffer, which will be filled with the audio data recorded. Then the data is read from the microphone in chunks (temporary buffer) of length inferior to the total recording buffer size. We then use the root mean square (RMS) to compute the 'quadratic mean' value of the chunk, since it contains the peak value for each sample. The advantage of using root mean squared over finding the maximum value is that root mean squared takes into account all data points. This makes the calculation robust against single or short-lived time periods of high amplitude and allows only meaningful high amplitude signals to have an effect. We then proceed to calculate the sound pressure level\(^\text{35}\) (SPL), which is a logarithmic measure of the effective sound pressure of a sound relative to a reference value and it is measured in decibels (dB), is obtained using equation 4.1.

\[
spl = 20 \times \log_{10} \frac{RMS}{REF.VAL} + CAL.VAL
\]  
\[
(4.1)
\]

In this equation the \(RMS\) stands for route mean square, \(REF.VAL\) reference the sound pressure in the air. This value is commonly \(200\mu PA\), which is usually considered the threshold of human hearing. The \(CAL.VAL\) is the sensor calibration value inferred by the user. In case no calibration was done, this value is 0.

The next step, the module informs the SensorService that the operation has finished, saving the data in a temporary structure, together with location and time data.

### 4.5.2 Inclination Client

The characteristics of the cyclist’s fitness include the steepness of the road/trails that the cyclist covers during his ride. The slope of a route can provide useful information for the planning of a ride. For instance, the cyclist may prefers to ride the high slope at the beginning when his energy is at the top, and plain/downhill at the end of the ride, or just choose a slope that better fits his cyclist characteristics.

The Inclination Client starts gathering inclination sensor data when the PhoneSensing SensorService receives a request to read inclination, as described in the section 4.5.1. The data is processed using an asynchronous thread, and when the operation finish it sends the sensor data to the SensorService, that will be responsible for connecting with the GUI to show the sensor data. The slope of a given route is obtained using the equation 4.2.

\[
slope(\%) = 100 \times \tan \theta
\]  

(4.2)

The angle “theta” represents a slope from a ratio of a change of y-coordinates, also known as the rise, over a change in x-coordinates, known as the run\(^{36}\). A high percentage indicates a steeper degree of “tilt”.

Most Android devices have embedded several sensors\(^{37}\) able to measure motion, orientation, and several environmental conditions. These sensors are able to supply raw data with great precision and accuracy, and are able to infer the three-dimensional device movement or positioning. From those, the Motion Sensors\(^{38}\) are the ones capable of providing data to infer the steepness. These sensors can be provided by the device hardware (e.g., the accelerometer and gyroscope) or by the software-based (e.g., gravity, linear acceleration, and rotation vector sensors). Some of the software-based sensors may not be available since they rely on one or more hardware sensors to obtain their data.

The frame of reference we monitor the device movement can be divided in two: (1) relative to the device’s frame of reference or (2) relative to the world’s frame of reference. For instance, in the first situation the user is using the sensors to control a ball in a game, while in the second case the device is moving with the user while the user drives his motorcycle).

Given these characteristic, we are interested in monitoring the user frame of reference relative to the world’s frame of reference. The outcome of these Motion Sensors are multi-dimensional arrays of sensor values for each SensorEvent. For instance, the accelerometer sensor event returns acceleration force information for the three coordinate axes. To compute the inclination, we analyze three different ways of measuring the inclination. As a reference the gyroscope is not available in the smartphone used to test the application (see section 4.1). We started by registering the acceleration sensor to the sensor


46
event listener. Accordingly with the Android documentation the acceleration sensor uses about 10 times less power than the other motion sensors. This sensor measures the acceleration in three axes in \( m/s^2 \). The result is the acceleration applied to the device by computing the forces applied to the sensor. This measure is always influenced by the force of gravity.

\[
a_d = -g - \sum \frac{F}{m}
\]  

(4.3)

The \( a_d \) stands for the acceleration applied to the device, \( g \) for the force of the earth's gravity applied on the device, and \( m \) is the mass of the device. The sum (\( \sum \)) symbolizes the sum of the three axes (x, y and z). Thus when the device is sitting on the table, the accelerometer reads a value of \( g = 9.81 m/s^2 \). After retrieving the acceleration value, we computed its magnitude. The magnitude is obtained by computing the ‘quadratic mean’ value of the read data.

\[
a_{magn} = \sqrt{ax^2 + ay^2 + az^2}
\]  

(4.4)

Next we proceed to the normalization of the \( a_{magn} \) value. The inclination is the value of angle whose cosine is the specified normalization. Then we apply the equation 4.2 that provides the inclination measured. The second method was the most easy to implement, which is the Orientation sensor. This sensor allows the application to infer the position of a device relative to the earth’s frame of reference (particularly, magnetic north). This sensor computes data for the following three dimensions: (1) Azimuth, which is the angle between magnetic north and the device’s y axis; (2) Pitch, provides data about when positive z axis rotates toward the y axis. If y axis is negative the pitch is negative, if y axis is positive, the outcome is positive; (3) Roll, practically the same as the Pith, but the positive Z axis rotates towards the x axis. The final result is obtained by applying the equation 4.2 to the pitch.

Another solution is to use the acceleration sensor together with other sensors. The Geomagnetic Field Sensor provides information about the earth's magnetic field and gives raw field strength data (in \( \mu T \)) for each of the three coordinate axes. Thus the application uses the accelerometer and geomagnetic field sensor in combination with the getRotationMatrix() method, provided by the Android API, to obtain the rotation matrix and the inclination matrix. Next the social sensing application use these matrices with the getOrientation() and getInclination() methods (also provided by the Android API), to obtain azimuth and geomagnetic inclination data.

Finally we apply equation 4.2 to the geomagnetic inclination, and we obtain the pitch, or steepness in percentage.

To analyze the accuracy and the behavior of the sensors we put them stationary on a flat surface, such as a table and calibrate them. The Android offers four different rates to acquire the sensor rates: (1) normal, (2) UI, (3) game, (4) fastest. The android normal sensor rate is fast enough for this project.


goal. Furthermore a higher sampling rate would consume more power, something undesirable for the mobile context. Then we proceed to several experiments moving the device between zero and eighty degrees. The table 4.1 shows the measured values.

<table>
<thead>
<tr>
<th>Angle (°)</th>
<th>0</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>45</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real values (%)</td>
<td>0</td>
<td>17.63</td>
<td>36.40</td>
<td>57.74</td>
<td>83.91</td>
<td>100</td>
<td>119.18</td>
<td>173.21</td>
<td>274.75</td>
<td>567.13</td>
</tr>
<tr>
<td>First solution (%)</td>
<td>2</td>
<td>16</td>
<td>36</td>
<td>56</td>
<td>83</td>
<td>100</td>
<td>119</td>
<td>173</td>
<td>275</td>
<td>567</td>
</tr>
<tr>
<td>Second solution (%)</td>
<td>0</td>
<td>18</td>
<td>36</td>
<td>57</td>
<td>83</td>
<td>99</td>
<td>119</td>
<td>173</td>
<td>274</td>
<td>567</td>
</tr>
<tr>
<td>Third solution (%)</td>
<td>0</td>
<td>18</td>
<td>36</td>
<td>57</td>
<td>83</td>
<td>99</td>
<td>119</td>
<td>173</td>
<td>274</td>
<td>567</td>
</tr>
</tbody>
</table>

As one can observe all of them provide very accurate values. The last accurate among them is the first solution, that uses only the accelerometer sensors. The first solution offers several issues. The first is that, as said before, gravity and acceleration are measured together using just the acceleration sensor, thus by the "equivalence principle", the outcome of the measure will be the vector sum of gravity and acceleration. There are some ways around to exclude the gravity force, either by applying a high-pass filter which may introduce delay on the final readings or by reading the linear acceleration (not available in all devices). Either way, the z-axis may not be related with the up and down of the earth, thus the relation will depend on the orientation of the phone.

Therefore the second and the third solution are the best solutions. Because the orientationSensor is deprecated since Android version 2.2, the SSMA will use the third solution, combination of the Geomagnetic Field Sensor with the accelerometer sensor.

### 4.6 Mobile Application

When developing for a mobile device there are critical elements that need to be taken into account, such as target market, application features, design, performance and energy consumption. This application must comply with the requirements and design goals for the system and should not compromise the user experience.

This section will describe how the modules introduced in section 3.3.3 were implemented. It starts by explaining the UI, proceeding to the background operations in a top-down way. In chapter 3 the PSF is separated from the application implementation. When we improved the PSF we joined up together with the SSMA. That is to say, the mobile application has the framework built in. This choice is due to structural aspects related to the PSF development, this framework was then incorporated in the application.
4.6.1 Application Core

The application Core extends Android’s native Service\textsuperscript{41} class. This class operates in the background and is used to perform long-running tasks. Most of the functionalities will be this module’s responsibility. Also most of the requests from the GUI will go to this module. That will process it or forward to one of the PhoneSensing services. This service use handlers to send objects of Message\textsuperscript{42} type to the activity or the other services offered by the PSF. All tasks in this module are executed asynchronously. For instance, to filter the devices that belong to a team already registered, or to find which device in the team has a requested resource.

4.6.2 Local Storage

An important requirement for the SSMA is to save the cyclist data gathered during his ride. This data should persist beyond the ride it is gathered. The Android API\textsuperscript{43} has three different ways of achieving this:

- Preferences - Save private primitive data in key-value pairs;
- Internal Storage - Save private data on the device memory;
- External Storage - Save public data on the shared external storage;
- SQLite Databases - Save structured data in a private database.

