Universidade de Aveiro
2011
Departamento de Electrónica, Telecomunicações e Informática

Carlos Miguel Cardoso Pereira

Sistema de localização utilizando redes sem fios

Carlos Miguel Cardoso Pereira

Sistema de localização utilizando redes sem fios

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau Mestre em Engenharia Electrónica e Telecomunicações (Mestrado Integrado), realizada sob a orientação científica do Dr. Nuno Borges de Carvalho, Professor Associado com Agregação do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedicatória Dedico este trabalho à minha família e amigos pelo apoio incondicional.

O júri

Presidente	**Prof. Dr. João Nuno Pimentel da Silva Matos** Professor Associado da Universidade de Aveiro
Vogais	**Prof. Dr. Rafael Ferreira da Silva Caldeirinha** Professor do Dep. De Engª Electrotécnica da Esc. Sup. de Tecnologia e Gestão do Instituto Politécnico de Leiria (Arguente Principal)
	Prof. Dr. Nuno Miguel Gonçalves Borges de Carvalho Professor Associado com Agregação da Universidade de Aveiro (Orientador)

Agradecimentos

Gostaria de agradecer a todas as pessoas que directa ou indirectamente me ajudaram e apoiaram ao longo deste longo percurso académico e que foram determinantes para a realização deste trabalho.

Ao meu orientador, Professor Nuno Borges de Carvalho pela possibilidade de realização deste trabalho, pelas suas ideias e motivação que em muito me ajudaram desde o início até à conclusão deste projecto.

Aos meus amigos que estiveram sempre presentes e que me ajudaram nos melhores e nos piores momentos, dos quais destaco o Tiago Carrão, a Carina Ferreira, o Fábio Ferreira, a Joana Rodrigues, o André Fonseca e o João Paulo. Uma palavra também para o Ludimar Guenda que mais recentemente contribuiu para a realização deste trabalho e ao qual agradeço.

Aos meus pais e irmão que me apoiaram todos os dias nesta grande aventura e que sem os quais nada disto seria possível.

Palavras-chave Localização, Redes sem fios, GPS, *Android*, *Google Maps*, Locate Me

Resumo

Com a evolução da tecnologia foi possível obter dispositivos móveis com uma grande capacidade de processamento assim como uma vasta variedade de funcionalidades. Um simples dispositivo pode conter um módulo de GPS, ligações de dados distintas (*Wi-Fi*, *GPRS*, *HSDPA*, *Bluetooth*), um processador com um elevado desempenho e vários tipos de memórias para armazenamento de dados. Tudo isto com autonomias de bateria bastante aceitáveis.

Tendo isto em conta, tem-se como objectivo desta dissertação projectar e desenvolver um sistema que permita a localização de pessoas estejam elas em ambientes interiores ou exteriores.

No caso de ambientes exteriores será utilizado o módulo GPS dos dispositivos.

Em ambientes interiores pretende-se desenvolver uma forma alternativa de localização baseada em redes sem fios.

Keywords Location, Wireless Networks, GPS, *Android*, *Google Maps*, Locate Me

Abstract Due to the technological evolution it is now possible to have mobile devices with great processing power, as well as a wide range of features. A simple device may have a GPS module, distinct data connections (such as Wi-Fi, GPRS, HSDPA, Bluetooth), a high performance processor, and several memory types. All of this with a very reasonable battery life.
Taking this into consideration, the objective of this dissertation is to design and develop a system that enables the localization of people, whether they are outdoors or indoors.
In outdoor environments it will be used the GPS module of the devices.
In case of indoor environments, the idea is to study and implement an alternative way of localization based on wireless networks.

CONTENT

FIGURES INDEX

TABLES INDEX

ACRONYMS LIST

AP – Access Point

API – Application Programming Interface

CEO – Chief Executive Officer

GPS – Global Positioning System

HSDPA – High-Speed Downlink Packet Access

LCD – Liquid Cristal Display

OS – Operating System

PCB – Printed Circuit Board

PDA – Personal Digital Assistant

RAM – Random Access Memory

RSSI – Received Signal Strength Indication

SDK – Software Development Kit

SSE - Sum of Squares due to Error

SSID - Service Set Identifier

TCP – Transmission Control Protocol

UART – Universal Asynchronous Receiver/Transmitter

UDP – User Datagram Protocol

USA – United States of America

USB – Universal Serial Bus

USD – United States Dollar

Wi-Fi – Wireless Fidelity

WSN – Wireless Sensor Networks

1. INTRODUCTION

1.1. Motivation

The localization of people and objects was always a demand of the population. For recreational, professional or even for emergency purposes there is a wide range of reasons to locate or to be located.

At the moment, the Global Positioning System (GPS) is the most used and well known localization method, but it has some weaknesses, which will be enumerated ahead in this work.

Furthermore, the smartphone's market has been growing greatly over the past recent years. Nowadays, nearly all smartphones have GPS[4], Wi-Fi[5] and Bluetooth[6] modules so there is a wide range of applications for these devices. Their high-performance processors combined with good battery autonomies make them excellent tools for mobile computing.

A cell phone is no longer just a phone. There are lots of other useful features embedded on the same device. The evolution can be seen in Figure 1.

Figure 1 – Mobile phones evolution[7]

In this thesis it will be made an attempt to take advantage of those functionalities and develop an integrated localization system using different methods simultaneously. It should work in outdoor and indoor environments.

For outdoor localization it will be used the existent GPS system so, the main focus of this work will be over the indoor localization and how to acquire precise locations using the wireless communications modules of the smartphones.

The project has a hardware section to design a Bluetooth gateway to allow communications between smartphones and other peripherals. In this case it is used to communicate with SimpliciTI[8] modules from Texas Instruments and provides additional features to the localization system.

1.2. Objectives

This thesis aims to:

1 – Study the current localization systems;

2 – Identify the weaknesses of the current localization methods and come up with alternative ways;

3 – Test the designed methods;

4 – Implement a functional location based application to run on mobile devices (smartphones) as well as the backbone systems to support the application;

5 – Design and implement a gateway between Bluetooth and Wireless Sensor Networks (WSN);

6 – Develop a localization protocol to implement on CC1110 Texas Instruments modules;

7 – Check the results and get conclusions.

1.3. Thesis Structure

This thesis is organized as follows:

Chapter 2 shows the current state of the technology in mobile computing, how was the evolution and the alternative options in the market. In this chapter, it is also mentioned some localization methods published by other authors as well as their advantages and disadvantages.

Chapter 3 presents the different operating systems (OS) for mobile devices. Their strengths and their weaknesses are compared and discussed. The ones taken into account are: Windows Mobile, iPhone OS, Symbian and Android. The Android is particularly explored because it is the one used in this work.

In Chapter 4, three new proposals of localization algorithms are presented for situations when there is no GPS signal available. They are explained and exemplified using practical cases.

Chapter 5 describes Locate Me, a location based application developed to implement the localization algorithm of chapter 4.

Chapter 6 is about the server operation and how it stores the data related to the users.

Chapter 7 explains the communication protocol between the server and the mobile devices. The frames structures for each function are shown here.

Chapter 8 presents the designed hardware for this project and how it operates.

Chapter 9 is the combination of the results of the application "Locate Me" and the hardware side of this project. It shows how the application can interact with the Bluetooth module.

Chapter 10 presents the project results, the conclusions and makes some suggestions for future work.

1.4. Contributions

As the result of this thesis the following paper was accepted to be presented on *2011 International Conference on Indoor Positioning and Indoor Navigation (IPIN):*

Carlos Pereira, Ludimar Guenda, Nuno Borges de Carvalho, "A Smart-Phone Indoor/Outdoor Localization System", 2011 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Guimarães, Portugal, September 2011

2. STATE OF THE ART

2.1. Mobile computing and the Smartphones

In the last two centuries, mobility has been redefined. Both physical and virtual objects are now mobile. Mobility of physical objects related to movement of matters, whereas movements of virtual objects relate to movements of bits and bytes[9].

The devices for mobile computing and mobile communication are in constant evolution and they are converging to the same device. This has started with the release of the first mobile computer: The Osborne I (Figure 2) in March of 1981. It has 64kB of RAM and a 5 inches monochromatic display. It also has two diskette drives (with 5.25 inches). It was not completely portable since its weight was 10kg. Another very important moment in mobile computation history was Apple's Macintosh (Figure 4) launch in the beginning of 1984. It was released with an audio processor and a user graphical interface, two revolutionary features in that time.

Figure 2 - Osborne I [1]

Figure 4 - Apple Macintosh [2]

Figure 3 - iPhone I [3]

In 1991, Jeff Hawkins designed the first "pen computer" for industrial purposes and in 1992 John Sculley, Apple's CEO, introduced the concept of "Personal Digital Assistant" (PDA). It was a sales failure, but other solutions appeared on the market quickly. In 2001, the first smartphone was released, the Kyocera from Palm.

The smartphones started getting more features and in 2007, Apple launched the first iPhone (Figure 3). It was a revolutionary device, because it joined a phone, an iPod[10] and Internet all-in-one smartphone. In the end of the same year the Open Handset Alliance was founded in order to develop a new mobile platform, Android[11].

Nowadays, the iOS and the Android running smartphones are leading the market all around the World (see appendix A).

2.2. Today's localization methods

On the present days, there are some localization systems used to acquire the location of people and equipment. All of those systems have advantages but also disadvantages, there is no perfect localization system. Some systems work better in outdoor environments, but they fail in indoor. Other systems are specifically designed for indoor environments, but they need an infrastructure in the building.

2.2.1. Global Positioning System (GPS)

The most known system is the Global Positioning System. It was originally developed in 1973 by the U.S.A Department of Defense with the primary objective of satisfying military navigation requirements. The secondary objective was to provide a separate, less accurate signal for both military and civilian use. This signal was intentionally degraded in accuracy to avoid its exploitation by potentially unfriendly users.

The GPS is based on a constellation of 24 Earth-orbiting satellites, which transmit radio signals with the satellite's position and the time it transmitted the signal. The distance between a satellite and a receiver can be computed by subtracting the time that the signal left the satellite and the time that it arrives at the receiver. If the distance to four or more satellites is measured, then a three-dimensional position on Earth can be determined [12].

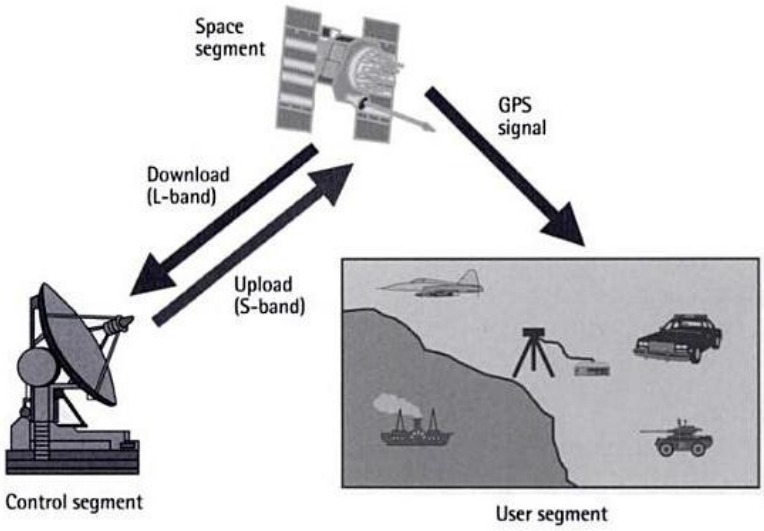


Figure 5 - GPS segments [4]

There are three important segments in the GPS: the space segment, the control segment and the user segment (Figure 5). The space consists of the 24-satellite constellation. The control segment consists on a worldwide network of tracking stations, with a master control station (MCS) located in the USA at Colorado Springs, Colorado. The primary task of the operational control segment is tracking the GPS satellites in order to determine and predict the satellite locations, the system integrity, the behavior of the satellite atomic clocks, the atmospheric data, the satellite almanac, and other considerations. This information is then packed and uploaded into the GPS satellites through a S-band link.

The user segment includes all military and civilian users. With a GPS receiver connected to a GPS antenna, a user can receive the GPS signals, which can be used to determine his or her position “anywhere” in the world[4].

2.2.2. Wi-Fi based localization

The decreasing cost and the proliferation of Wi-Fi access points have made them very attractive to use on indoor localization. Most of the existent techniques require an offline training phase to build a map overlaying the received signal strength on the physical space. These methods have some limitations due to the environment changes.

There are several papers published on this topic. On [13] it is shown how it is possible to build a database with geo-referenced roadside Wi-Fi access points using directional antennas on the top of a car. It is used by Google’s Street View cars: when they are collecting pictures from streets, they are also collecting the roadside access points.

The literature presents a wide range of localization algorithms but most of them need the exact location of the Wi-Fi access points. In this work it will be presented a new one where this information is not necessary.

2.2.3. Cell based localization

The cell based localization operates exactly the same way as the Wi-Fi based localization (2.2.2) but using the detected mobile base stations instead of the Wi-Fi access points. The coverage area of a base station is much larger than the access point’ area and therefore the localization errors will be far bigger.

2.2.4. Wireless Sensor Networks

Wireless sensor networks consist of a large number of small, resource-constrained computing devices that are distributed in the environment and communicate over a wireless interface[14]. The nodes can use integrated sensors to monitor certain aspects of their environment and cooperate with each other in fulfilling of a joint sensing task.