The fastest way to save or access persistent data is by using "Preferences". It’s ideal to save small data, because is fast and the access is limited and straight forward. Thus the application will use this option to save user preferences, like how the screen should look or the settings menu is configured.

The "Internal" and "External" storage are the so-called "flatfile" approach. The application does not use this approach since as it scales and gets more and more data, this approach has issues. For one thing, searching for saved files are slower, especially in the "External" storage. And the most important reason, the application extracts information from the "flatfile" in a different manner from the way in which it is saved. For example parallelized reads/writes, quick searching, and easy reporting via the Structured Query Language (SQL) - is facilitated by changing our data from "flatfile" and keeping it in a relational database. Also the database are ideal for repeating or structured data, such as the cyclist ride. Therefore the database will be used to store the user login, and data related with his ride and teams.

The database recommended by the Android documentation is the SQLite\textsuperscript{44}. The SQLite is open source and has supports relational database features like SQL syntax, transactions and prepared statements and is embedded into every Android devices. But there are other options being the most famous among them: the LevelDB\textsuperscript{45}. The LevelDB is also Open Source and based on the concepts from Google’s BigTable\textsuperscript{46} database system and as such does not support relational features. The advantage

\textsuperscript{43}http://developer.android.com/guide/topics/data/data-storage.html - accessed in September of 2013
\textsuperscript{44}http://www.sqlite.org/ - accessed in September of 2013
\textsuperscript{45}https://code.google.com/p/leveldb/ - accessed in September of 2013
\textsuperscript{46}http://research.google.com/archive/bigtable.html - accessed in September of 2013
of using an entity relational database is that it provides a schema that is useful to catch errors and prevent invalid operations. Also the application will remotely connect with a MySQL database, becoming more easy to design the same scheme for both. Another advantage of the SQLite is, since it is recommended by the documentation it is the most used, providing good documentation and community support. Given these facts, the application will use the SQLite. The access to the database implicate accessing the file system. Since this access can be slow, the application perform database operations asynchronously.

The SQLite database is private to the application that makes it. But in the future this application can also be used by third party application. For instance the user may want to copy the data from the application to another that combines the data and offer an extra service. With this in mind, the application will use a "Content Provider" to manage the access to the database. The content provider encapsulate the data, and provide ways for characterizing data security.

4.6.3 Internet Manager

The SSMA offer real-time interaction between the back-end and the biker in support of real-time location. Also the user can collect sensed data and then upload’s it into his personal repository. This module is responsible for all communications with the back-end and operates in background in an asynchronous way, separate from the same thread that runs the GUI of the application. This is particularly important because connecting with the back-end may take some time, and can block the GUI or crash the application. This block works on requests. This is, he only uploads information when it is requested by the Application Core or the Action Bar. Every time a request arrives, the first action performed is to verify if the device has connectivity. If so the requested is uploaded, otherwise the action ends.

Reading and writing JSON are critical operations, since the server processes and creates JSON messages, so a fast JSON parser is needed. There are several JSON parser libraries available in the internet, but we decided to use one developed by Ralf Sternberg because is fast, minimal (read JSON into a simple Java representation and generate JSON from Java), no dependencies (just Java) and simple to use so we can modify it to our needs.

The module makes asynchronous HTTP requests, handle responses in anonymous callbacks, and as mentioned before, HTTP requests happen outside the UI thread. For instance, if the user wants to register to the application, this module will POST data to the URL responsible for handling the user registration, and retrieve the JSON response from the back-end. If the response is ‘1’, thus it was successful, otherwise if it was ‘0’, thus the user was not able to register, and the element "message" will be shown to the user, indicating the reason.

Since usually the internet connection is provided by a Telecom operator, and the user pays per Megabyte, the application tries to upload data in burst. For instance, if the user has several rides to upload, it uploads all at the same time.

Another example is when the user requests a team from the back-end, the process runs as represented in the diagram illustrated by the figure 4.3.

![Figure 4.3: User requests a team from the back-end.](image)

The user wants to ride with his friends, thus he requires the teams available on the back-end. The application starts by performing an HTTP POST to the back-end which will once reply with all the teams the user is part of. The application will parse the JSON and shows this information to the user, where he can decide the team he will cycle. After selecting his team, the application sends another HTTP POST to the back-end which will reply once with the users enrolled in the team, and their respective capacities. Once again the application parse the data, saves the teams and respective capacities in the SQLite and shows a message of success to the user.

### 4.6.4 Location Provider

The SSMA allow the cyclist to map his route and track his activity. This allows the user to know where he is going, where he has been and record his workouts. To do so, the application uses a Network Location Provider as described in 3.3.3.

Deciding which provider is the most suitable source of location data is a necessary compromise, since requesting updates from all of them would drain off battery. The cycling activity often occurs in places where the Wi-Fi access point is not in range, and although the cellular network provider consumes less energy than the GPS, it has much less accuracy. Since we decided to choose just one provider, the GPS also has the benefit of providing extra valuable accurate services, such as speed and distance data.

For the aforementioned reasons, there is no doubt that GPS is the most suitable location provider for a cycling application. As with any project targeted at mobile devices, energy consumption is a major concern [20]. If the battery lifetime of the device is severely affected, users will not use it. This raises the following questions:
- How often do we need updates?
- How accurate does it need to be?
- What is the impact in the battery life?
- What happens if location ‘jumps’?

The social sensing application must balance the energy used to sensing against the accuracy of the measures. If the location updates are too frequent the battery will be drained faster, on the other hand if they are too infrequent will increase the delay of the user location and accuracy. The social sensing accuracy tries to reduce the updates without significantly degrading the location and metrics accuracy by requesting GPS updates with a frequency of 2 minutes or a distance of 20 meters.

Before the application starts receiving updates it needs to register the Location Listener\(^{50}\) with the Location Manager\(^{51}\) to receive data about location, speed, distance, between others. This listener contains a group of callback methods which are called every time the device’s current location changes. When the application pauses or stops it register’s the location listener, since it does not need any more location updates, leading to an improve in the battery life. The setback is that the application will need to acquire the GPS signal again, which may take some time. This GPS information is saved and in case the map is in the foreground it sends information to the Ride Screen to update the user current location.

The map allows the user to have visual representation of the area. As mentioned before, we opted for using the Google Maps library\(^{52}\) to plot the user location. But it is not part of the standard Android SDK distribution, so we need to install and SDK ad-on and import the library to the project. Next we need to create a valid Google Maps API key. This key is obtained through the Google APIS Console, providing the application signature key and the application package name.

Using the methods provided by this library is possible to plot traces representing past user locations, points illustrating current user/team position, and add overlays to show the pollution noise and steepness.

If the user has selected to share his location and environment conditions with friends, this data is sent to the Upload Manager. If the user does not have internet, the application search the local database to find a team device with internet. If any is available, tries to connect, to send the data. The remote team device, if has internet forwards the data to the Upload Manager, otherwise says it is not possible to forward the data. But the mobile application provides more added value services to the cyclist, in special when he cycles in group. The cycling team should not be dependent of the GPS embedded in the smartphones to know the team location. In case none of the team devices have a GPS, the application will generate a two-dimensional layout of nearby team elements.

There are several techniques to obtain the relative position, which will be described briefly. The localization of nodes frequently relies on determining the distance from these nodes to dissimilar referential points with well known locations\([3][17]\). Thus, distance calculation is an important point in positioning

\(^{50}\)http://developer.android.com/reference/android/location/LocationListener.html - accessed in September of 2013


\(^{52}\)https://developers.google.com/maps/documentation/android/ - accessed in September of 2013
systems, as Wi-Fi provider and cellular network can attest. The most often cited techniques for measuring the distance between nodes are the following:

- **Time of Arrival (TOA):** it is obtained by calculating the time a signal takes to travel from a source node to an target node. For instance, the GPS technique introduced before is a good example. One limitation of this system is that it needs precise time synchronization between nodes.

- **Time Difference of Arrival (TDOA):** it takes into account the time a signal delays to arrive to a target node, by comparison to a given time reference. For example, by comparing the travel time of sound with the light.

- **Angle of Arrival (AoA):** it is obtained by determining the direction of propagation of the signal occurrence on an antenna. To put it another way, allows obtaining the angle from where the measured signal was received by the node. Although it is not a direct measuring of the distance, it is used as a complementary technique for TOA and TDOA methods, despite it needs a wide range of nodes to wrap a 360° area.

- **Received Signal Strength Indicator (RSSI):** it is based on the propagation aspects of radio-frequency wave, particularly on the relation between the received signal strength and the corresponding travel distance. The main drawback of this technique is related with the non-linearity of the propagation phenomenon, that rely upon different factors such as: temperature, relative humidity, multipath fading, shadowing, etc.

The TDOA and AOA methods require expensive implementation and infrastructure, while the Time of Arrival provides high accuracy, requires a high complex system of time synchronization between the nodes. Thus the Received Signal Strength Indicator that depends on the theory that received signal power is inversely proportional to the square of the distance is a much more practical technique to implement in the Social Sensing project.