Localization is one of the fundamental research problems of wireless sensor networks and it has received a considerable amount of attention by many research groups in previous years. There are many localization approaches for wireless sensor networks but the biggest part of them needs a support infrastructure to operate. They use nodes strategically positioned that will work as references to acquire the localization of the mobile modules. The different approaches for localization using WSN differ on the method how the mobile modules location is acquired from the reference nodes. Some systems just assume the location of the reference node with the greater RSSI as the mobile modules location, others use the RSSI to calculate a distance to the reference node[15]. There are systems with complex algorithms which collect data from more than one reference node and use triangulation techniques to get precise locations. They can also use other measurement techniques besides the RSSI, for example the time of arrival (TOA) to reduce the interference errors.

2.3. Localization and the Social Networks

Nowadays, the social networks play an important role in people's lives. Almost everyone has at least one account in a social network. For example, Facebook has already more than 500 million people. If we take into account that the World's population is around 6000 million people it means that one person out of twelve people has an account on Facebook. And this is just regarding Facebook, as there are lots of other social networks on the Internet.

People want to share their information with their friends, family, colleagues and even people they do not know. This information can be not only their name and age, but also photos, social status or current location. Considering this, Google released a mobile application, "Google Latitude"[16], where the users can add their friends and see where they are.

Google Latitude uses GPS and wireless networks to acquire their locations but it has some limitations. There is no possibility to specify the users locations when they are indoor and thereby obtain room-level localization. It also does not allow the upload of building plans to overlay on maps.

Figure 6 - Google Latitude view

Figure 6 shows an example of Google Latitude operating using wireless networks to acquire the location (the device was inside the building).

3. MOBILE OPERATING SYSTEMS

At the moment, there are some operating systems available on the market having as target the mobile devices. The competition between the manufacturers is quite severe and the market shares are always changing. It is not the same for every region in the globe and the market leader differs with the geographic region. For example, in USA the leaders are the iOS and the Blackberry OS but, on other hand, in China, the Android OS is the prominent leader. These stats can be seen in appendix A.

Usually, these types of operating systems run in different devices from different manufacturers but there is an exception: the iOS only runs on the iPhone devices.

In this chapter will be explained the strengths and the weaknesses of some operating systems. The Android OS will have a greater relevance because it is the one studied during this project.

3.1. Android OS

Android OS was originally created by a group of 80 technology and mobile companies that founded the Open Handset Alliance. Together they have developed "the first complete, open and free mobile platform"[17].

The OS was built the ground-up to enable developers to create compelling mobile applications that take full advantage of all features available on the devices. It was built to be completely open. This means that a simple application can call any of the phone's core functionality such as making calls, sending messages or using the camera. Android is built on the open Linux Kernel. In addition, it uses a custom virtual machine especially designed to optimize memory and hardware resources in a mobile environment. Android is open source and as a result of that, it can be incorporated in a large number of devices, not only smartphones but also in tablets or other new cutting edge technologies as they emerge.

The device's core applications and third-party applications are not differentiated. They can all be built to have equal permissions to the device's capabilities providing users with a broad spectrum of applications and services.

Android provides access to a wide range of useful libraries and tools that can be used to enrich the applications. For example, Android enables the developers to obtain the location of the device, and allows devices to communicate with others enabling excellent peer-to-peer social applications. In order to publish their applications on the official Android Market, the developers have a registering fee of $25 USD.

The graphical environment has not been forgotten and the appearance of the operating system has been improved. It also depends on the device's manufacturer, but the general appearance is quite similar. Here is an example of a device running Android 3.0 (Figure 7):

Figure 7 - Capture of an Android 3.0 screen[18]

There is an official application market called *Android Market* where the users can easily search and download applications to the device. Those applications can be free or paid in some cases. In the developers' point of view it is easy to register and publish on the market. The applications are not validated before releasing to the market, so every application published by a developer will be available to download.

3.2. Windows Mobile OS

Windows Mobile is a mobile operating system developed by Microsoft to be used in smartphones and mobile devices. It originally appeared as the Pocket PC 2000 operating system and most devices came with a stylus pen, which is used to enter commands by tapping on the screen. After that, other versions of windows mobile have been released with some improvements.

Recently, Microsoft announced a completely new phone platform, the Windows Phone 7. This new platform does not have backward compatibility. *With the move to capacitive touch screens, away from the stylus, and the moves to some of the hardware choices we made for the Windows Phone 7 experience, we had to break application compatibility with Windows Mobile 6.5*[19], as Terry Myerson, corporate vice president of Windows Phone engineering, once said.

The Windows Phone 7 OS is not open source but there are many expectations around it. The full integration with social networks is other advantage of this OS. Several features of Windows Phone 7 are organized into “hubs”, which combine local and online content. For example, the Pictures hub shows the photos captured with de device’s camera and the user’s Facebook photo albums.

Here is an example of a Windows Phone 7 screen capture (Figure 8):

Figure 8 - Windows Phone 7 screen[20]

In order to publish applications on *APP HUB* (official Windows Phone 7 and XBOX 360 market), the developers have an annual subscription fee of $99 USD. There is a limit of 100 submissions; additional submissions are $19.99 USD per submission[21].

3.3. iOS

The iOS (known as iPhone OS prior to June 2010) is Apple's mobile operating system. It was originally developed for the iPhone, but nowadays it is used in other devices such the iPod touch, iPad and Apple TV. Apple does not license iOS for installation on third-party hardware.

The user interface of iOS is simple and intuitive. In Apple's official web page they wrote: *The first time you pick up an iPhone, you know how to use it. That's because the revolutionary Multi-Touch interface in iOS was designed for the most natural pointing device ever: your finger*[22].

Here is an example of the iOS 4 (Figure 9):

Figure 9 - iPhone 4 running iOS[22]

This operating system is now on the fourth version and it is keeping the backward compatibility. In the official application store are now available more than 350 thousands applications[22]. It is a market that is growing rapidly.

One of the main advantages of this system is the design. It is carefully planned to give the best experience to the user. On the other hand, the features available to the developers (libraries and tool) are too much limited.

To be an iOS application developer it is necessary an Apple computer, running Mac OS and an annual subscription fee of $99 USD. The applications need to be approved by Apple to be available to download.

3.4. Others

The mobile market is very competitive and there are other mobile operating systems available. The Blackberry OS is one those operating systems. It is very popular in USA and it is starting to appear in Europe. The main target of Blackberry OS is the business sector with special attention to the email services. The most part of the Blackberry devices have complete keyboards. The following figures show some Blackberry examples (Figure 10):

Figure 10 - Blackberry devices: Torch 9800, Curve 8900, Curve 8520 and Storm 9500[23]

Other important mobile operating system is Symbian OS. It was originally created in 1998 by six manufacturers (Nokia, Siemens, Samsung, Ericsson, Sony Ericsson and Panassonic) but it is currently maintained by Nokia. The biggest concern of Symbian OS is to minimize the phone's resources waste and with its last version that was reduced in 30% [24]. It is still being used in Nokia smartphones, but recently Nokia and Microsoft have made a commitment where they agreed that Nokia devices will be shipping Windows Phone 7. Some Symbian phones are shown on the following figure:

Figure 11 - Symbian devices (from Nokia)[25]

Samsung bada is another plataform for smartphones created in 2010 by Samsung. "bada" is a Korean word that means "ocean" and "seashore". It is called that way because *"bada" in itself embodies the open possibilities of the ocean: it can accommodate the various applications created by developers and it provides an interesting new space that offers unprecedented enjoyment to its users*[26]. The developers have a software development kit (SDK) available to build new applications, but it is bit limited. There are few applications running on this operating system

(when compared with the diversity of other OS). On the present date, Samsung has six smartphones running bada. Here are some examples (Figure 12):

Figure 12 - Examples of smartphones running bada OS[27]

There are other operating systems for smartphones on the market, but they have much less visibility or they are developed specifically for a smartphone. The ones explained before are market leaders.

3.5. Mobile operating systems comparison

All operating systems have advantages and disadvantages (see appendix B), but the Android OS stood out because of its features and devices' prices. It is possible to buy a smartphone running Android OS with GPS, Wi-Fi, Bluetooth and a good processor by a reduced price (less than 200€). Another upside of Android OS is the open SDK. There is a wide range of libraries available to interact with all features of the device and these tools are free. The low fee ($25 USD) to register as an Android developer was also considered when choosing the OS.

To sum up, Android OS was chosen as the platform for this project because of the overall lower cost and the variety of development tools.

4. LOCALIZATION ALGORITHMS

The purpose of this chapter is to explain a localization method for outdoor and indoor environments. It should be able to get the location of a device even when there is no GPS signal. The method was designed to acquire the location of smartphones by taking advantage of their connectivity features. Nowadays, smartphones have GPS, Wi-Fi and Bluetooth modules, so they can be used to get accurate locations.

In order to answer to a localization request, the system uses the following sequence, represented on Figure 13:

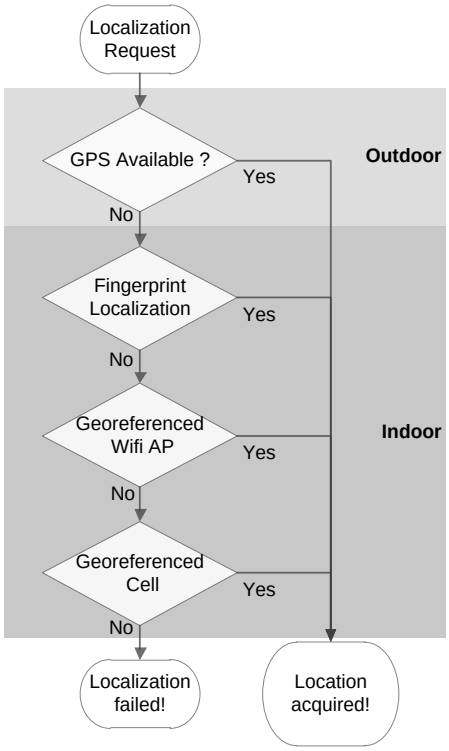


Figure 13 - Localization sequence diagram

4.1. Global Positioning System (GPS)

The GPS is the first method in the priority list to acquire the device's location. If the device is outdoor and there is no buildings around it will be able to get a precise location. However, it is not always true, so some alternative methods of localization will be purposed in this chapter.

4.2. Fingerprint Localization

The fingerprint localization is a technique that analyses the device's surrounding environment and tries to find a good location match with a known place in the database. This database has to be previously populated with places fingerprints.

4.2.1. Fingerprint generation

The fingerprints are generated by performing a Wi-Fi scan and associating the result to the geographical coordinates where the scan was made. In this case, the system stores the following info for each place:

- Name;
- Description;
- Floor;
- Latitude;
- Longitude;
- The user who add the fingerprint;
- The list of detected AP (SSID, MAC and RSSI);

The name, description and floor are introduced by the user in a form; the latitude and the longitude are also introduced by the user, but via map pointing. The other data is automatically associated by the device.

4.2.2. Fingerprint matching algorithm

The fingerprint matching algorithm uses the result of a Wi-Fi scan to get the device's location. This algorithm is explained bellow using an example of a Wi-Fi scan result with 3 APs:

- AP_1 with RSSI_1
- AP_2 with RSSI_2
- AP_3 with RSSI_3

} with RSSI_1 > RSSI_2 > RSSI_3

The algorithm starts by sorting the APs by RSSI (in this example they are already sorted), and then it uses the diagram of Figure 17 to get the best match for that environment fingerprint.

Figure 15 - Database query for fingerprint localization

Figure 14 - Access points' set for a database query

The block of Figure 15 represents a database query looking for a place match with all of the access points of the previous APs' set (Figure 14).

As a result of an implementation of this algorithm it is possible to distinguish a different number of places depending on the number of detected access points as shown on Figure 16:

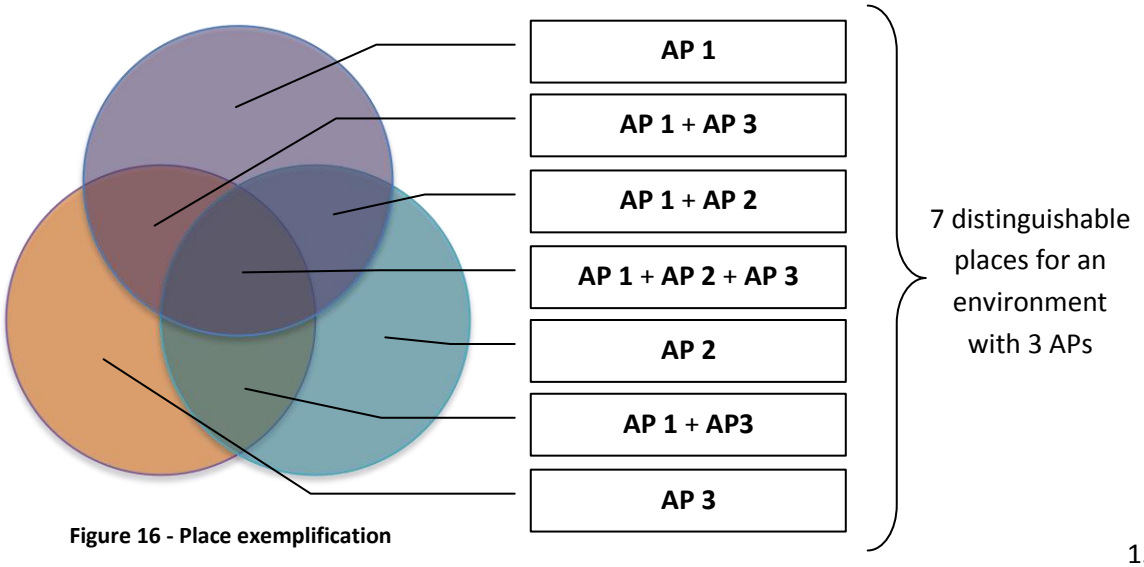


Figure 16 - Place exemplification

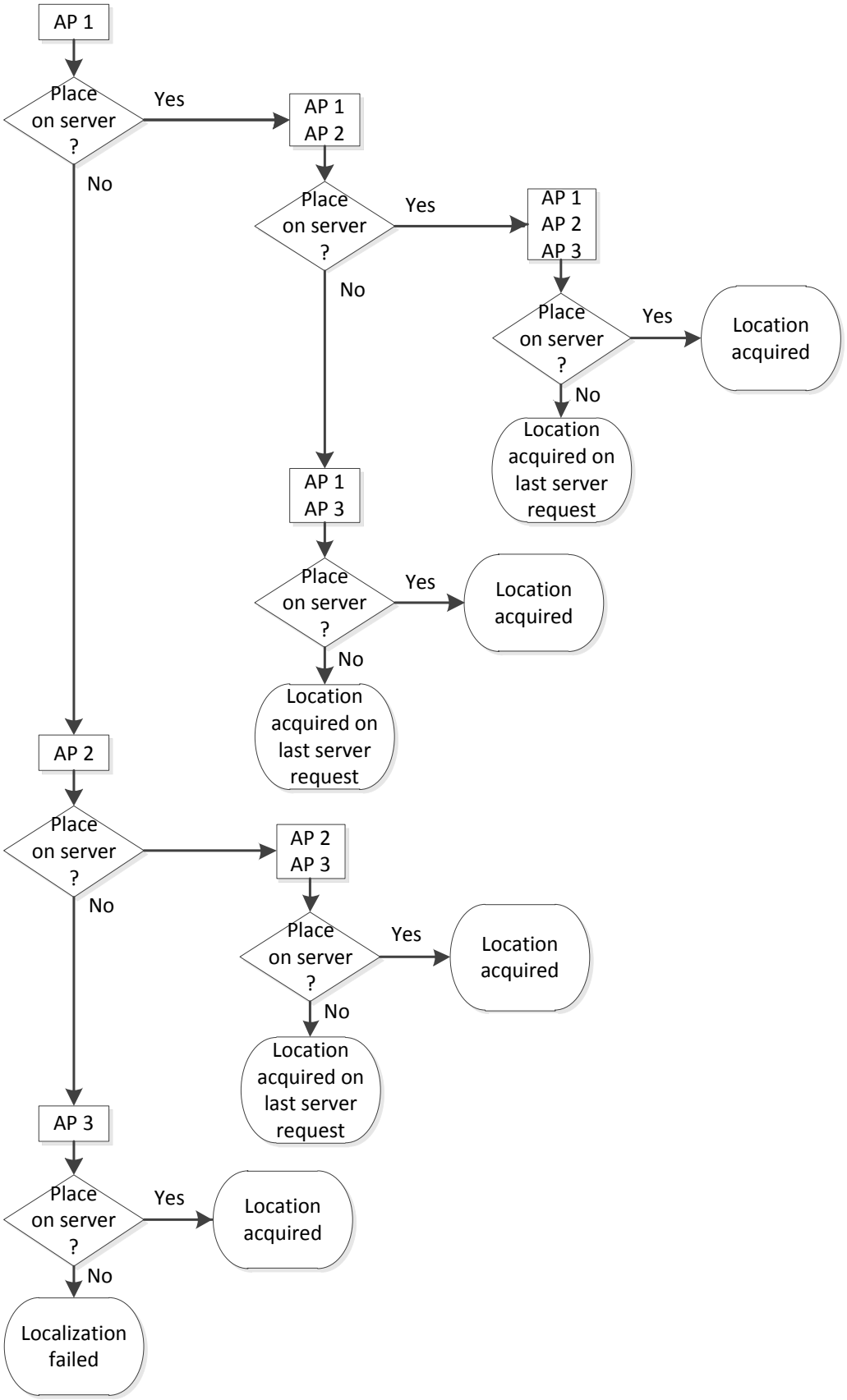


Figure 17 - Fingerprint matching algorithm

4.3. Localization based on geo-referenced Wi-Fi access points

The localization based on geo-referenced Wi-Fi access points takes advantage of the GPS and Wi-Fi modules of the devices to build a database which can be used later when the GPS signal is not available.

When it is possible to the GPS module to acquire the device location, the Wi-Fi module performs a scan for access points and the result is sent to the server associated with the GPS coordinates. Once on the server, there are two alternatives:

- **The AP is not in the database:** It will be added to the table with the received GPS coordinates;

- **The AP is already in the database:** The new coordinates where the AP was detected will be used on the following formula (4-3) in order to obtain a better location:

$$new\_upds = old\_upds + 1 \qquad (4\text{-}1)$$

$$new_{mean_{rssi}} = \frac{old_{upds} \cdot old_{mean_{rssi}} + rssi}{new_{upds}} \qquad (4\text{-}2)$$

$$new\_pos = \frac{old\_pos \cdot old\_mean\_rssi \cdot old\_upds + pos \cdot rssi}{old\_upds \cdot old\_mean\_rssi + rssi} \qquad (4\text{-}3)$$

old_pos – last calculated position of the access point;

pos – last position where the access point has been detected;

new_pos – new position of the access point;

old_mean_rssi – mean value of the received signal strength;

$rssi$ – last received signal strength of the access point;

new_mean_rssi – new mean value to the received strength;

old_upds – number of updates already made;

new_upds – number of updates already made plus the current update;

The values of RSSI are previously converted from logarithmic units (dBm) to linear units (mW) and this algorithm is applied on latitude and longitude values. With this method the coordinates in the database converge to the real location as shown on Figure 18.

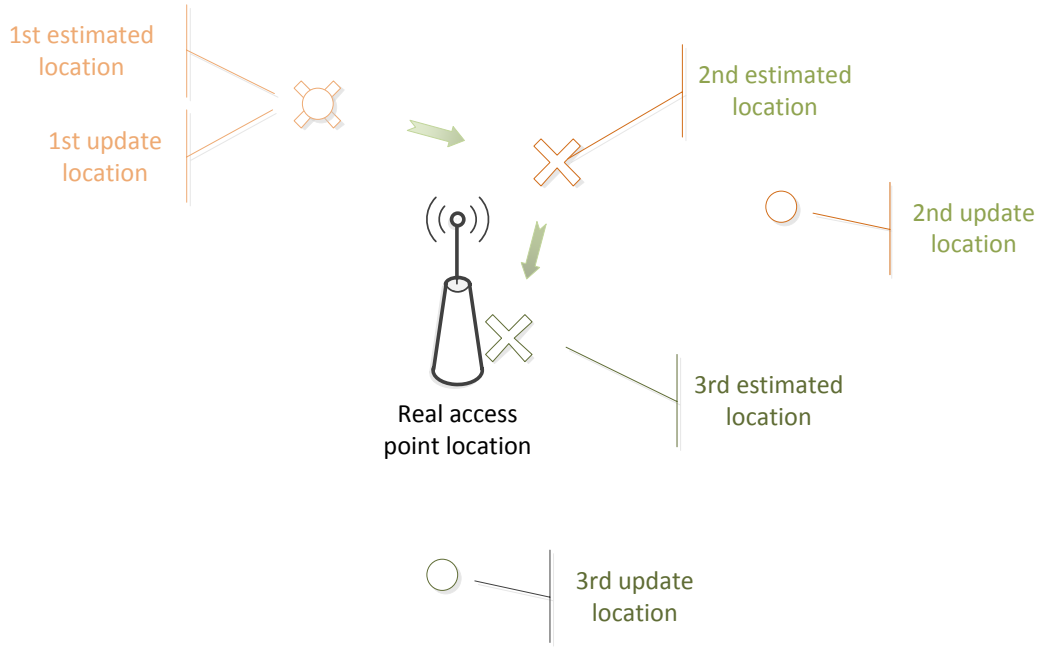


Figure 18 – Geo-referencesd Wi-Fi access points algorithm exemplification

4.4. Localization based on geo-referenced mobile cells

The localization based on geo-referenced mobile cells operates exactly the same way of the localization based on geo-referenced Wi-Fi access points and uses the same algorithm. In this case the mobile cells are used as reference. This localization method returns worst results than the previous one, because the covered area of a mobile base station is usually far bigger (in the order of hundreds of meters) than the covered area of a Wi-Fi access point (in the order of tens of meters).

5.LOCATE ME

5.1. What is it?

"Locate Me" is a location based application for Android smartphones. With this application it is possible to see the location of friends whether they are indoors or outdoors. It is a practical implementation of the localization algorithms explained in chapter 4 (see Figure 20) and uses the Google Maps API for Android to show the localization results of the algorithms. The application uses the diagram of appendix C and it is also able to overlap building plans in case of indoor environments. When the application starts there are four tabs in the main screen (Figure 19): Map Viewer, Friends, Places and Settings.

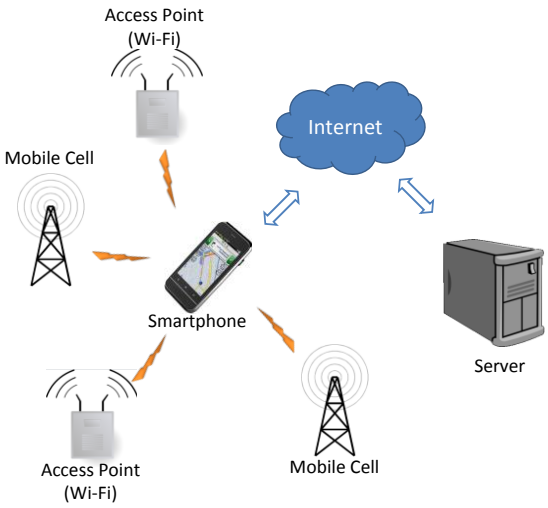


Figure 20 - Locate Me system diagram

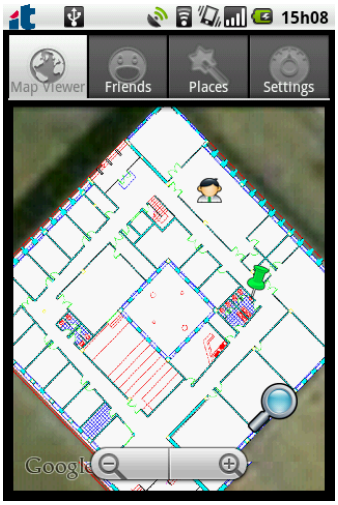


Figure 19 - Locate Me main screen

The Map Viewer shows a map with the current location of the device ( ) and the friends' locations ( ). It has two view types: map view and satellite view. If there is a building plan for a friend location it can be overlapped as shown in Figure 19.

The Friends tab shows the friends already added and it is possible to add new ones or delete them.

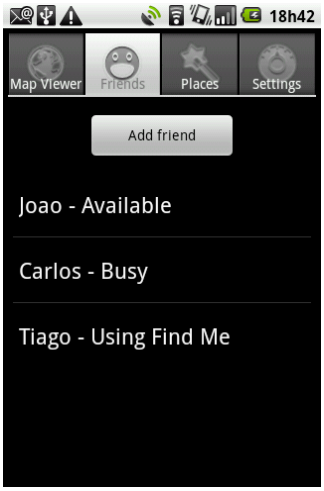


Figure 21 - Friends tab example

The Places tab is used to manually add new locations where there is no GPS signal. It means that the user can point on the map his current location (Figure 22) and the system will perform an environment scan, looking for Wi-Fi access points and mobile base stations. These results are associated to the place description (Figure 23) and they are sent to a server. The next time that a user goes to that place again, it can be localized. This makes use of the fingerprint localization explained on chapter 4.2.

Figure 22 - Pointing a place on the map

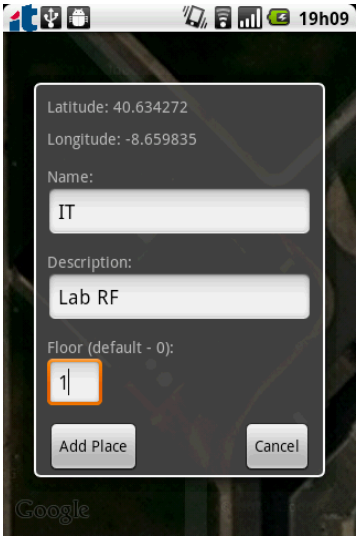


Figure 23 - Place description

The Settings tab allows the users to customize the application with their own configurations (Figure 24). There is a form with four fields: name, status, refresh rate and refresh rate on background. The name and the status are the user info shown to the friends. The refresh rate is the period between localization requests and friends locations updates when Locate Me is the activity in the front. If the activity is running in the background the period used is the one set on refresh rate on background.

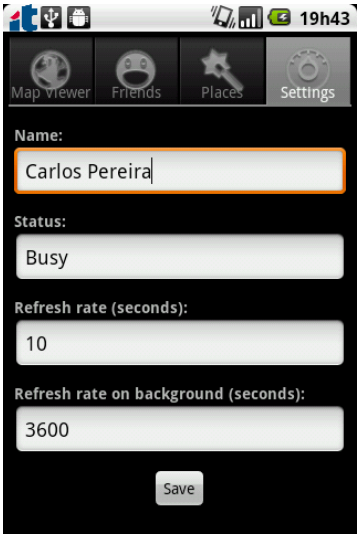


Figure 24 - Settings tab

5.2. How to use it?

Locate Me was designed to be simple and easy to use. The first thing to do before starting is to go to the Android Market, download the application and install it as shown on the following figure (Figure 25):

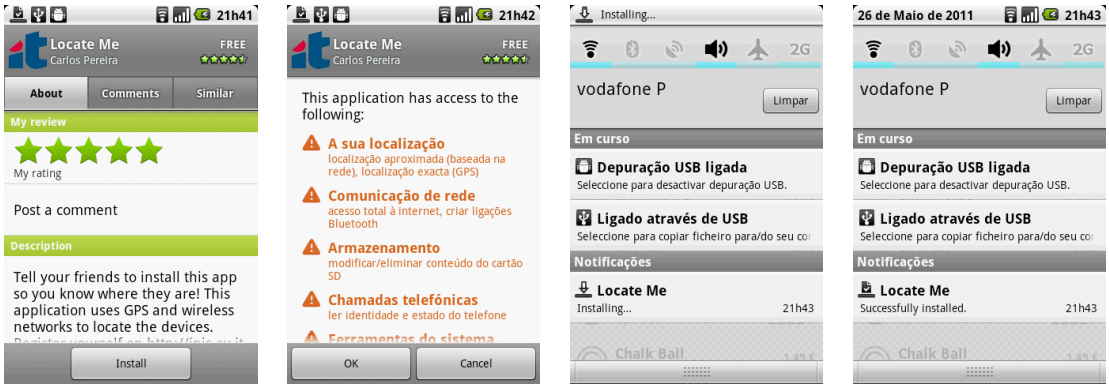


Figure 25 - Installation process

The user is now ready to use the application. When it starts it verifies if the GPS is turned on and if it is not, a dialog is box shown (Figure 26):

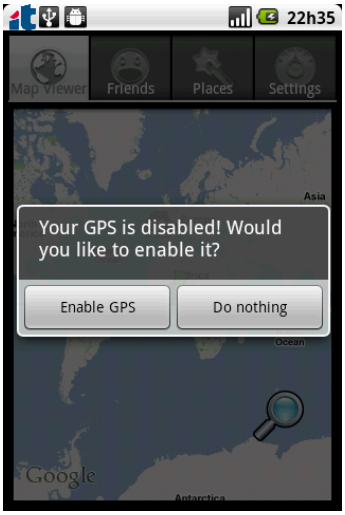


Figure 26 - GPS dialog box

5.2.1. Add Friends

In order to add a friend it is necessary to know the friend's ID. This ID changes with the device and it is available by pressing the menu button and select "Help". There is the ID of that phone and it is now possible to add the friend in other device.

The proceedings to add a friend are very simple:

1st – Go to "Friends" tab;

2nd – Press the button "Add Friend";

3rd – Insert the friend ID;

4th – Press "Add" to confirm.

Here is an example (Figure 27):

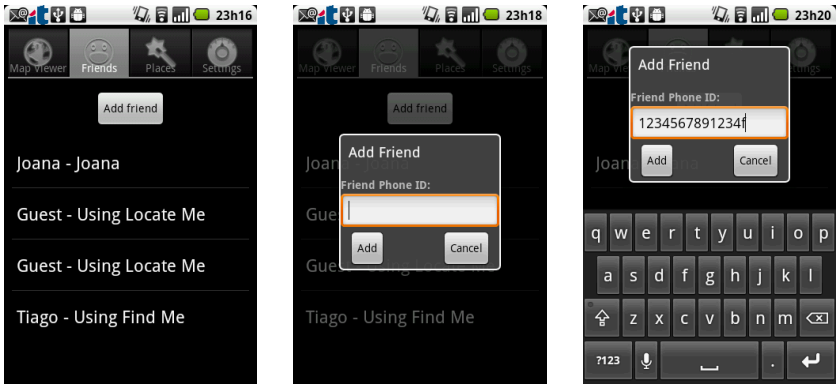


Figure 27 - Add friend example

5.2.2. Delete Friends

This operation is available by selecting a friend on the list. A dialog box will be shown with the "Delete" button.

5.2.3. Add a place for fingerprint localization

The proceedings to add a place for this type of localization are:

1st – Go to "Places" tab;

2nd – Press "Map It";

3rd – Navigate on the map to the current location and press it once;

4th – Verify if the circle is in the right place and if it is there, touch the circle again;

5th – Fill in the form and press "Add Place".

Example (Figure 28):

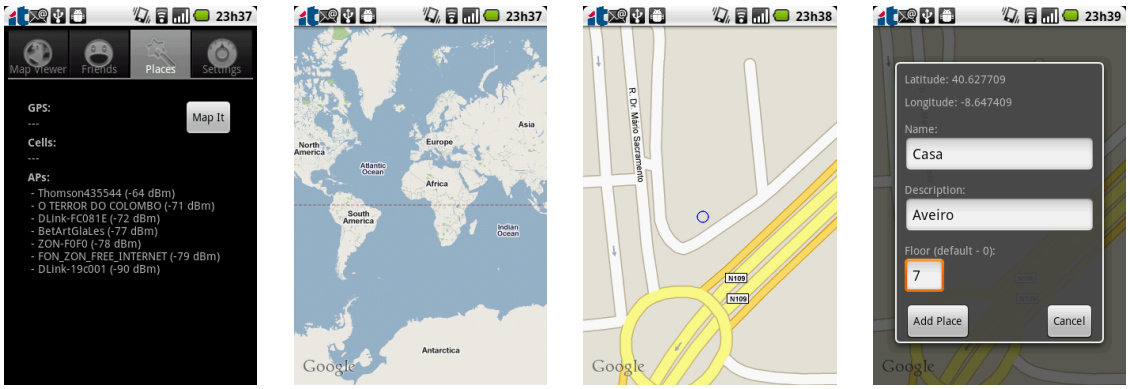


Figure 28 - Add place example

5.2.4. Customize Locate Me

In the "Settings" tab it is possible to choose the name and the status to be shown on the other devices.

5.2.5. See the historic positions

Locate Me is integrated with a web site[28] where the users can see the historic positions of their devices. This web site was not developed within this project and this integration was made as an additional feature to the system.

The users must register on the web site and add their devices' IDs to the system.

Proceedings:

1st – Registering on the site (Figure 29):

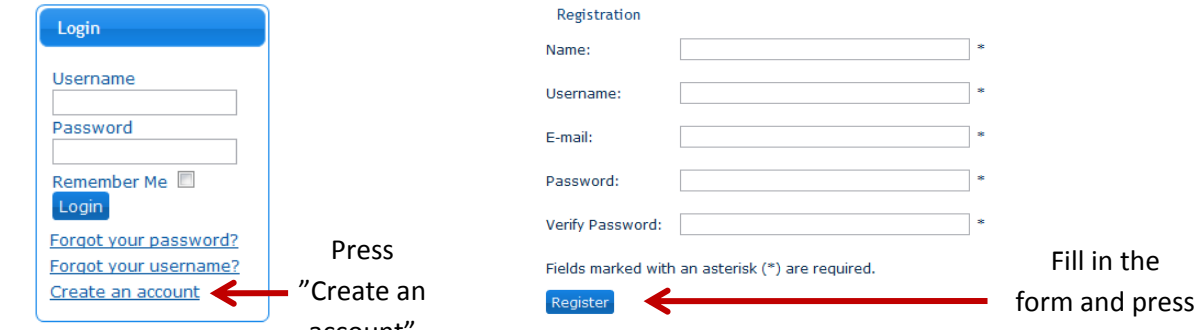


Figure 29 - Registering steps

2nd – Login with your account

3rd – Navigate to "localize" (link on the top of the web page)

4th – Add your devices to your account (Figure 30):

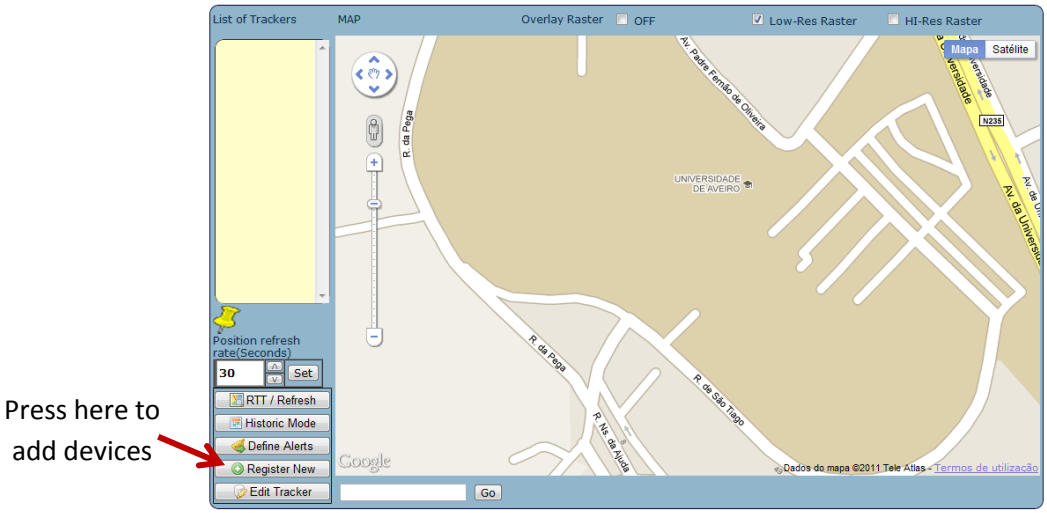


Figure 30 - Add devices to the web site account

6. SERVER OPERATION

In order to store the information related to the localization system, it was developed a server application, which works together with a MySQL database. It was developed in Java and its main objective is to wait for requests from the Locate Me, process them and if necessary answer them.

The algorithms explained on Chapter 4 are implemented here because they need to make several requests to the database and it will be faster if both the server application and the database are running on the same machine.

The server has a range of functions to interact with the database and answer user requests:

- **Wi-Fi Update** → Used to add or update a geo-referenced AP. If the AP is already in the database it is updated using the algorithm explained on chapter 4 for geo-referenced Aps. If not, a new entry in the table will created.
- **Wi-Fi Request** → Receives as an input a MAC address and returns the coordinates of the AP if they are available.
- **Cell Update** → Works as the Wi-Fi Update but instead of using APs, it uses mobile base stations.
- **Cell Request** → Receives as an input a Cell ID of a mobile base station and returns its geographical coordinates if available.
- **User Update** → This function is called to update the users information. It could be the name, the status and/or the location.
- **Get Friends Locations** → Returns the information available about the friends of a user specified as input.
- **Search For Place** → This function is a practical implementation of the fingerprint localization explained on chapter 4. It receives a list of detected APs and tries to find a location match in the database.
- **Add Place** → Adds a new location to the database. It can be used later with the fingerprint localization.
- **Search For Plan Info** → Search in the database for a plan that matches with location given as an input and returns all the info about that plan, except the image file.
- **Get Plan Image** → Returns the image file of the specified plan.
- **User Add Friend** → Adds a friend relation between two users.
- **User Del Friend** → Removes a friend relation between two users.

Those functions can be called trough the UDP socket listening on port 9420 and using the protocol described on the chapter 7.4.

6.1. Database Structure

The supporting database for this system is organized in tables with some relations between them. As a result of the database sharing with a web site [28], there are several columns in the tables which are not directly used by this system.

The database is composed by three groups of tables: the users data, the localization data and the plans data. The **users data group** has three tables (Table 1, Table 2 and Table 3) and they are arranged as follows:

➔ "trackers" (used to store the users data)

Field	Type	Comments
Idx	int(5)	User index
Name	varchar(30)	User name
Status	varchar(150)	User status
TrackerID	char(14)	Device ID
img	varchar(35)	*
rate	int(11)	*
doKml	bit(1)	*

* - Field not used in this system

Table 1 - trackers structure

➔ "positions" (used to store the users' locations historic)

Field	Type	Comments
Device_id	char(14)	Device ID
Date	datetime	Location date
Latitude	double	Latitude (degrees)
Longitude	double	Longitude (degrees)
Speed	int(3)	Speed (by GPS)
Heading	int(3)	Heading (by GPS)
hasPic	varchar(15)	*
floor	int(4)	Floor (fingerprint localization)

* - Field not used in this system

Table 2 - positions structure

➔ "user_friends" (used to store the friendship relations between users; these relations are bi-directional: the user 1 is friend of the user 2 and the user 2 is friend of user 1)

Field	Type	Comments
person1	varchar(15)	User 1
person2	varchar(15)	User 2

Table 3 - user_friends structure

The **localization data group** is divided in 3 subsections: geo-referenced access points data, geo-referenced cells data and fingerprinting data. The first two are constituted by one table each (Table 4 and Table 5 respectively):

➔ "wifi" (used to store the geo-referenced access points data)

Field	Type	Comments
ID	int(11)	AP index
SSID	varchar(32)	Network name
MAC	varchar(12)	AP physical address
updates	int(7)	Number of updates already made
latitude	int(9)	Estimated latitude
longitude	int(10)	Estimated longitude
mean_RSSI	int(4)	Average RSSI
upgradable	int(1)	Upgradable coordinates flag

Table 4 - wifi structure

➔ "cell" (used to store the geo_referenced mobile cells data)

Field	Type	Comments
ID	int(11)	Cell index
cell_id	int(11)	Cell ID
updates	int(7)	Number of updates already made
latitude	int(9)	Estimated latitude
longitude	int(10)	Estimated longitude

Field	Type	Comments
mean_RSSI	int(4)	Average RSSI
LAC	int(7)	Local Area Code
upgradable	int(1)	Upgradable coordinates flag

Table 5 - cell structure

The third section of the localization data group is constituted by three tables related by the place index (Table 6, Table 7 and Table 8):

➔ "places" (used to store the info of each place)

Field	Type	Comments
ID	int(8)	Place index
Name	varchar(30)	Place name
Description	varchar(150)	Place description
Latitude	int(9)	Latitude inserted by the user
Longitude	int(10)	Longitude inserted by the user
Floor	int(4)	Place floor
Uploader	varchar(15)	User who added the place
Date	datetime	Date when the place was added

Table 6 - places structure

➔ "places_aps" (used to store the detected APs on each place)

Field	Type	Comments
place_ID	int(8)	Place index
MAC	varchar(12)	AP physical address
SSID	varchar(32)	Network name
RSSI	int(4)	Received signal strength (dBm)

Table 7 - places_aps structure

➔ ”places_cells” (used to store the detected cells on each place)

Field	Type	Comments
place_ID	int(8)	Place index
cellID	int(11)	Cell ID
RSSI	int(4)	Received signal strength (dBm)

Table 8 - places_cells structure

The **plans data group** has just one table (Table 9) where the image files and the information regarding to the building plans are stored:

➔ “plans” (used to store the building plans information and image files)

Field	Type	Comments
ID	int(11)	Plan index
Name	varchar(30)	Plan name
Description	varchar(150)	Plan description
minLatitude	int(9)	Bottom margin latitude
maxLatitude	int(9)	Top margin latitude
minLongitude	int(10)	Left margin longitude
maxLongitude	int(10)	Right margin longitude
Floor	int(4)	Plan floor
Plan	longblob	Plan image file
Uploader	varchar(15)	User who added the plan
Date	datetime	Date when the plan was added

Table 9 - plans structure

7. PROTOCOL BETWEEN SERVER AND MOBILE DEVICES

7.1. Transmission Control Protocol (TCP)

The Transmission Control Protocol [29] is one of the core protocols of the Internet Protocol Suite. TCP is one of the two original components of the suite, complementing the Internet Protocol (IP), and therefore the entire suite is commonly referred as TCP/IP. TCP is a reliable connection-oriented protocol that allows a byte stream originating on one machine to be delivered without error on any other machine over the Internet. It fragments the incoming byte stream into discrete messages and passes each one over the Internet layer. At the destination, the receiving TCP process reassembles the received messages into the output stream. TCP also handles flow control to make sure a fast sender cannot swamp a slow receiver with more messages than it can handle.