The Android API only allows to read the RSSI from the Bluetooth discovery process. This process commonly include an inquiry scan of about 12 seconds, followed by a page scan of each found device to retrieve its Bluetooth name. As said before, to start discovering devices, we simply call startDiscovery(), but to read the RSSI value we need to add an extra field to the discovery intent. This extra field provides the RSSI value of the remote devices as reported by the Bluetooth hardware. Although we rely on RSSI, methods such as propagation time become possible, the mobile application can use them alternatively. The RSSI unit returned by the Android device is dBm, which is an abbreviation for the power ratio in decibels (dB) of the measured power referenced to one milliwatt (mW).

In theory, we can obtain the distance based on the radio propagation model and power level using

---

the equation 4.5:

\[ RSSI (dBm) = P_{TX} + G_{TX} + G_{RX} + 20\log\left(\frac{c}{4 \times \pi \times f}\right) - 10\log(d) \]

\[ = P_{TX} + G - 40.2 - 10n\log(d) \]  \( (4.5) \)

where \( P_{TX} \) is the transmitted power; the \( G_{TX}, G_{RX} \) are the antenna gains and \( G = G_{TX} + G_{RX} \) is the total antenna gain; \('c'\) stands for the speed of light in the free space (approximately \( 3.0 \times 10^8 \text{m/s} \)); \('f'\) is the central frequency (2.44 GHz for Bluetooth); \('n'\) is the attenuation factor (2 in free space) and \('d'\) is the distance between transmitter device and the receiver device in meters. Thus \('d'\) is given by equation 4.6.

\[ d = 10^{\left( \frac{PTX - 40.2 - RSSI + G}{10n} \right)} \]  \( (4.6) \)

In theory the signal strength falls off logarithmically with the distance, but this model can only be used as a theoretical reference, since the measurements are corrupted by reflection, obstacles, noise, antenna orientation and device context (e.g., inside a backpack, on the pocket) leading to a more complex relationship between received signal strength and the distance. Experimental methods were developed by running several propagation experiments in outdoor in order to convert each RSSI reading to a distance estimation with an error bound.

The devices were positioned at known distances and several samples of the RSSI were measured. The position of the devices were parallel to each other, also accordingly with several studies done by the authors of [17], the position of the Android device (different antenna orientation) seems to have little impact on the measurements. They assume that since the orientation of the device, antenna design is optimized for these situations.

Since the android device used does not mention the power transmitter or the antenna gain, we use the values predicted by the propagation model with \( P_{TX} = 2.9 dBm \) and \( G = -4.82 dBi \) [17]. The results obtained were different from the theoretical values, as expected. In average the RSSI was approximately different \(-26.9739 dBm\) from the expected value. After introducing this error in the equation we obtained a relatively close results to the theoretical values. The figure 4.4 shows the results obtained before and after the correction.

Due to the environment conditions where the RSSI is measured and the phone context it is not straightforward to draw a definitive conclusion about the relation between Bluetooth RSSI and distance.

Most of the examined algorithms use geometrical calculations based on successive triangulations or multilaterations to assess a position relatively to some reference points. The algorithms may proliferate possible measurement mistakes even though they use minimization methods so typical in these algorithms. Given these circumstances, the MultiDimensional Scaling (MDS)[2][3][33] was chosen as the most qualified for the system’s requirements. It is possible to make a matrix “\( Y \)” of Cartesian coordinates of points in Euclidean space, knowing the Euclidean distances “\( D \)” amongst those points. Thus the MDS is a set of data analysis techniques that converts a distance matrix into a set of coordinates, such that
the (Euclidean\textsuperscript{55}) distance derived from these coordinates approximate as close as possible the original distances. The following steps explain the functionality of the MDS\textsuperscript{3}.\n
A distance matrix D receives the known measured distances between N devices in a network. The double-centring method is used, in which the matrix is centered around the average value of its entries, in this manner acquiring the B Matrix is done with equation 4.7.

\[
B = \frac{1}{2} \times (I - \frac{U}{N}) \times D^2 (I - \frac{U}{N})
\]

(4.7)

where \(I\) is an identity matrix with \(N \times N\) dimensions and \(U\) a matrix with the same dimensions as \(I\) but filled with ‘1’s. Later using 4.8 we compute the eigenvalues (matrix A) and the eigenvectors (matrix V) of the B matrix.

\[
B = VA \times V^T
\]

(4.8)

The \(m\) biggest positive eigenvalues and its correspondent eigenvectors are retained. The \(m\) is the desired number of spatial dimensions (2D, \(m=2\)).

The spatial placement which refers to the devices coordinates relatively to each other is retrieved from the calculation of the X matrix provided by the equation 4.9.

\[
X = VA^{\frac{1}{2}}
\]

(4.9)

In this project we used the MDSJ library\textsuperscript{56}, which is a free Java library for MDS. However if localization is opportunistically available, as we can see in the algorithm 4.5, the application uses it to translate the relative location to latitude and longitude coordinates of all the devices without absolute location. We

\textsuperscript{55}http://www.princeton.edu/ achaney/tmve/wiki100k/docs/Euclidean_space.html - accessed in September of 2013

\textsuperscript{56}http://www.inf.uni-konstanz.de/algo/software/mdsj/ - accessed in September of 2013
need at least two nodes to report their absolute location. The application fixes the spatial location of these devices using their geographic coordinates in place of their relative coordinates.

Figure 4.5: Translate the relative location to latitude and longitude coordinates.

However, it may happen that the devices does not comply with the characteristics of the Euclidean space, and in that case we might have an X matrix filled with Not a Number (NaN). If such happen, we will try to minimize the situation by showing the location of the team members which uploaded their location to the cloud. Once again, in case the devices take too much time to reply, the application runs a thread in background to proceed with the algorithm. The devices contacted for the GPS request, will read their GPS data and reply back to the team leader. When the team leader receives all the requested GPS data, will lock the table, like the algorithm in 3.8, but the logic follows the count down thread described in the algorithm 4.5.

The translation of Cartesian coordinates into Latitude and Longitude is as follows: Since we know the distance relation of the devices, we center the map in one of the devices with GPS. If only one GPS is available, it is fixed at the center, and the position of the other devices are translated from Cartesian to Polar coordinates, using the equations 4.10 and 4.11.

\[
r = \sqrt{x^2 + y^2} \tag{4.10}
\]

\[
\theta = \tan^{-1}\left(\frac{y}{x}\right) \tag{4.11}
\]

With 4.6.6 and 4.6.7 and knowing the distance relation in the 2D diagram, we can convert all positions to latitude and longitude. But with only one GPS, the real position of the other devices can be in any of the 4 quadrants. In case we have one more GPS is possible to fix the second point, and rotate the map.
in order for the rest of the points follow the translation of the second fix point. After successfully obtain
the location of peers in the Bluetooth range the team leader sends a message to the Internet Manager
which will then request the location of the remaining peers. Upon receiving the reply form the Internet
Manager module, the Location Provider will contact the Application Core to connect over Bluetooth to
nearby team devices in order to share latitude and longitude of the team elements. This information is
also sent to the screen responsible for sharing fitness and location data.

4.6.5 Social Client

A social network authentication allows to simplify the user registration and login. It does so, by reducing
barriers to register like eliminating the need for users to fill out monotonous registrations forms and
remembering the email and password combination to access the platform. Using a SN account to login,
users are able to authenticate with one step.

For the time being, the Facebook is the only SN supported (see section: 2.4). There were two
main reasons to choose the Facebook SN, which were: (1) Facebook is without doubt the SN with
more number of active users; (2) The Test User API, that allows to create hidden user accounts
associated with an application for the intent of testing the functionality of the application.

In order for the SSMA to support Facebook authentication one needs to generate application signa-
ture (keyhash) for the Facebook settings and import the Facebook SDK for Android. After generating
the application signature successfully it’s necessary to register as Facebook developer and proceed to
create the Facebook application. Subsequently the Facebook SDK must be added to the project. This
SDK provides a Session object that is used to authorize the user to use the mobile application and
manage the Facebook Login flow and the application session.

The authentication process can be divided in three steps:

- Set up the UI - The SDK provides a customized Facebook Login button, that is used in the Regis-
ter/Login screen;

- Wire up the authentication logic - After the user select the Facebook authentication, a pop up
will be show with the permissions the application desires to have access to the user. If the user
agree, profile information will be retrieved such as email, name and user identification. Then his
information is forward to the back-end as it was a Social Sensing authentication;

- Add the logout flow - When the user logout, in case he had login with the SN, the application close
the session and clears session token.

The SSMA allows the user to share his ride data with his friends. The Facebook SDK has a method
to publish stories to the user’s timeline. In case the user wishes to publish his data on Facebook,

the data is published on user’s behalf, as soon as the ride finish. In case the user did not login, but has this function enabled, a Facebook Login form is prompted.

The publish logic is as follow: We need to add a method from the SDK, called publishStory(), which checks if the user has granted publish permission to the application, if not, the application will prompt o reauthorize the application and give the missing permission. Then, it creates a Request object that is executed asynchronously that will pass the user’s current session to the Graph API, a key-value of Post parameters using the HTTP method (POST). The method has a callback to handle the response when the call finishes. In the end the user will be informed of the success or failure of the outcome of this action.

4.6.6 User Interface

The UI design has become one of the key issues in developing Android applications. The developer needs to find a compromise between building an interface that is clear but rich and making it simple [25].

When the user starts the application he is presented with an image of the IST logo, which stays in the user screen for 2 seconds. We verify if the user table is empty or not, in case is empty, means it is the first time the user interacts with the application, and he is redirected to a authentication screen, otherwise the user is directly submitted to the main Menu screen.