7.2. User Datagram Protocol (UDP)

The User Datagram Protocol [29] is an unreliable, connectionless protocol for applications that do not want TCP's sequencing or flow control and wish to provide their own. It is also widely used for one-shot, client-server-type, request-reply queries and applications in which prompt delivery is more important than accurate delivery, such as transmitting speech or video.

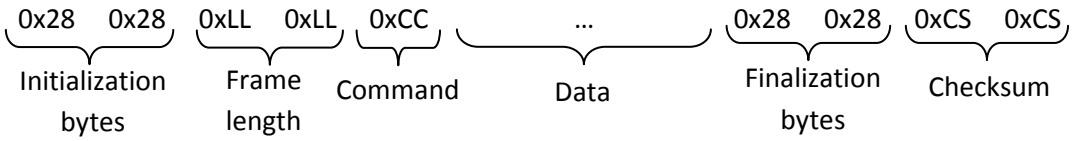
7.3. TCP vs. UDP

A first analysis of the communication between the server and the mobile devices for this project led to the choice of TCP. The application running in the smartphones needs a permanent connection to the server to make requests and get answers. It is a bi-directional connection and the reliability is extremely important. So, it was developed an application protocol over TCP and started testing the result. It was noticed that the time delay between a request and an answer was around a few seconds with the mobile device in the same network of the server. It should not be that long so some research was done to find out why it was happening, which resulted in the discovery of reports non-official forums claiming that the Android OS has some issues dealing with TCP. Shortly after, other references to the same problem [30] were found and it was decided to implement the protocol again over UDP. The communications over UDP turned out to be much faster. It was not made an exhaustive study about this, but the time delay between a request and an answer became around a few milliseconds.

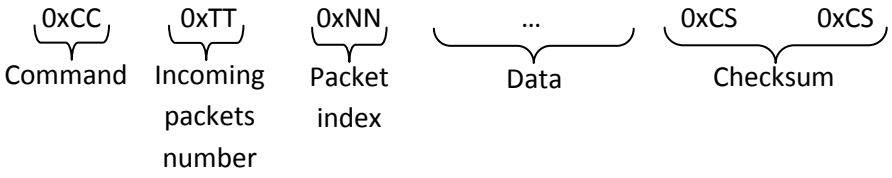
7.4. Application Protocol

The application protocol used in this project is based on two general frames, one for the requests and other for the answers:

Requests' frame (from smartphone to server):



Answers' frame (from server to smartphone):



These frames were developed from others initially created to operate over TCP but they were adapted to operate over UDP. The UDP is an unreliable protocol so it was added a checksum field to guarantee the data's integrity. It uses a CRC16 algorithm. This algorithm is an error detecting code designed to detect accidentally changes to raw computer data, and is commonly used in digital networks and storage devices such as hard disk drives.

The UDP implementation in Android OS limits the packet datagram to 64kB. In the requests' frame there is no problem because it is always shorter than that, but in the answers' frame it can cause some troubles since the application needs to transfer files (building plans). It was necessary to implement a data fragmentation process to divide the data in smaller arrays and add a field on the frame where it is indicated the total number of incoming packets. Another problem caused by the use of UDP is the possibility that the packets do not arrive in the same order they are sent, so it is essential to include a new field in the packet frame, which contains the packet index making it possible for the receiver to reorder the packets. That field is called "Packet Index", as depicted in the picture above.

For each function in the server there is a specific data field structure to encapsulate the information. Those structures are listed below:

➔ **wifiUpdate** – The following structure contains the necessary information to execute the function "Wifi Update" present in the server and explained on chapter 6.

Command: 0x01
Request data structure:

MAC : SSID : Latitude : Longitude : RSSI : Upgradable

MAC	MAC address of the detected Wi-Fi access point (without colons)
SSID	Public name of the Wi-Fi network
Latitude	Latitude where the AP has been detected in the format: degrees x 10^6 with a cast to integer
Longitude	Longitude where the AP has been detected in the format: degrees x 10^6 with a cast to integer
RSSI	Received signal strength in dBm
Upgradable	"0" if the coordinates are precise and cannot be changed; "1" if the coordinates are not precise and can be changed

Answer data structure: there is no answer from the server

➔ **wifiRequest** – This structure allows the mobile application to access to the "Wifi Request" function of the server.

Command: 0x02
Request data structure:

MAC

MAC	MAC address of the AP from which you want to know the location

Answer data structure:

Latitude : Longitude : 00000000000000

Latitude	Latitude stored on the database for the requested AP in the format: degrees x 10^6 with a cast to integer
Longitude	Longitude stored on the database for the requested AP in the format: degrees x 10^6 with a cast to integer

If the requested AP was not in the database the server's answer will be an 18 bytes sequence of zeros:

{ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }

➔ **cellUpdate** – Data structure to interact with the "Cell Update" function (explained on chapter 6).

Command: 0x03
Request data structure:

Cell ID : Latitude : Longitude : RSSI : LAC : Upgradable

Cell ID	Identification number of the detected mobile cell
Latitude	Latitude where the cell has been detected in the format: degrees x 10^6 with a cast to integer
Longitude	Longitude where the cell has been detected in the format: degrees x 10^6 with a cast to integer
RSSI	Received signal strength in dBm
LAC	Local area code
Upgradable	"0" If the coordinates are precise and cannot be changed; "1" if the coordinates are not precise and can be changed

Answer data structure: there is no answer from the server

➔ **cellRequest** – Data structure to execute the "Cell Request" function available on the server.

Command: 0x04
Request data structure:

Cell ID : 000000000

Cell ID	ID of the cell from which you want to know the location

Answer data structure:

Latitude : Longitude : 00000000000000

Latitude	Latitude stored on the database for the requested cell id in the format: degrees x 10^6 with a cast to integer
Longitude	Longitude stored on the database for the requested cell id in the format: degrees x 10^6 with a cast to integer

If the requested cell id was not in the server's answer will be an 18 bytes sequence of zeros:

{ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }

➔ **userUpdate** – Data structure with the user information to update the database through the "User Update" server function.

Command: 0x05
Request data structure:

Phone ID : Name : Status : lastLatitude : lastLongitude : lastFloor : lastUpdate

Phone ID	Identification number of the device
Name	Current user name
Status	Current user status
lastLatitude	Current user latitude in the format: degrees x 10^6 with a cast to integer
lastLongitude	Current user longitude in the format: degrees x 10^6 with a cast to integer
lastFloor	Current user floor
lastUpdate	Device date in milliseconds from January 1, 1970

Answer data structure:

Operation without errors:

Result : 00000

Result	String with the operation result (“New user added” or “User updated”)

Operation error:

The server returns an 18 bytes sequence of zeros:

{ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }

➔ **getFriendsLocations** – Structure with a device ID to execute the “Get Friends Location” function and get the friends’ information.

Command: 0x06

Request data structure:

Phone ID

Phone ID	Device identification from which you want to know the friends’ locations

Answer data structure:

Friend ID : Name : Status : lastLatitude : lastLongitude : lastFloor : lastUpdate ; … ; end

Friend ID	Friend phone id
Name	Current friend name
Status	Current friend status
lastLatitude	Current friend latitude in the format: degrees x 10^6 with a cast to integer
lastLongitude	Current friend longitude in the format: degrees x 10^6 with a cast to integer

lastFloor	Current friend floor
lastUpdate	Friend location date in milliseconds from January 1, 1970

If that phone id has no associated friends, the server's answer will be an 18 bytes sequence of zeros:

{ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }

➔ **getPlanImage** – Data structure to acquire the desired plan image file through the "Get Plan Image" function of the server.

Command: 0x07
Request data structure:

Plan ID : 000000000000000

Plan ID	Plan identification number

Answer data structure:

If there is an image available:

.png image file (it can be split in more than 1 packet)

If there is no image available the server's answer will be an 18 bytes sequence of zeros:

{ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }

➔ **searchForPlace** – Structure the execute the fingerprint localization algorithm on the server ("Search for Place" server function).

Command: 0x08
Request data structure:

MAC1 : RSSI1 : … : MACn : RSSIn

(Sequence with the MAC addresses and the RSSIs of the detected APs separated by colons)

Answer data structure:

Place match found:

Place ID : Name : Description : Latitude : Longitude : Floor : Uploader : Date

Place ID	Place identification number
Name	Place name
Description	Place description
Latitude	Place latitude in the format: degrees x 10^6 with a cast to integer
Longitude	Place longitude in the format: degrees x 10^6 with a cast to integer
Floor	Place floor
Uploader	User who added the place
Date	Date when the place was added, in milliseconds from January 1, 1970

Place match not found:

The server's answer will be an 18 bytes sequence of zeros:

{ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }

➔ **searchForPlanInfo** – The following structure is used to interact with "Search for Plan Info" function of the server (chapter 6).

Command: 0x09

Request data structure:

Latitude : Longitude : Floor

Latitude	Plan latitude in the format: degrees x 10^6 with a cast to integer

Longitude	Plan longitude in the format: degrees x 10^6 with a cast to integer
Floor	Plan floor

Answer data structure:

Plan(s) found:

Plan ID : Name : Description : minLatitude : maxLatitude : minLongitude : maxLongitude : Floor : Uploader : Date ; … ; end

Plan ID	Plan identification number
Name	Plan name
Description	Plan description
minLatitude	Minimum latitude (bottom margin of the plan) in the format: degrees x 10^6 with a cast to integer
maxLatitude	Maximum latitude (top margin of the plan) in the format: degrees x 10^6 with a cast to integer
minLongitude	Minimum longitude (left margin of the plan) in the format: degrees x 10^6 with a cast to integer
maxLongitude	Maximum longitude (right margin of the plan) in the format: degrees x 10^6 with a cast to integer
Floor	Plan floor
Uploader	User who added the plan
Date	Date of the plan in milliseconds from January 1, 1970

If that phone id has no associated friends, the server will answer a sequence of 18 bytes with zeros:

{ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }

➔ **userAddFriend** – Structure with two device IDs that allows the "User Add Friend" function to add a friend relation between them.

Command: 0x0A
Request data structure:

Phone ID : Friend ID

Phone ID	User device identification number
Friend ID	Friend to be associated with the user

Answer data structure:
Friend successfully added:

Done:0000000000000

If an error occurs the server's answer will be an 18 bytes sequence of zeros:

{ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }

➔ **userDelFriend** – Structure to delete a friend relation through the "User Del Friend" server function (chapter 6).

Command: 0x0B
Request data structure:

Phone ID : Friend ID

Phone ID	User device identification number
Friend ID	Friend to be disassociated from the user

Answer data structure:
Friend successfully deleted:

Done:0000000000000

If an error occurs the server's answer will be an 18 bytes sequence of zeros:

{ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }

➔ **addPlace** – Data structure to add a place for fingerprint localization (the "Add Place" function is executed on the server side).

Command: 0x0C
Request data structure:

Name : Description : Latitude : Longitude : Floor : Uploader : Date :
APs number : AP MAC1 : AP SSID1 : AP RSSI1 : …
Cells number : Cell ID1 : Cell RSSI1 : …

Name	Place name
Description	Place description
Latitude	Place latitude in the format: degrees x 10^6 with a cast to integer
Longitude	Place longitude in the format: degrees x 10^6 with a cast to integer
Floor	Place floor
Uploader	User who added the place
Date	Date in milliseconds from January 1, 1970
APs number	Number of detected APs
AP MAC1	First detected AP MAC
AP SSID1	First detected AP SSID
AP RSSI1	First detected AP RSSI
Cells number	Number of detected Cells
Cell ID1	First detected Cell ID
Cell RSSI1	First detected Cell RSSI

Answer data structure:

Place successfully added:

Done:0000000000000

If an error occurs the server's answer will be an 18 bytes sequence of zeros:

{ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }

➔ **getServerStatus** – Data structure to get the server status.

Command: 0x0D

Request data structure:

An 18 bytes sequence of zeros:

{ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }

Answer data structure:

Total requests : Connections to database : wifiUpdate : wifiRequest : cellUpdate : cellRequest : userUpdate : getFriendsLocations : getPlanImage : searchForPlanInfo : userAddFriend : userDelFriend : getServerStatus : getServerInfo

Total requests	Number of requests to the server
Connections to database	Number of established connections to database
wifiUpdate	Number of Wi-Fi updates
wifiRequest	Number of Wi-Fi requests
cellUpdate	Number of cell updates
cellRequest	Number of cell requests
userUpdate	Number of user updates
getFriendsLocations	Number of friends locations requests

getPlanImage	Number of plan images requests
searchForPlanInfo	Number of searches for plans
userAddFriend	Number of friends relations added
userDelFriend	Number of friends relations deleted
getServerStatus	Number of server status requests
getServerInfo	Number of server info requests

➔ **getServerInfo** - Data structure to get the server info.

Command: 0x0E

Request data structure:

An 18 bytes sequence of zeros:

{ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }

Answer data structure:

Cells : Places : Plans : Positions : Devices : Users friends : APs

Cells	Number of geo-referenced cells in the database
Places	Number of places in the database (for fingerprint localization)
Plans	Number of plans in the database
Positions	Number of saved positions
Devices	Number of registered devices on the system
Users friends	Number of friends relations in the database
APs	Number of geo-referenced Wi-Fi access points in the database

8. HARDWARE

This project has a hardware component, which consists in a Bluetooth gateway to smartphones. Bluetooth is the best way available to interact with this type of devices, because it is universal between smartphones models and it is present in all Android devices. The only disadvantage is the power consumption and that cannot be disregarded.

This circuit allows Bluetooth devices to communicate with other devices over an UART port. In this case it is used to receive data from a SimpliciTI module.

SimpliciTI is a simple low-power RF network protocol aimed at small RF networks. Such networks typically contain battery operated devices which require long battery life, low data rate and low duty cycle and have a limited number of nodes talking directly to each other or through an access point or range extenders[8].

8.1. Bluetooth to SimpliciTI Gateway

This circuit is based on a Bluetooth module, the RN-42 from Roving Networks, which operates in a transparent mode, the read data from Bluetooth side is written in the UART and the same happens in the opposite direction. A battery or an external power supply between 3.5 V and 16V can power it. The average operating current consumption (measured with an external power supply) is approximately 45mA (31mA to the Bluetooth module and 14mA to the SimpliciTI module). See Figure 32:

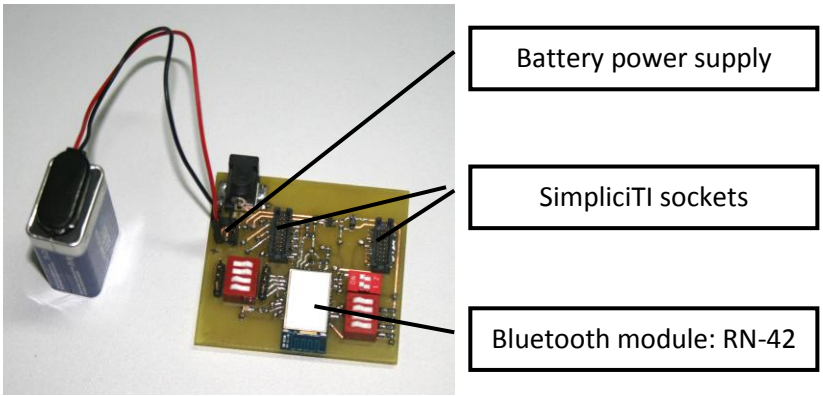


Figure 32 - Bluetooh to SimpliciTI gateway

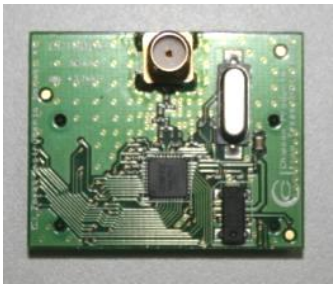


Figure 31 - CC1110EM module

The schematic of this circuit is in the Appendix D as well as the PCB layout (appendix E).

The SimpliciTI module is the CC1110EM from Texas Instruments (Figure 31) and operates at 433MHz. It is connected to the circuit using the two sockets available in the board and the gateway circuit gives the power supply to this module.

8.2. SimpliciTI modules

The SimpliciTI modules used in this work can be programmed using C language and already has an API developed by Texas Instruments[31]. This API has a wide range of functions to interact with the hardware. There are functions to initialize the modules (board, radio and stack), to establish links between them, to control the input/output ports, etc. The firmware development was made on "IAR Embedded Workbench" and downloaded to the modules through the SmartRF04 programming board (Figure 33).

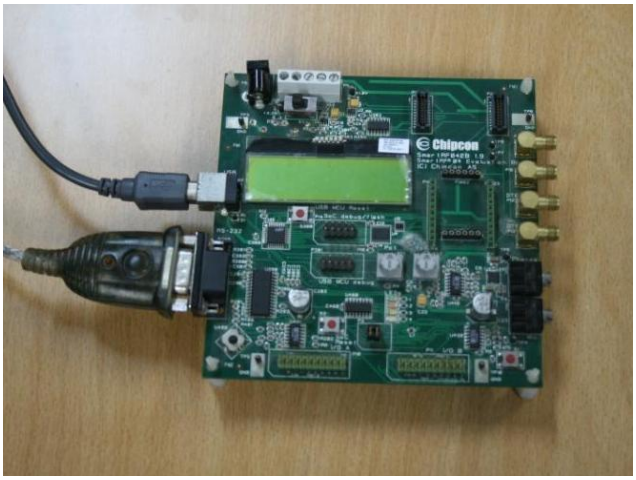


Figure 33 - SmartRF04 programming module

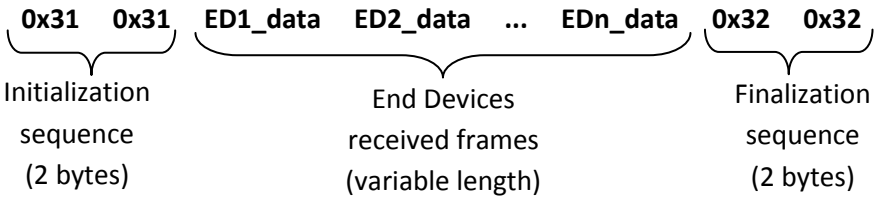
The SmartRF04 has a USB interface to communicate with it and a RS-232 connector to communicate with the attached module. It also has a LCD but it was not used in the project.

With these modules and making use of the Bluetooth to SimpliciTI gateway it was developed a localization system to smartphones. It is possible to locate people who are within 1 km (open field). If they are outdoor it is possible to use the GPS info, otherwise the distance between the access point and the end device can be estimated taking into account the RSSI.

8.2.1. Access Point module

The access point module is in charge of receiving localization requests through Bluetooth, inquiring the surrounding end devices (8.2.2) for their locations and returning the results through the Bluetooth channel. When this module is switched on both parts start operating (Bluetooth and SimpliciTI) and follow the diagram of Figure 34. It is still a prototype (Figure 35), so its dimensions are slightly higher than a final product.

This module uses a simple protocol to communicate through the Bluetooth channel. The localization requests can be made by sending the character '1' (0x31 in hex) to the module and it will answer with the following frame:



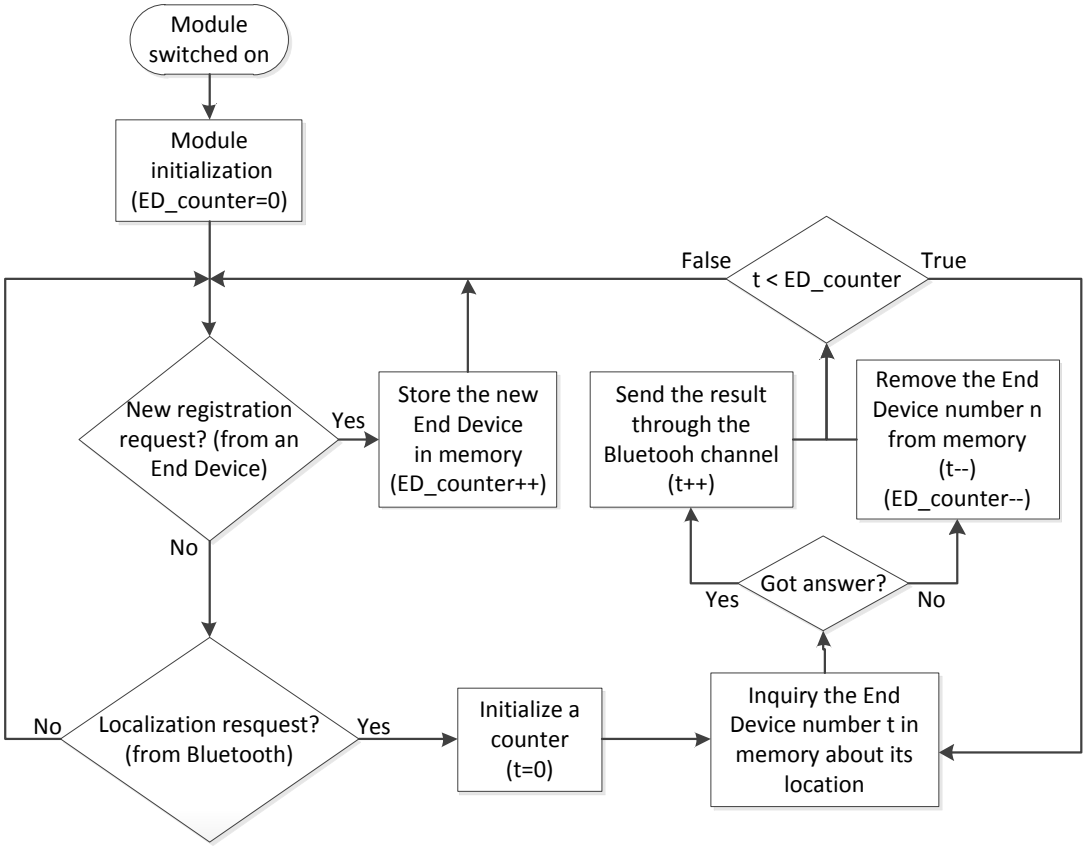


Figure 34 - Access Point operation diagram

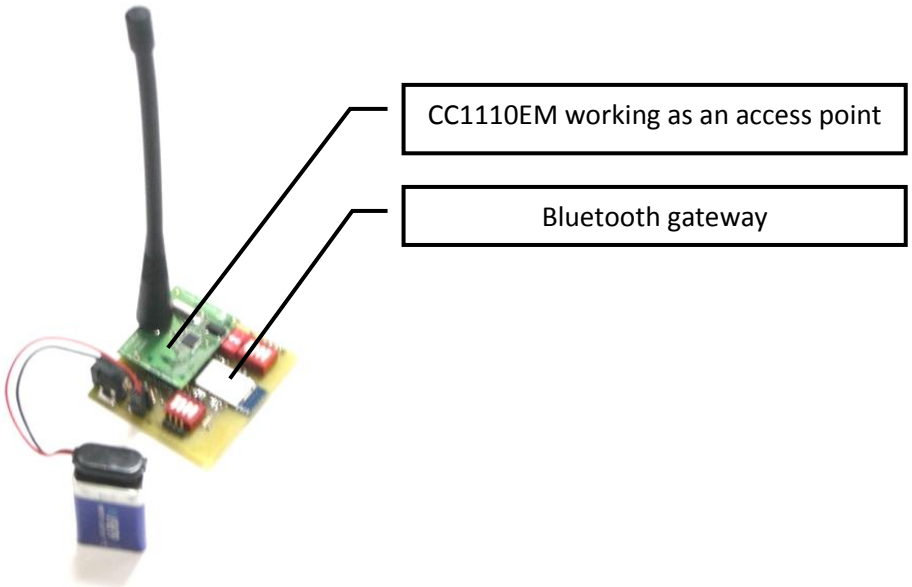


Figure 35 - Bluetooth to SimpliciTI gateway

8.2.2. End device (with GPS)

The end device is a CC1110EM from Texas Instruments with a GPS module attached (Figure 36). When it is switched on, it continuously tries to connect to an access point. Once connected, it waits for requests from the access point and answer them with the GPS coordinates. If the link fails it tries to reconnect to an access point, see Figure 37.