The authentication screen allows the user to perform the following options:

- **Register** - If it is the user first time in the SSP he can login using his name, an email account and a password. The user can also register to the system using his Facebook;

- **Login** - In case the user is already registered, he just needs to insert the email and password, or use his Facebook social account to Login;

- **Skip** - The user can simply skip any authentication process and directly proceed to the main Menu screen, although functionalities related with the back-end will not be available;

- **Exit** - The android SKD does not provide a method to exit the application, usually what developers do is bring the application to background. This action does exactly that, move the screen to the back of the activity stack, giving the sensation the application is gone.

The figure 4.6, illustrates the functionalities described, plus using the Facebook social account to authenticate.

The register/login, either by the Social Sensing credentials or by using the the Facebook social account, operations and done by sending a message to the Upload Manager module.

After a successful authentication or by skipping, the user is brought to the main screen. The main screen provides four screens with different functionalities, such as show ride data, edit team, configure ride profile and application settings. Complementary to this main screen it is used a personalized action bar and non-intrusive to screen context, which is dedicated for navigation and frequently used operations.
Figure 4.6: Authentication screens.

The action bar developed extends the Activity class, implementing all its main methods. This allows to all activities presented to the user, after a successful authentication, to extend this action bar class, providing a consistent user experience through all the screens. The action bar menu is illustrated in the section A.6.

The application logo is not only used as the app’s identity, but also as a navigation button, since the user can navigate up the hierarchy. The plus button functionalities are mostly for when the user is standing still, and display the following actions:

- **Request Route Noise** - allows the user to perform a quick search for a more suitable route, with less noise pollution;

- **Request Route Steepness** - same functionality as the previous, but shows the steepness instead of the noise;

- **Request Location** - quick action to discover his current location in a map, using friend’s GPS. For instance the user is resting, with the application paused, and desires to know his current location. To do so, he just need to click in menu, and a pop up is shown with the outcome of the request (GPS data successfully gathered or no device nearby with GPS);

- **Request Team Location** - the user can instantaneous request his team location. For example the user is moving out of home and desires to know where his teammates are, this action provides the user the location, time and environment conditions of the most updated version of his team peers;

- **Join Team** - this action allows the user to join a team upon selection, it is performed using Bluetooth.

The people button is responsible for presenting team functionalities, such as build a local team, using the Bluetooth technology, download a team formed in the web application, or to simple choose a team already saved in the device.

Privacy is an important issue in the composition of the SSMA. The application provides a Log out button that allows the user to delete all sensitive data stored in the device. The user has perception of what is happening since a loading bar is displayed to the user. After performing this action the user is submitted to the authentication screen. Since it is not desirable for the UI to block until the corresponding operation finishes, the application uses a handler. This handler sends information to the application core.
that runs in the background and is not dependent on the UI, providing a good user experience, since all potentially slow running operations will run in background.

The Ride Information screen is responsible for showing the user fitness metrics and location. This screen is fully customized, as the user only see what he wants. The Team Information screen allows the user to use all team functionalities, described before. The user is also able to edit locally a team. For instance, the user downloads the team from the cloud, but not all members goes on the ride, so the user can simply remove those cyclists from the device. The ride profile screen allows the user to save different ride profiles. For example, the user may want to define a cycling profile for when he goes with his friends and another when he goes alone. This screen also allows the user to see and delete the last rides stored in the device.

An important aspect of mobile crowdsensing systems is that they potentially collect sensitive data belonging to individuals. For example, GPS sensor readings can be used to deduce private information about an individual, such as the rides they take and time. But they are also important for showing the user location to his friends or give some context to the environment data gathered. Therefore, it is important to preserve the privacy of an individual, while at the same time allow the system to work properly.

The Setting screen allows to user to enable and disable the sensing modalities supported on the phone. Thus the user can control the privacy policy from the device and determine what part of his presence is shared on the Social Sensing Project. Also all data shared in the communal project (e.g., environmental mapping) is shared anonymously. In another words, it is not possible to identify the users who share data into the environment mapping. The drawback of this approach is that malevolent users may share erroneous data, becoming difficult to maintaining the integrity and veracity of the sensor data gathered to the project. In the future, as the project has more users, the data can be combined and since probably this kind of users will be a minority, the average of the data for that specific route becomes more reliable.
Chapter 5

Evaluation and Results Analysis

The application was developed for cyclists and a set of tests were designed and carried out with that objective in mind. The first tests will focus on two main points: the noise pollution and inclination level. The tests will expose cyclists to flat terrain, gradual downhill and steep uphill, as to different noise environments. Later some tests will compare power performance of using a centralized or a decentralized approach for forming a team. Additionally, a power consumption comparison between the SSMA and a professional mobile application will be described. The relative positioning method compared with the GPS one. In the end, there is an analysis to the usability tests performed to the SSMA and SSWA.

The power consumption was measured using PowerTutor\(^1\), which could provide accurate power consumption estimation, including CPU and Wi-Fi, among others\([34]\). This application uses a power consumption model built by direct measurements under the cautious control of device power management states. This model usually gives power consumption estimates within 5% of actual values. The PowerTutor offers a text-file containing the detailed results. The PowerTutor was used in two situations: (1) to calculate the power consumption of each functionality offered by the social sensing application (2) to compare the power consumption characteristics of SSMA and a rival application.

5.1 Cyclist Experience Mapping

This section presents the results from several group experiments targeted at the cyclist experience mapping. It starts by analysing the environmental data gathered along an individual cycling experience, which afterwards will be analysed.

5.1.1 Route environment mapping

One of the objectives of our project is to provide useful cycling information for cyclists to help them choose a better route. A usual preoccupation of the cyclist is to find good routes in a cycling place, of which the safety will be considered an important factor. Apart from that, noise pollution and steepness will also be taken into account.

\(^1\)http://ziyang.eecs.umich.edu/projects/powertutor/ - accessed in September of 2013
The safety and noise pollution problems of a place are related with the traffic there. To infer the presence of traffic along the cyclist route the mobile application uses the device microphone. The enjoyment of a route is related with it's steepness and can be inferred using a variety of sensors embedded in the device.

A pilot evaluation of the system was conducted in the city of Cascais in Portugal, with a cycling team "Alfa" of three members. During this evaluation, the users performed several rides along the same route, which will be named the ground truth route. It took cyclists 30 minutes to finish the ground truth route at a normal speed. The ground truth route has a variety of urban cycling terrain, including busy roads and peaceful back roads with which we can detect the noise. As to the steepness, this route exposes the cyclists to flat roads, gradual downhill and steep uphill.

The devices were attached on the bicycles using a support for smartphones and fixed with adhesive tape so that they could not move. Before the start, it is necessary to calibrate the noise and the inclination of the devices. The calibration should be done to every device, since their positions on each bicycle are different and a sensor's accuracy differs from device to device.

After several rides along the ground truth route, an amount of data was gathered. The figures in 5.1 illustrate the data gathered along the ground truth route for the noise and steepness. The high, low, and average data values are associated with the colours on the map: red, green and yellow, respectively.

As one can observe, the downtown area appears redder than other places on the map, which means the pollution noise is higher there. The reasons are apparent: there are more cars and people in the center. As the cyclists move away from the downtown, the pollution noise decreases as expected, indicating a more peaceful area. However, the traffic is not the only reason affecting the noise level. There are other factors related like the wind, cyclists yelling comments among them and the vibration on a rough pavement.

The steepness ride data illustrated in the figure 5.1 only shows the intensity of the uphill or downhill. This happens because a downhill may be a uphill for another cyclists depending on the perspective. When the cyclist is idle, a device can register accurate measures. When the bicycle is moving, the error increases as the speed changes or the terrain becomes tougher. For example, the steepness may
register a big value because of an unexpected stone which causes vibrations to the bicycle. Apart from that, any slight movement in the device may cause a huge distortion in the measured slope, due to the non-linear nature of the inverse tangent function used to obtain the slope. We believe that if a filter is used to remove the bicycle vibrations from the data, a better result can be achieved.

In conclusion, the Social Sensing platform provides not only accurate noise pollution mapping to the cyclist community, but also, in a more personal perspective, important information about safety. Also the system supplies information about route steepness so that the users can get some guidance for cycling routes, although additional studies need to be done in this area.

5.1.2 Social cycling

Cycling can be a social activity. Quite often do cyclists perform their activities in group. If everyone of a team uses the Social Sensing application, one could expect some correlation and consistency of the data collected by the sensors of the same team. This can be used for calibration purposes, irregularity detection and noise reduction. For instance, instead of every device in the team calibrating itself, one of the devices can share his calibration data with the others, either using a decentralized or centralized approach. That is to say, they will automatically calibrate themselves. Let’s have a look at the data of one of the many cycling experiments to examine this aspect of the SSP. In this ride, cyclist were asked to remain about 5 meters from each other. The rough used is the same as in section 5.1.1. The team is constituted of three elements, Silva, Mansur and Yao. Silva and Mansur started the ride together and calibrated the device before they started the ride. Yao joined the team in the middle of the ride and then calibrated the device.

The figure A.24 shows the noise pollution gathered along the ride.

According to the figure A.24, it’s obvious that the data they collected share some correlations. For instance, areas of high noise are captured by all the three cyclists although the intensities may be different. Silva and Mansur share more noise correlation than Yao. This could be explained by the fact that Yao did not calibrate device in the same context as Silva and Mansur. Additionally, there are several factors affecting the noise sensing, such as multipath reflections of sound, changes of the wind, sound

Figure 5.2: Noise data gathered along the route. a) Silva, b) Mansur and c) Yao
made by the cyclists, inverse square fading of volume with distance, cars passing by each cyclist at different times, etc.

The steepness data gathered by the cycling team is illustrated in figure 5.3. The steepness data correlation is less evident than the noise, but one can still observe some tendency lines over the map. The reasons could also lie in the road condition, wheel size, tire condition and the cyclist behaviour. Comparing the data inside the group, Yao shows less intensity steepness data than the rest of the team, showing the importance of calibrating the devices in the same context. Also this may be related with the fact that Yao’s device was better fixed on the bicycle than the other two. In places of high steepness, all the cyclist's devices register a high steepness value.

When it comes to analyse the data, the GPS data is strongly correlated among the cyclist team members, as expected. Thus, the idea of sharing some types of sensor data is not only possible but also useful.

5.1.3 Comparison with other Application

In order to evaluate the CPU and battery consumption of the mobile application, one needs to compare with another application for reference. The application chosen was Edomondo, as described in section 2.8.5. Edomondo is probably the most widely used mobile cycling application in the Android O.S world. The trial was ran as follow: First the biker will cycle for ten minutes with both the devices on. After these ten minutes, he pauses the application and stops to rest for five minutes, enjoying the views around. After restoring the strength, the cyclist gets on his bicycle and restarts the application. After cycling for another ten minutes he decides to stop the application. The functionalities used on both of them were mostly the same (e.g., provided by the GPS). In this way it is possible to compare the power performance of both.

The figure 5.4 illustrates the CPU consumption between the two applications.

As is possible to observe, the Social Sensing application uses more CPU than the Endomondo. One of the reasons may be related with the periodicity of the location requests, probably the Social Sensing application requests more often GPS updates than the Endomondo. But the most important is the fact
that, when the application starts, it also starts the PSF even if the modules are not used. Thus it means the applications is running two services that are just waiting for requests.

The battery consumption is shown in the figure 5.5. Since the Social Sensing consumes more CPU, it leads to a higher battery consumption as expected. Another problem in this test is: since the devices are from 2010 and they have been widely used it leads to a deterioration of the condition of the battery.

5.2 Comparing distributed and centralized approaches

The SSMA can use a distributed or a centralized approach to form a team. In order to understand the impact of each approach in the mobile application, we performed several testes using the team formation process as testbed. The tests were mainly focused in understanding: (1) Time to finish the process, (2) CPU spent during the process and (3) battery used to end the process.

In the centralized approach, this team process is very straightforward, the user click the action “Request team from cloud”, and the teams the user is registered is shown. Then the user just selects the team he desires, and the back-end reply all the data necessary. In the centralized test, the device was registered in nineteen teams, and always choose the team that goes by the name “Alfa”. This team has 4 users registered.

In the decoupled design, three mobile devices with the characteristics described in section 4.1 were used. As said in section 3.3.4, this process consist of two phases. The first phase happens when one of the users starts the discovery and filtering process, while the seconds is when the devices running
the the PSF are shown to the user. Next the devices selected send their capabilities to the team leader, which will be responsible for sharing them with all the team members. The figure 5.6 shows the response time it takes to have the team on the device.

![Figure 5.6: Time in seconds to form a team.](image)

As is possible to analyze, the centralized approach is much more faster than the distributed approach. This happens because the team is already done in the cloud, and the user just needs to choose and download it. While in the distributed goes through a discovery, filter and exchange team information process. The figure 5.7 shows the delay in the decoupled team formation process divided in the two phases.

![Figure 5.7: Time in seconds to form a team using distributed approach.](image)

The delay in the distributed approach is related with the Android Discover process². This discovery commonly involves an inquiry scan of about 12 seconds, followed by a page scan of each found device to retrieve its Bluetooth name. Also one needs to add the delay due to the PhoneSensing filtering process. In relation to the second phase, one of the tests shown a time of 5 seconds, if the user is quick choosing a name for the team and its elements.

The figure 5.8 shows the CPU and battery consumption for each of the approaches. The distributed approach is divided in two: (1) device that starts the process (e.g., team leader), (2) device involved in the process (e.g., team member).

Analysing the figure 5.8, the distributed process has a higher CPU and battery consumption than the centralized. This is normal, since is has a higher response time and involves more asynchronous tasks. Comparing the devices involved in the team formation with the centralized approach, is possible

Figure 5.8: Power consumption for centralized and distributed approach. The distributed approach is divided in two: team leader and team member.

to observer the CPU consumption is almost similar. This is probably due to the fact they use the same number of asynchronous tasks. The battery diverges due to the time the device is active in the team formation process and the Bluetooth usage during that time.

Comparing the different devices involved in the distributed team formation, is feasible to observe the team leader is the one that has a higher CPU consumption. This is expected since the team leader has to discover and filter the devices. Also it has to perform more Bluetooth connections than the other devices involved in this process. As a final analyse, is possible to see the battery consumption is not the major cause to the battery consumption, but probably the time the device is active. For instance when a device is active in a task, the screen is on (fully bright).

5.3 Evaluating Relative location algorithm

While the GPS is a natural solution to the design of the SSMA, not every device may be equipped with a GPS receiver. Furthermore GPS is restricted in its application because one of the prerequisite is to be in line of sigh of at least three satellites to obtain a two dimension position. Thus, it may not be consistently functional in some ordinary cycling environments, such as among high buildings in a city, or other places where the sky is not visible.

The SSMA provides a node-based relative positioning functionality that depend uniquely on the mobile device of the cyclist.

The primary goal of this algorithm is to accurately localize nearby peers. Therefore in this section we pretend to infer the location error provided by the algorithm. In order to achieve this goal, we started by placing three devices (e.g., let's call them: "A","B" and "G") running the mobile application apart from each other. The separation among the devices was selected taking into account a range of distances that is probably used by bicyclist when they cycle in small groups (e.g., three elements). This distances are also distances covered by the Bluetooth Technology present on the devices. The distances were measured using a tape. We expect the values measured by the tape to have a deviation from the exact ones, but be trustful enough for the tests.

Next we run the node-based relative positioning functionality, obtaining this way the position and
distance among the devices. After successfully running more than thirty experiments, we proceed to
do the same experiment, but this time using the GPS receiver embedded in the devices. The website
csgnetwork\(^3\) was used to convert the latitude and longitude of the GPS coordinates into meters.

The table 5.1, contains the values obtain from the several tests performed and their respective error.
The error was obtained by subtracting the distances between the devices measured by the relative
location and the GPS location with the "real" location.

Table 5.1: Average values measured during the tests and their respective errors.

<table>
<thead>
<tr>
<th>Measure type (m)</th>
<th>Alfa to Beta</th>
<th>Beta to Gama</th>
<th>Alfa to Gama</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Location (m)</td>
<td>4.20</td>
<td>3.50</td>
<td>4.15</td>
</tr>
<tr>
<td>Relative Location (m)</td>
<td>5.24</td>
<td>3.28</td>
<td>5.08</td>
</tr>
<tr>
<td>Error of Relative Location (m)</td>
<td>1.04</td>
<td>0.21</td>
<td>0.93</td>
</tr>
<tr>
<td>Absolute Location (m)</td>
<td>31.46</td>
<td>11.15</td>
<td>38.24</td>
</tr>
<tr>
<td>Error of Absolute Location (m)</td>
<td>23.72</td>
<td>4.20</td>
<td>25.74</td>
</tr>
</tbody>
</table>

Analysing the table 5.1, one can attest that the relative location algorithm is good to estimate dis-
tance, at least for small distances between the peers, as the ones used by cyclist when ridding in small
groups and not in competition mode. Additionally the algorithm supply more trustful measures than the
GPS for this specific case.

Although accuracy in localization is the main goal of this algorithm, energy consumption is also an im-
portant concern for mobile devices. The figure 5.9 illustrates the average CPU and battery consumption
measured while executing the relative location algorithm.

In order to have a point of reference, we are going to compare the power consumption of this algo-

\(^3\)http://www.csgnetwork.com/gpsdistcalc.html - accessed in September of 2013
this relative location algorithm, the device that starts the process already knows the devices information that belongs to his team.

In summary, the results presented in this section shows that the algorithm can be used to place the devices in a two-dimensional plane with low distance estimation error between them. Moreover in case some nodes have GPS, their location can be pinned and then translate the relative location to absolute of those who do not possess GPS receiver or wish to save battery.

5.4 User feedback

This section describes a test plan for conducting a usability test for the web and mobile application.

The debate over how many test participants one needs to test an application has raged for a long time in the usability community. Accordingly with [14] testing with five users will find 85 percent of the problems. Therefore to validate the usability of the application, we followed the approached described in [14]. This is an iterative approach that focus on finding and fixing the top usability problems, taking the most of test participants to improve the application. The group of users that tested the application may or not had ever seen the application, since in the case of the usability tests to SSWA was done using google forms and spread in several online pages and groups related with cycling and sports. The test performed with the mobile application were done in presence.

In both the tests we pretend to pinpoint critical usability concerns within the applications, make recommendations for addition improvements, enhancements and provide usability targets for further application development. The test whether user needs have been met by the application was already described in 2.7, and has already been validated by the cycling community.