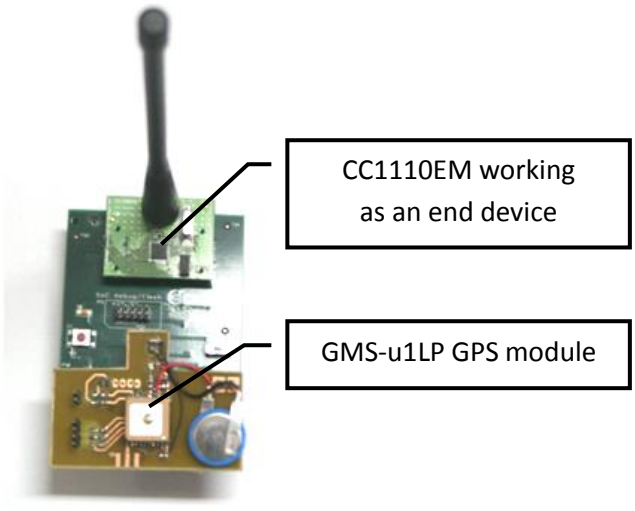


Figure 36 - SimpliciTI end device (with GPS)

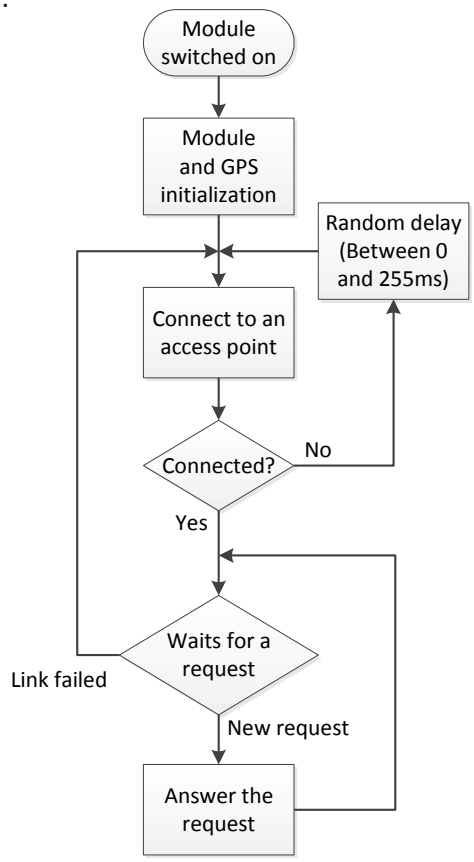


Figure 37 - End device operation diagram

The requests to the end devices should be made by sending the character '1' through the RF channel (0x31 in hex). They will answer a frame with the following structure:

0x11 0x11	**ed_address**	**link_rssi**	**coordidates_length**	**coordinates_data**	**0x22 0x22**
Intialization Sequence (2 bytes)	End Device address (4 bytes)	Link RSSI (1 byte)	GPS data length (1 byte)	GPS data (usually 22 bytes)	Finalization Sequence (2 bytes)

The coordinates data field has the latitude and the longitude separated by the character ','. In the absence of GPS data zeros will be sent instead.

9. LOCATE ME WSN

Locate Me WSN is an upgrade to the standard Locate Me application. This version has an additional feature to interact with the modules presented in chapter 8.2. It uses the Bluetooth to make localization requests to a configured access point module and presents the result on the application map viewer.

The access point module for which the application connects to can be directly set in the application configuration file with a text editor. It is in the installation directory and it is called "configuration.conf". The parameter to be set is the MAC address.

The result on the map viewer is a circle with a proportional radius to the distance between the access point and the end device and it is centered in the smartphone location. This distance is estimated taking into account the RSSI of the end device. The conversion from dBm to meters is made using the Friis formula with some changes:

Original Friis formula:

$$\frac{P_r}{P_t} = G_t G_r \left(\frac{\lambda}{4\pi R}\right)^2 \quad (9\text{-}1)$$

Pr – Received power
Pt – Transmitted power
Gt – Gain of the transmit antenna
Gr – Gain of the receive antenna
λ – Signal wave length
R – Distance between the transmitter and the receiver

Adapted Friis formula (with P_r in milliwatt):

$$P_r = K_1 \left(\frac{\lambda}{4\pi(R-K_3)}\right)^{K_2} \quad (9\text{-}2)$$

Adapted Friis formula (with P_r in dBm):

$$P_r = 10\ log_{10}\left(\frac{K_1\left(\frac{\lambda}{4\pi(R-K_3)}\right)^{K_2}}{0.001}\right) \quad (9\text{-}3)$$

Substituting the λ by its value:

$$P_r = 10\ log_{10}\left(\frac{K_1\left(\frac{0.6928}{4\pi(R-K_3)}\right)^{K_2}}{0.001}\right) \quad (9\text{-}4)$$

In order to use the (9-4) formula it is necessary to determine K_1, K_2 and K_3 constants, so it was measured and registered the received power (RSSI) for different distances (the results are in

appendix F). After that and using the *cftool* (a *Matlab* curve fitting tool) it was possible to estimate those constants:

Coefficients (with 95% confidence bounds):
K_1 = 0.04642 (-0.1607, 0.2536)
K_2 = 2.836 (2.223, 3.449)
K_3 = 11.71 (6.801, 16.61)

Goodness of fit:
SSE: 1.714e+004
R^2: 0.641
Adjusted R^2: 0.6386
RMSE: 7.585
Number of samples: 317

The goodness of fit is not good as the R^2 is too much low, but this is a consequence of measuring distances based on received powers. The path loss suffers big variations due to external interferences like material barriers and multipath effect.

It is now possible to rewrite the (9-4) and get the distance in terms of received power:

$$R = \frac{0.6928}{K_2\sqrt{\frac{0.001\times 10^{\frac{P_r}{10}}}{K_1}}} + K_3 \qquad (9\text{-}5)$$

K_1 = 0.04642
K_2 = 2.836
K_3 = 11.71
P_r = received power (dBm)
R = estimated distance (meters)

The Locate Me WSN uses this formula (9-5) to estimate the distance between the access point module and the end devices. In addition to the circle, if there are valid GPS coordinates in the end device, a point will be drawn on that place. The figure bellow shows an example of that (Figure 38):

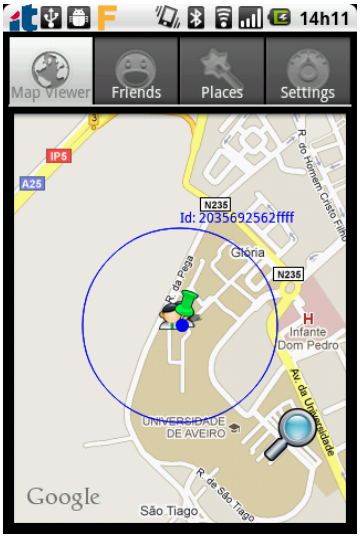


Figure 38 - End Device localization example

The "Id" written on the top of the circle represents the identification of the End Device. It can be inserted on the web site[28] where the historic positions can be seen.

10. CONCLUSIONS AND FUTURE WORK

The result of this project is an integrated system for indoor and outdoor localization. It uses not only GPS to acquire the location but it is able to use Wi-Fi, mobile base stations and wireless sensor networks. The diversity of alternative methods for localization makes this system robust, flexible and functional in almost all environments. The high number of Wi-Fi access points in urban areas increases the probability of get a good localization result. Figure 39 shows the geo-referenced Wi-Fi access points in the system database and the Figure 40 their evolution. They show that Wi-Fi networks are present in large numbers in almost all developed countries.

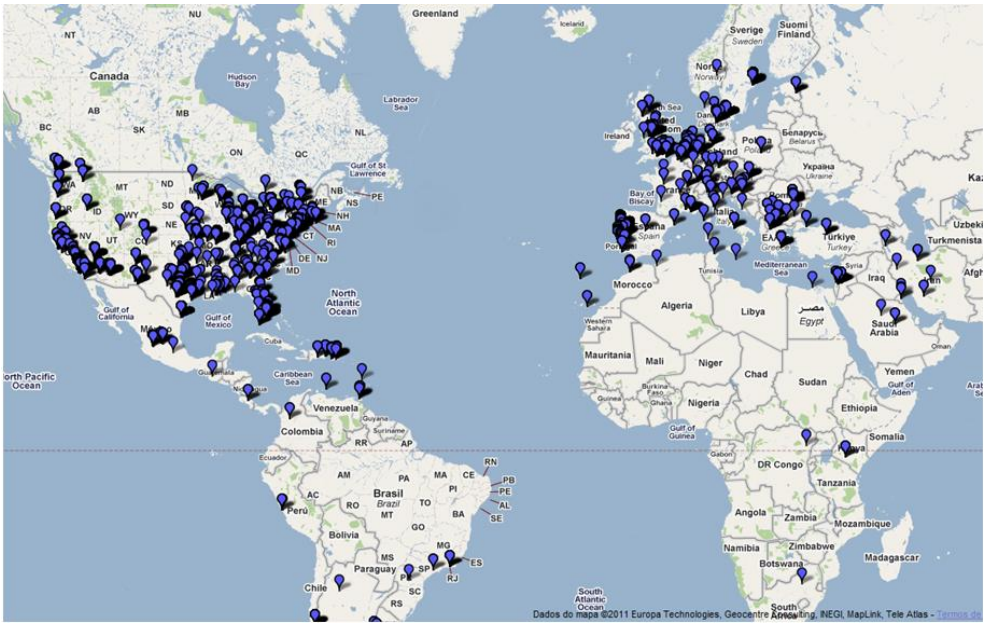


Figure 39 - Wi-Fi access points in the database (at the 1st of June of 2011)

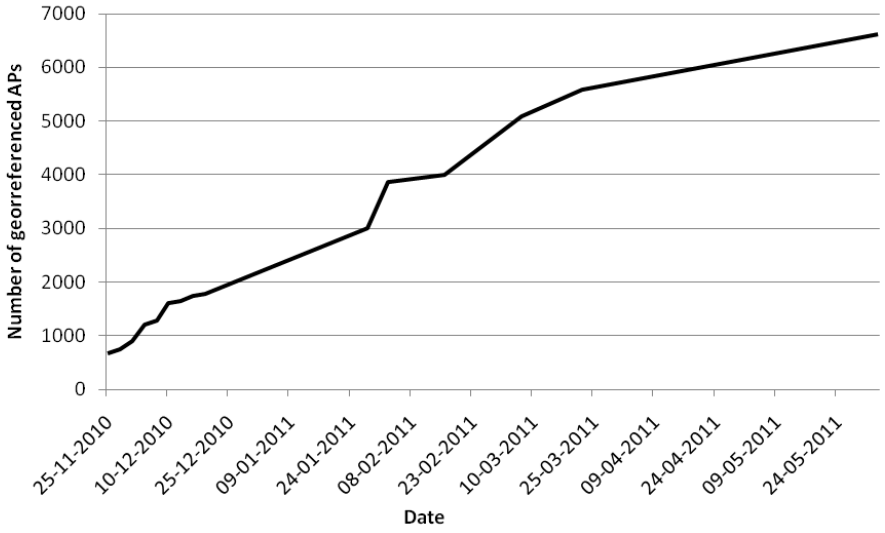


Figure 40 - Wi-Fi access points in the database evolution

In terms of geo-referenced mobile cells the results were also good. Figure 41 shows the cells on the map (they seem to be less than in Figure 42 graph but that is because of some icons' overlap):

Figure 41 - Mobile cells in the database (at the 1st of June of 2011)

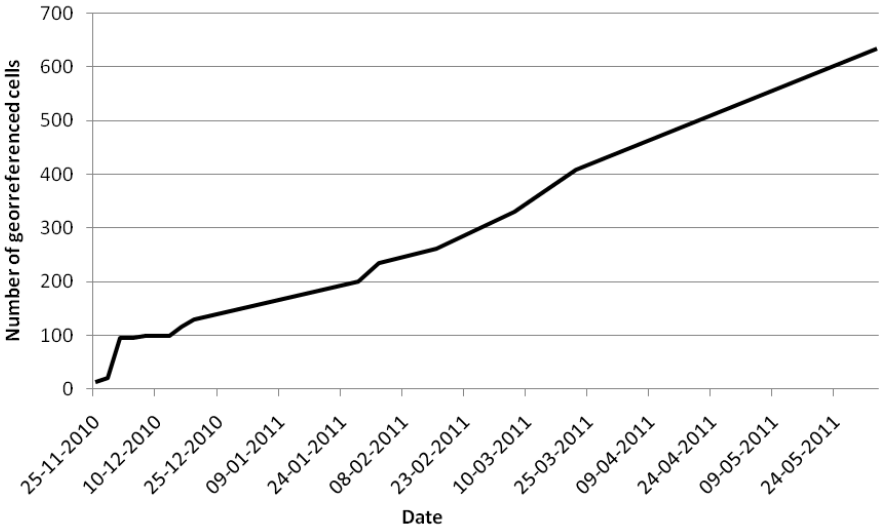


Figure 42 - Mobile cells in the database evolution

The number of users in the system was another surprise, because without any advertisements and just publishing the application on the Android Market it was possible to achieve more than one thousand and five hundred users all around the World. See Figure 43:

Figure 43 - Locate Me users all around the World

The hardware component of this project is a helpful accessory to use together with “Locate Me”. There are several practical applications in which this system can be used. For example when a mother goes out with her children it is possible to know if they are close to her and if they moved away that can be seen on the smartphone. In addition to that, the mobile modules (the end device explained on 8.2.2) have an integrated GPS antenna, so if this situation takes place outdoor it will be possible to know their exact location in real time.