5.4.1 Web application

To each user was asked to execute a set of eight tasks, and in the end to fill a survey. The set of tasks essential refer to the interaction of the user with the interface developed. Since the user did not had any data, after the user register to the SSWA, he was asked to log out and log in with a predefined user which already had ride data. Most of the users were male, and all of them have between 19 and 34 years. The figure 5.10 shows the browsers the user used to access the web application.

As is possible to observe, the most used browser was the Chrome. Independently of the browser accessed the application, all users could successfully access it and authenticate in the application. In all

---

due fairness, one can assume the web application can run in multiple platforms, at least for the browsers used to access the application.

In order to understand the simplicity and efficiency of the web application it was questioned the time spent on each task and the number of mistakes done during each task.

![Figure 5.11: Average time spent for each task.](image)

The figure 5.11 illustrates the time spent for tasks, as is possible to see 50% of the users manage to it in less of 2 minutes as intended. To gather a full picture of the efficiency we also asked the users to register the number of mistakes done in each task, as can be seen in figure 5.12.

![Figure 5.12: Average number of mistakes for each task.](image)

As is possible to analyse, 63% of the users did between 0 and 1 mistake, a low percentage. But the figure 5.11 and 5.12 shows that the web application can be improved. One of the most features the users requested to change was the sub-menu, a recommendation for addition future improvements.

One of the requirements of the application was to provide information to the cyclist about the healthiness of a given route in terms of noise pollution and steepness. This request was successfully achieved since all users manage to understand the environmental data displayed on the map.

### 5.4.2 Mobile Application

Fifteen participants were selected to experience the mobile application, all of them had cycled at least once in the summer of 2013. 6 participants were female, and the ages of the participants were comprised between 22 and 25, with an average of 23.21 years old. From these 10 had previous knowledge of the social sensing application, while the rest just know it was an application for cyclist. They were asked to execute a set of eight tasks, and in the end a set of functionalities related with team functionalities was shown. After this, the participants filled a survey. The tasks are mainly related to the single cyclist functionalities, due to the low number of devices available and time constrains.

A good application is designed to be rich and intuitive. To understand how easy the application is we tracked the number of errors committed by the users. Most of the users, did the tasks without any mistake and in less than two minutes per task. Four users had difficulty to find the road with less noise.
In average these four users committed 2 mistakes to find this functionality. Another difficulty felt by the participants, was to personalize the ride screen, with 5 of them making in average one mistake. Thus these two areas, needs to be revised.

Analysing the survey given to participants, when asked if the application improves the group cycling experience, 93% of them said yes an extraordinary high rate of validation. This information is illustrated in figure 5.13.

![Figure 5.13: Improvement of the group cycling experience.](image)

When asked why the mobile application improves the group cycling experience, most of the replies were related with the team tracking functionality. Most of the participants (87%), thinks the route information provided helps them to choose a better route and more than 80% of the participants manifest interest in buying the application or recommending it to their friends if it was available in the market.

From those who use a cycling application to track their rides, 69% were interested in changing to the Social Sensing application, as shown in the figure 5.14.

![Figure 5.14: Switch of application.](image)

From those who did not want to switch, the reasons where mainly related with interface improvements and the data already saved in the “other” application. As a final comment 73% of the users would give between zero and five euro for the application.

In the course of this study we learn that some further tests and improvements need to be done in the mobile application. For instance the edition option of the “new ride” screen needs to be removed from the “settings” screen. This action should be implemented in the “new ride” screen. But, with good judgement one may say that the main goal of this project was achieved. The participants validated that the application manage to supply added value to the group of cyclist.
Chapter 6

Conclusions

The Social Sensing platform aims to provide several benefits to the users on a personal, social and public level. These benefits can be either by providing personal monitoring, leverage the opportunities made available when devices form a swarm or by collect/sense and share environmental ride data with the cycling community.

The motivation behind this project is to explore different research topics, while at the same time improve the group cycling experience, something not taken into account by most applications.

To achieve the proposed goals we started by creating a mobile application that would fit the project requirements. The mobile application focused on providing accurate fitness measurements, route healthiness and advanced services like tracking team members, resources sharing and position estimation. To achieve this goals, three different resources sharing systems were analysed. The required system needed to have a decentralized nature, so the devices were not dependent on any infrastructure support and the protocol used needed to be fast and light due to the mobility of the cyclists. In the end, the PhoneSensing project/framework was chosen, since it is the one that better fits the application requirements. To overcome some limitations of this framework, we added new modules. This new modules allow the framework to gather accurate audio and inclination data and map the environment ride of the user.

The position estimation was achieved using an algorithm that provides localization of every node relatively to others by placing them on a two dimensional plane without the need of a GPS receiver embedded in the devices or any infrastructure support. In case some devices have a GPS receiver they can be used as anchors so that relative coordinates may be converted to absolute ones. In some situations the group of users may fall apart, in that situation the application connects to a back-end to retrieve the location of the remote users.

The back-end provides way of centralizing important information. This way it is possible the user to be able to access his data from different places and devices. Additional, as said before, serves as a bridge for the devices to exchange crucial information such as their location. This back-end has a service oriented architecture, were virtually any client can request information from the database. The API provided to the clients allows them to submit application requests like user authentication, upload
Another application requirement of the platform was to give the ability to the user to perform trend analysis and share route information with the community. That was achieved using a web-based application that uses a web browser as a client. This web application has graphical information of a cyclist data and route health, in terms of safeness, noise pollution and inclination. This way the user is able to analyse his fitness data and plan in advance his rides.

To evaluate the system, several tests were performed. We started by evaluating the accuracy of the route environment mapping and the correlation of the data when the bikers cycle in groups. The tests show the microphone can provide reasonable accuracy, while the method used to obtain the inclination needs further improvements due to the cyclist ride environment like type of road, and bicycle vibrations, which leads to distortion in the measures, due to the non-linear characteristic of the inverse tangent function. It's possible to observe correlation in the data gathered by the cyclist when they cycle in group, proving that sharing some types of sensor data is not only possible as it also can be useful. Then we proceed to analyse the power consumption of the application. The preliminary studies suggest improvements can be achieved to save battery. For instance, the PhoneSensing modules should be started and stopped on request, and not when the application starts and be stopped when the application is turned off. A comparison between the decentralized and centralized approaches was made using the team formation process as "Fireproof". As expected, if the team is already formed and the devices possess internet connection it is fast, and more power efficient to use the centralized approach to get the team information. In the case no internet connection is available the decentralized approach can prove quite useful.

Later on, the usability of the system was tested with a group of cyclists that filled an online survey. In all due fairness we observe the users liked the system and were eager to use it, but also showed that it need some improvements, specially on the UI.

There are some privacy and security issues that we have not address in the SSP. In our current application, a user that starts a team creation process can have information about the users (e.g., user email) without his permissions, or the data uploaded by a mulling process (e.g., a team member requests another team member to upload his location data) can be changed with some malice, since it does not go encrypted. However, every team a device is part of a team formation process a notification is shown to the user. Additionally the range of the Bluetooth Tecnology is low, and if the user is cycling alone he can disable the Bluetooth. In relation to the mulling, the user only use this process if he want, also since the mulling process only happen between team members we expect a social relationship between them.

The main system limitations are associated with the decentralized approach. The Bluetooth Tecnology presented several limitations both associated with the Bluetooth hardware and protocol implementation. The protocol have a high discovery time and low bandwidth that would be disadvantageous to a large group of users. Another drawback, which comes from the time of PhoneSensing implementation is the filter process. The time needed to discover a device that is running an instance of the PhoneSensing is very high, leading probably to stressful situations in case the cycling group is constituted by several elements.
Summing up, while the current experiments have concentrated on sensing for the cyclists, the platform can also be targeted to any user that enjoys to perform outdoor group activities or even to map traffic and noise pollution in a city scale.

6.1 Future Work

This work is a base for many future developments that can make this project more capable or, as said in the conclusion, extend the experiments scope beyond the cycling activity. It would be also interesting for the devices to interact with external sensors like a heart monitor, which can provided more rich information to the user. In relation to the SSMA, the power consumption is a obvious concerns and it needs further optimizations. The mobile application could manage better the PhoneSensing services running in background, among other general code optimizations.

Although the Bluetooth is good for prototype development it is not ideal for this application environment, due mainly to the high latency and short range. In this project the bluetooth was mainly chosen because of the device's limitation to establish ad-hoc WiFi connections. An interesting improvement would be to use other wireless technologies like the WiFi direct.

In relation to the relative positioning protocol although it provides accurate distance estimation, sometimes the devices position estimation is not possible for not complying with the Euclidean space. It would be interesting to compare the accuracy of the MDS with other algorithms like geometrical calculations based on successive triangulations or multilaterations to assess the relative position of the nodes. The mobile application should also extend the SNs supported.

Security is an important issue that was not properly addressed in this project. Therefore further optimizations at this level is desired. Also the mobile and web application need further UI improvements, as commented by the users in section 5.4.
Appendix A

Annexes

A.1 Cyclist survey - Functionalities
Resumo

1. Cycling Information

1.1 - What is your gender?

- Male: 114 (86%)
- Female: 19 (14%)

1.2 - How old are you?

- <12: 1 (1%)
- 12-18: 67 (50%)
- 19-24: 32 (24%)
- 25-34: 22 (17%)
- 35-55: 11 (8%)
- >55: 0 (0%)

1.3 - What kind of cyclist are you?

Figure A.1: Cyclist survey - Functionalities
On-road bicyclist, primarily 47 35%
Off-road bicyclist, primarily 29 22%
Both on-road and off-road 54 41%
Not a cyclist (Ends here) 3 2%

1.4 - Approximately, how often do you ride?

Every day 31 23%
3 times a week 29 22%
Once a week 33 25%
Once every two weeks 13 10%
Once a month 27 20%

1.5 - Are you currently training or planning to train for a race?

Yes 22 17%
No 111 83%

1.6 - Are you part of any cycling team?
1.7 - How often do you go cycling with friends/team?

- Always: 17 (13%)
- Usually: 36 (27%)
- Sometimes: 36 (27%)
- Seldom: 18 (14%)
- Never -- go to question 2.1: 26 (20%)

1.8 - How many members are there in your cycling group/team?

- 2: 28 (28%)
- 2-4: 49 (49%)
- 5-7: 10 (10%)
- 7<: 12 (12%)

2. Technology Information

2.1 - Do you own a smartphone?

Yes: 9 (7%)
No: 124 (93%)

https://docs.google.com/forms/d/1N7CqAYxO4dbB9eV4nF5bY12d2F4DuJyYe8tp2iloQI/viewanalytics

Figure A.3: Cyclist survey - Functionalities
2.2 - What is the operating system of your smartphone?

- iOS: 20 (22%)  
- Android: 65 (73%)  
- Windows: 0 (0%)  
- Outro: 4 (4%)

2.3 - Do you use any fitness App to track your results?

- Yes: 29 (32%)  
- No: 61 (68%)

2.4 - Where do you store the phone while you are cycling?

Figure A.4: Cyclist survey - Functionalities
2.5 - How important is the cycling application to you when you are riding?

- Mandatory: 6 (17%)
- Nice to have: 21 (60%)
- Neutral: 4 (11%)
- Not needed: 4 (11%)

2.6 - Do you think the application you use improves your group cycling experience?

- Yes: 18 (53%)
- No: 8 (24%)
- Not sure: 8 (24%)
2.7 - Which would be the most important feature for this kind of application?
- Contagem de Km: tracking ups and downs when you plan the routes
- Distance: Tell me if I should go faster or slower
- GPS with bike lanes and tips: see my friends on a map
- Statistical tracking (speed, altitude, distance, ...) Tracking my route
- Time/distance: Tracks sex
- Accuracy: Don’t know, sorry
- Kms/Time: Never tracking
- Offline tracking

2.8 - How useful would it be for your application to track your team members on a map?

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Very useful</td>
<td>8 24%</td>
</tr>
<tr>
<td>Useful</td>
<td>17 52%</td>
</tr>
<tr>
<td>Not useful</td>
<td>7 21%</td>
</tr>
<tr>
<td>Not useful at all</td>
<td>1 3%</td>
</tr>
</tbody>
</table>

2.9 - How would you rate the utility of sharing feeds (via Bluetooth broadcast or to the cloud) with the team members?

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Very useful</td>
<td>6 19%</td>
</tr>
<tr>
<td>Useful</td>
<td>15 47%</td>
</tr>
<tr>
<td>Not useful</td>
<td>10 31%</td>
</tr>
<tr>
<td>Not useful at all</td>
<td>1 3%</td>
</tr>
</tbody>
</table>

2.10 - Would you share the phone resources (GPS, microphone, internet) with other team members?

https://docs.google.com/forms/d/1N7QqAxaOc8d89aN4nF8YfYabZG4uUyeXYpr_KJdttH/viewanalytics
2.11 - Is there any reason?

- Privacy concerns: 15 (83%)
- Outro: 3 (17%)

2.12 - Would you collect noise samples from a cycling route?

- Yes: 18 (53%)
- Maybe: 11 (32%)
- No: 5 (15%)

2.13 - Why don’t you currently use a fitness application?

Figure A.7: Cyclist survey - Functionalities
2.14 - How much do you think a state-of-the-art cycling application suitable for your needs should cost (per year)?

- 0-1€: 54 (41%)
- 1-5€: 39 (29%)
- 5-10€: 27 (20%)
- >10€: 13 (10%)

2.15 - Do you use any other device to measure your rides?

- Speedometer: 48 (34%)
- Heart Rate Monitor: 11 (8%)
- No: 80 (57%)
- Outro: 1 (1%)

3. Maps information

3.1 Do you use maps applications (web/mobile) to plan in advance your
3.2 - How useful would it be for a cycling application to have a map?

- Very useful: 58 (44%)
- Useful: 53 (40%)
- Not useful: 14 (11%)
- Not useful at all: 8 (6%)

3.3 - How useful it is knowing the route’s noise and steepness in advance?

- Very useful: 54 (41%)
- Useful: 55 (41%)
- Not useful: 18 (14%)
- Not useful at all: 6 (5%)

3.4 - What features would you add to a map?

- trilhos de montanha
- sad
- everything
- Relevo
- Traffic info
- location in real time

Figure A.9: Cyclist survey - Functionalities
type of road  MTB tracks  Option to select routes that are generally more calm
interesting points like resting, eating places  location of known cycling routes  degree of
inclination  Whatever  Places where I could get water from.  The coffee’s
Distância, acumulado.  traffic volume of traffic  Cycling lanes, bridges and dirt tracks
check points  speedometer  water places  Points of interest  water nearby  Trânsito
workshop  Danger parts of the ride, nice spots to stop, jumps in the middle of the ride,
traffic parts of the ride and how many time it takes dependently of the speed.  Trajetos de
interesse  Tracks  optional different routes, places of interest like museums, historical
buildings, market etc.  sex  Route made/Route prepared  altitude, making a scheme
of the route  to know if is possible to bikes ride in a specific rode  Gas stations near my
position  transito  hotspots  where can I find whater and medical help  código de
cores para a inclinação

Número de respostas diárias

![Graph showing number of daily responses from 24/08/13 to 05/09/13.](https://docs.google.com/forms/d/1N7QqAxaO4d89aV4-o-FrFrYha2Z4uLyeYpr_jDjtuJViewanalytics)

Figure A.10: Cyclist survey - Functionalities
A.2 Web application tasks
Survey para ciclistas

Obrigado por participares neste questionário sobre ciclismo em grupo. Este questionário faz parte de uma tese de mestrado que tenta melhorar a experiência de andar de bicicleta em grupo. Os dados serão mantidos em sigilo e não serão usados de qualquer outra forma para além do objectivo desta tese.

1. Aplicação Web

1.1 - Entrar na aplicação web: [http://social.webnice.pt/](http://social.webnice.pt/) e faz o registo na aplicação. Como ainda não tens dados registados com este utilizador, vamos supor que és o utilizador “Joao” com o email joao@gmail.com, que tem a password “123” e já usaste a nossa aplicação móvel varias vezes para monitorizar os percurso e o teu desempenho.

    **Resultado esperado:** Ver o percurso percorrido, com marcadores.

1.2 - Consulta os dados do teu ultimo percurso.

    **Resultado esperado:** Ver a hora que passou nesse ponto.

1.3 - Imagina agora que no proximo fim-de-semana vais organizar um passeio de bicicleta com os teus amigos.

    Como vais ser tu a organizar o passeio, vais:

    1.3.1 - Procura nos percursos existentes na comunidade, um percuso com pouca inclinação.

        **Resultado esperado:** Ver no mapa a intensidade da inclinação, com marcadores a indicar a data e o valor.

    1.3.2 - Após consultar, vais agora criar uma equipa para o evento que estas a organizar. Como farias isso?

        **Resultado esperado:** Não aparecer nenhum erro, e ser reencaminhado para a pagina onde se adiciona membros para a equipa.

    1.3.3 - Regista o participante “pedro@gmail.com” na equipa. Como fazes isso?

        **Resultado esperado:** Mensagem a indicar que o utilizador foi adicionado.

    1.4 - Antes de saires de casa para o encontro de ciclismo, vais espreitar na aplicação web se os teus amigos da equipa “alfa” já estão no ponto de encontro ou não.

        Como fazes isso?

        **Resultado esperado:** Marcadores a representar os membros da equipa num mapa.

Figure A.11: Web application tasks
1.4.1- O rui@gmail.com parece já estar na zona da Alameda. Consegues ver se esta informação está actualizada?

**Resultado esperado:** Informação a indicar a que horas o rui@gmail.com fez o upload da sua localização.

1.5 - Após teres completado o percurso com os teus amigos, queres ver dados do desempenho da equipa “Alfa”, como vês os dados desta equipa?

**Resultado esperado:** Tabela a indicar dados estatísticos da equipa.

---

**Figure A.12: Web application tasks**
A.3 Web application survey
18 respostas

Ver todas as respostas  Publicar estatísticas

Resumo

1.1 - What is your gender?

- Male: 16 (89%)
- Female: 2 (11%)

1.2 - How old are you?

- <12: 0 (0%)
- 12-18: 0 (0%)
- 19-24: 11 (61%)
- 25-34: 7 (39%)
- 35-55: 0 (0%)
- >55: 0 (0%)

What was the browser that you use to test this application?

- Chrome: 14 (78%)
- Firefox: 2 (11%)
- Safari: 2 (11%)
- Opera: 0 (0%)
- IE: 0 (0%)
- Other: 0 (0%)

Could you access the application?
Could you register and login in the application?

- Yes: 18 (100%)
- No: 0 (0%)

If not, what was the problem?

Aesthetically, the Web interface is pleasant?

- Less than 2 minutes: 10 (56%)
- Between 2 minutes and 5 minutes: 6 (33%)
- Between 5 minutes and 10 minutes: 2 (11%)
- More than 10 minutes: 1 (6%)

Figure A.14: Web application survey
How many mistakes by task did you do on average?

<table>
<thead>
<tr>
<th>Mistakes</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>22%</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>28%</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>22%</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>28%</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>&gt;5</td>
<td>0</td>
<td>0%</td>
</tr>
</tbody>
</table>

What does the red, yellow and green colors on the map related to the task 1.3.1 mean?

Intensidade da inclinação Areas with high inclination The number of routes? it means the inclination red- Date & hour .. the other one i don’t see inclinado, media inclinação e plano (respectivamente) muito inclinado, inclinado, plano red is very inclined, yellow is inclined and green is plain inclinação muito inclinado, inclinado, plano steepness os varios percursos com as variasdas inclinações red - very inclined | Yellow - inclined | green - plain interessante intensity of slope Alta inclinação

If you had power to, what would you add or change in this application?

the team button option, some words are too big The sub-selection menus on the top, like the “create team” in Team, should be different since i failed to select it (the “create team” and others”) a couple of times. An alternates to select where to do the routes (dont know if it already have, since i couldnt create a team(always error, whatever name i put in)) falo ctg pelo gmail as janelas de seleção quando o rato encontra-se em cima de my route, team, my profile, etc.. No nothing os títulos dos menus mais simples Change the location of drop menu when selecting the route the pop up sub-menus (i don’t like them)

Please, give a final comment about the application.

Nice jobreate team” in Team, should be different since i failed to select it (the “create team” and others”) a couple of times. An alternates to select where to do the routes (dont know if it already have, since i couldnt create a team(always error, whatever name i put in)) falo ctg pelo gmail as janelas de seleção quando o rato encontra-se em cima de my route, team, my profile, etc.. No nothing os títulos dos menus mais simples Change the location of drop menu when selecting the route the pop up sub-menus (i don’t like them)

Please, give a final comment about these user’s tests.

https://docs.google.com/forms/d/1fXbIFLYLnl58GXqM55f-7X1C1HeagYXICMDQcFH27XneIvueanalytics

Figure A.15: Web application survey
Please, give a final comment about this questionnaire.

- It helps the developer to understand if it easy or not to used the website.
- Estão bons, concisos e fáceis de fazer. I couldn't create a team...
- Parecem bons, mas será que testas-te a aplicação toda?
- Very good. They are ok.

Número de respostas diárias

Figure A.16: Web application survey
A.4 Mobile application tasks
Survey para ciclistas

Obrigado por participares neste questionário sobre ciclismo em grupo. Este questionário faz parte de uma tese de mestrado que tenta melhorar a experiência de andar de bicicleta em grupo. Os dados serão mantidos em sigilo e não serão usados de qualquer outra forma para além do objectivo desta tese.

1. **Aplicação Mobile**

2.1 Imagina que vais fazer um percurso e queres monitorizar o percurso e as tuas metricas.

2.1.1 – Regista-te na aplicação com o teu utilizador (xxx/yyy)

2.1.2 - Ainda não decidiste bem qual o percurso que vais fazer. Como queres relaxar um pouco, procura um percurso com pouco barulho.

2.1.3 - Configura os settings para recolher dados de GPS e som.

2.1.4 - Como vais recolher amostras de som, calibra o microfone.

2.1.5 - Configura os dados que vao ser mostrados quando estiveres a monitorizar o percurso

2.1.6 - Inicia a monitorização do percurso

2.2 Voltando a volta de bicicleta que estavas a organizar chegou a hora do evento, portanto:

2.2.1 Faz download da equipa Alfa para o telemovel (visto que a criaste na cloud)

1.2.2 Inicia a monitorização do percurso

Figure A.17: Mobile application tasks and demo
3. Demonstração de Funcionalidades

Durante os testes tu criaste uma equipa na aplicação web e depois sincronizaste a aplicação movel para usar essa equipa. No entanto também é possível criares equipas sem acesso à internet, usando Bluetooth.

Imagina que o teu telemovel tem internet, mas nao têm gps, e dois dos membros da tua equipa que estão junto a ti, nao têm internet mas têm gps, a tua aplicação sabe isto automaticamente porque conhece as capacidades dos dispositivos dos membros da tua equipa.

É possível usares o gps dos membros da tua equipa para obteres a tua posição e a dos teus colegas. Todos os outros colegas que tenham acesso à internet, também vão aparecer no mapa, bem como as condições ambientais.

A tua aplicação vai enviar estes dados aos teus colegas que estão ao alcance do Bluetooth, de modo a que todos tenham acesso a mesma informação.

Os teus colegas podem usar o teu acesso à internet para fazerem upload da sua localização para a cloud, para que qualquer elemento com acesso a internet tenha acesso a essa informação.

Não sei se reparaste, mas enquanto fazes o teu percurso a aplicação vai recolher dados sobre a inclinção e a poluição sonora do percurso (desde que permitas), para que depois outros membros da comunidade possam consultar esta informação.

Figure A.18: Mobile application tasks and demo
A.5 Mobile application survey
15 respostas

1) Usas alguma aplicação para ciclistas?

- Sim: 9 (60%)
- Não: 6 (40%)

2) Tens por habito andar de bicicleta em grupo ou com amigos?

- Sim: 10 (67%)
- Não: 5 (33%)

3) Achas que a aplicação melhora a experiência de andar de bicicleta em grupo?

- Sim: 14 (93%)
- Não: 1 (7%)

3.1) Porquê?

- posso partilhar o gps com meus amigos, fazer equipas pois há mais informações que podemos retirar do caminho
- há varias informações interessantes que podemos retirar das rotas e dos passeios
- I can follow my team members. posso perceber-me da

https://docs.google.com/forms/d/13O0WVbdDSD1bacfXCYxGKU5U7LgN7hyS5i5LxLxviewanalytics

Figure A.19: Mobile application survey
localização dos membros do grupo. Reinforce the team spirit. O conceito de grupo, partilha de dados é possível que pessoas que não possuem recursos no telemóvel, tenham acesso a informações adicionais. Partilha de dados, criar equipas, há diversas informações interessantes que é possível recolher da rota, e podemos também saber a localização de todos os elementos da equipa, podemos rever as informações do grupo posteriormente, além de encontrar os amigos que estão atrasados, é possível saber onde estão os outros integrantes do grupo com a aplicação, não há necessidade de parar para ligar para as pessoas. Permite fazer equipas no telemóvel, posso saber minha localização sem ter GPS. Dá trabalho ao usar uma cycling aplicação não preciso.

4) Achas que a informação disponível acerca do ambiente (poluição sonora, inclinação) dos percursos, ajuda-te na escolha de um melhor percurso?

<table>
<thead>
<tr>
<th>Sim</th>
<th>Não</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>87%</td>
<td>13%</td>
</tr>
</tbody>
</table>

5) Se a aplicação estivesse disponível hoje no mercado, estarias interessado em compra-la?

<table>
<thead>
<tr>
<th>Sim</th>
<th>Não</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>86%</td>
<td>14%</td>
</tr>
</tbody>
</table>

6) Se a aplicação estivesse disponível hoje no mercado, recomendarias aos teus amigos?

<table>
<thead>
<tr>
<th>Sim</th>
<th>Não</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>87%</td>
<td>13%</td>
</tr>
</tbody>
</table>

7) Se respondeste sim a questão 1), estarias disposto a mudar de aplicação?

https://docs.google.com/forms/d/13OXWbDSDbDSD8acFCXc5GKUkLUGN7hyRSielfINUK/viewanalytics

Figure A.20: Mobile application survey
7.1) Caso tenhas respondido não, qual o motivo?
Ainda precisa de algumas melhorias a nível estético
não costumo andar em grupo
The button is difficult to press.
ja tenho muitos dados na outra aplicação

8) Quanto estarias disposto a dar por esta aplicação por ano?

<table>
<thead>
<tr>
<th>Valor</th>
<th>Número de respostas</th>
<th>Percentagem</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-5€</td>
<td>11</td>
<td>73%</td>
</tr>
<tr>
<td>5-10€</td>
<td>3</td>
<td>20%</td>
</tr>
<tr>
<td>10€&lt;</td>
<td>1</td>
<td>7%</td>
</tr>
</tbody>
</table>

Número de respostas diárias

![Número de respostas diárias](https://docs.google.com/forms/d/13O0WVbdDSD/8acFCYC1xGK/U5U/LgN7hy6I_ge0NULK/viewanalytics)

Figure A.21: Mobile application survey
A.6 Mobile Application Screenshots

Figure A.22: Screens. a) Main menu, b) New ride and c) Team menu
Figure A.23: Screens. a) Profile menu, b) Settings and c) Action bar - quick actions

Figure A.24: Action bar - choose team
A.7 Adapter
Figure A.25: Illustration of the adaptor used for the smartphone
Bibliography


[17] Shu Liu and Aaron Striegel. Accurate extraction of face-to-face proximity using smartphones and bluetooth.


[30] Jan Magne Tjensvold. Comparison of the ieee 802.11, 802.15.1, 802.15.4 and 802.15.6 wireless standards. 2007.


[33] Forrest W. Young. Torgerson’s classical mds derivation.