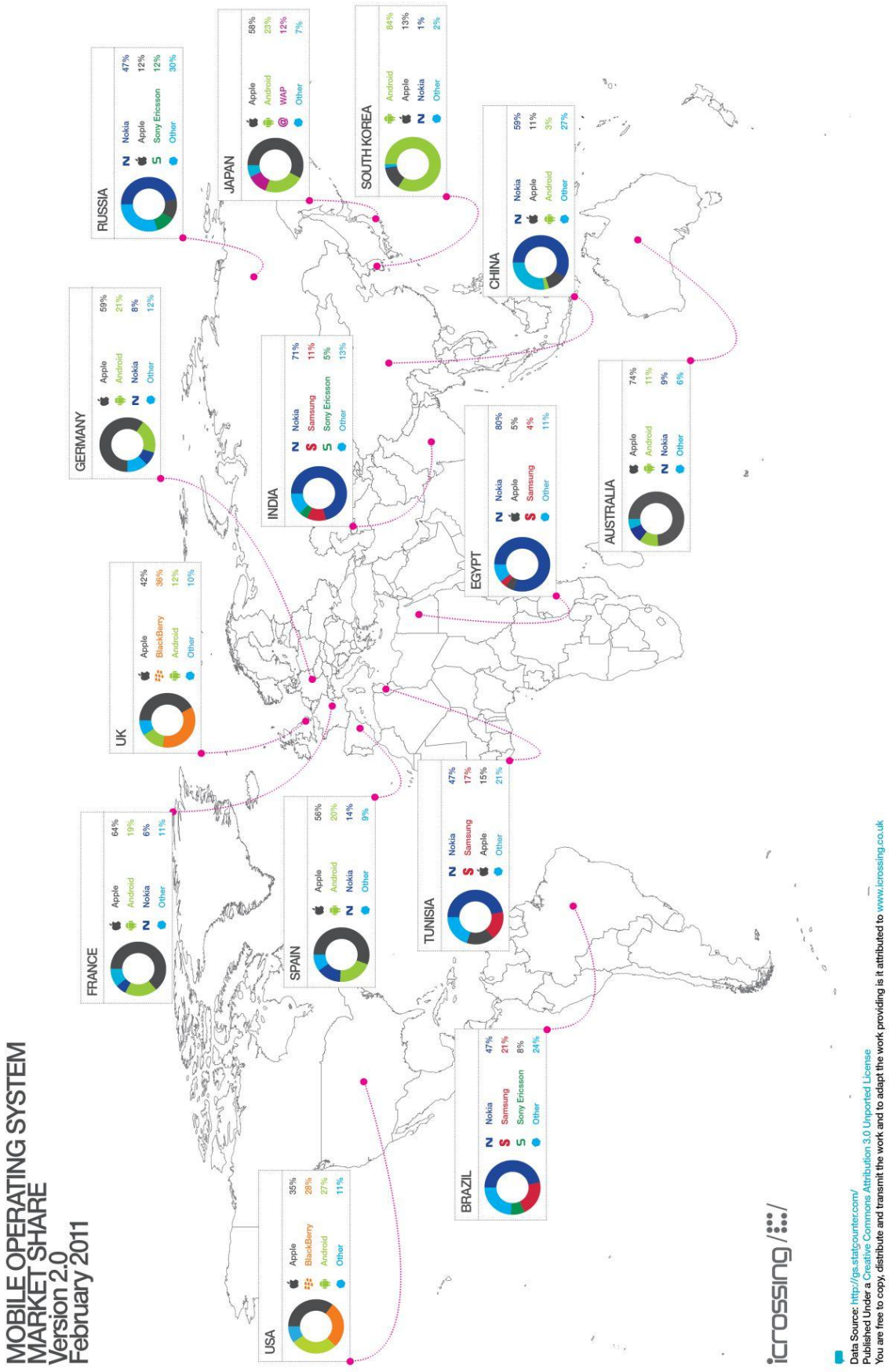
With these results it is possible to conclude that all the objectives for this project have been achieved and successfully tested. “Locate Me” has been tested by “random” people since it was published on the Android Market and can be downloaded for free. The hardware component was just tested in a controlled environment, because there is only one module available of each (one end device and one access point module).

For future work some improvements regarding the features of “Locate Me” can be made (e.g. alarms when the distance between modules is bigger than a configured value). The power consumption is a critical issue in these mobile devices and some upgrades can be made, either software or hardware.

The communication protocol between the smartphone and the server is something that can be also optimized in order to reduce the data traffic and prevent communication errors.

APPENDIX

A. Mobile Operating Systems Share[32]

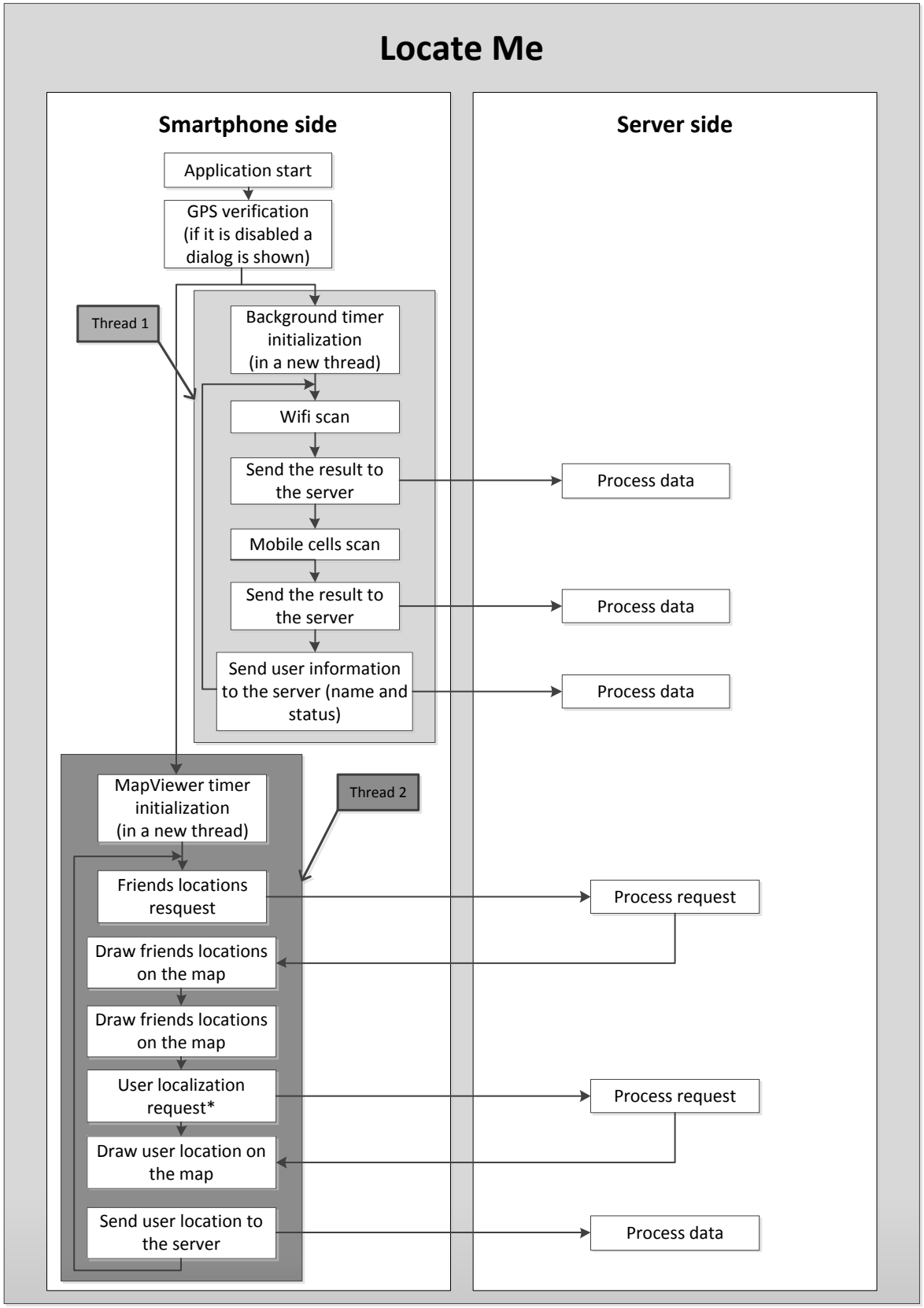


B. Windows Phone 7 vs. iOS vs. Android OS[33]

PCWorld

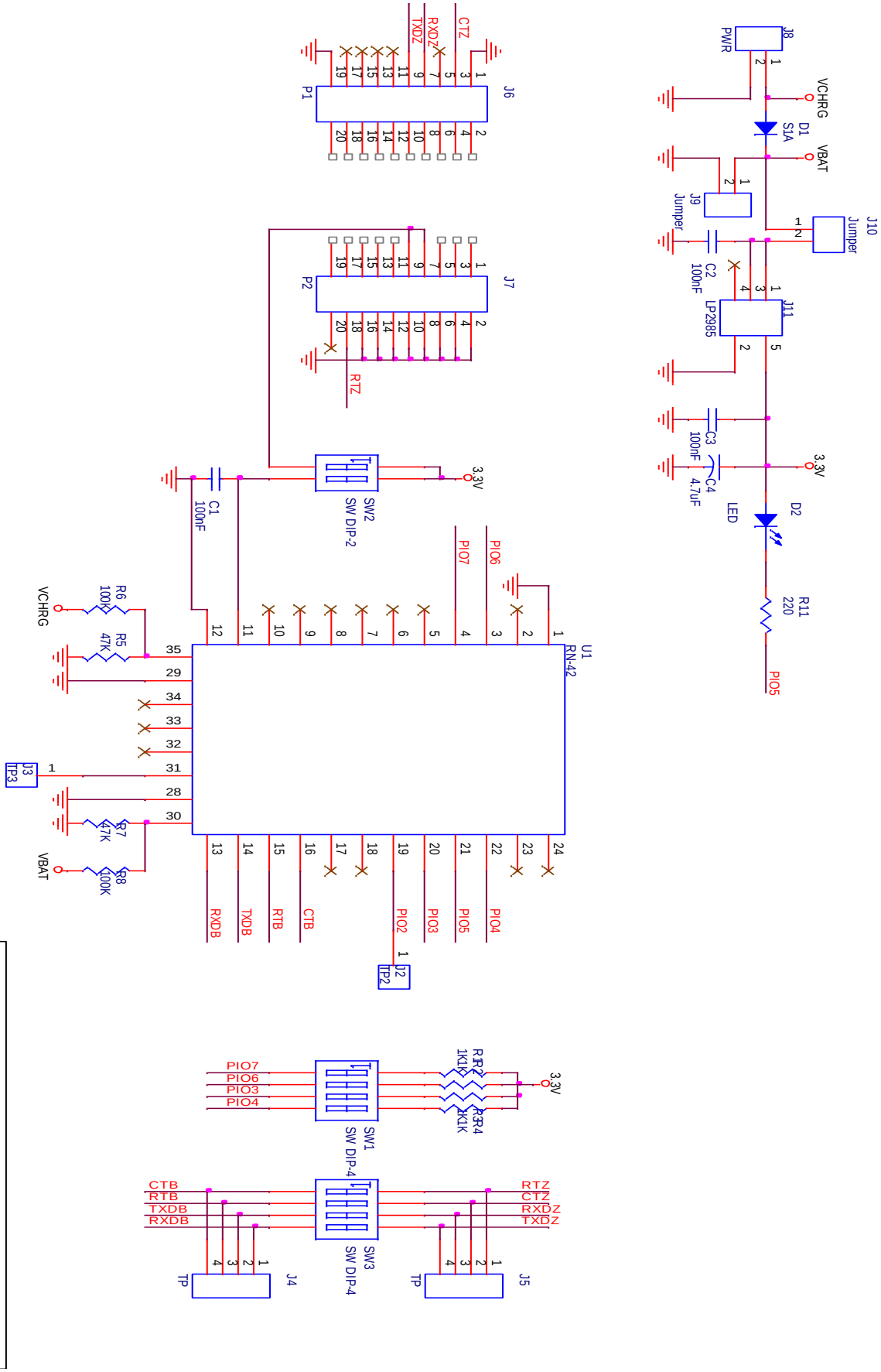
	Windows Phone 7	iOS (iPhone)	Android
Developer	Microsoft	Apple	Google
Copy/Paste	✘	✔	✔
Multitasking	✘	✔	✔
Flash Support	✘	✘	✔
Silverlight Support	✘	✘	✘
HTML5 Support	✘	✔	✔
Unified Inbox	✘	✔	✔
Exchange Support	✔	✔	✔
Threaded Email	✘	✔	✔
Visual Voicemail	✘	✔	✔
Video Calling	✘	✔	✔ Third Party App
Universal Search	✘	✔	✔
Internet Tethering	✘	✔	✔
Removable Storage	✘	✘	✔
Facebook Integration	✔	✘ (Third Party App)	✔ (Third Party Integration)
Twitter Integration	✘	✘ (Third Party App)	✔ (Third Party Integration)
Folders	Hubs	✔	✔
Apps Organization	Alphabetical	Customizable	Customizable
App Store	1,000+ Apps	300,000+ Apps	90,000+ Apps
Microsoft Office Support	Built-In	Third Party App	Third Party App
Widgets	Tiles on Home Screen	✘	✔
Media Sync	Zune Software Mac & PC	iTunes Mac & PC	Direct File Transfer + Third Party Software
X-Box Live Integration	Built-In	Via Third Party App	Via Third Party App

C. Locate Me operation diagram



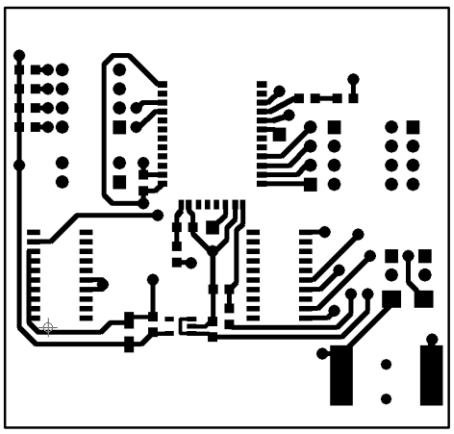
* Follows the sequence of Figure 13. It makes several requests to the server.

D. Bluetooth to SimpliciTI gateway schematic

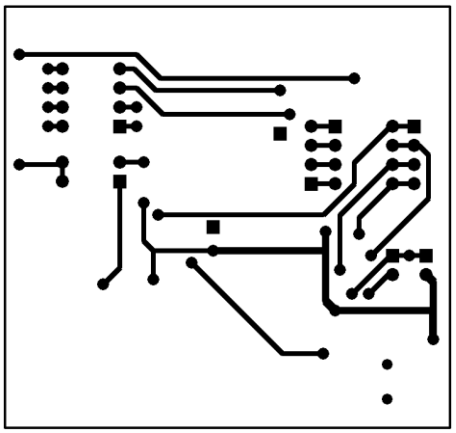


Title	Bluetooth to SimpliciTI gateway
Size	Document Number
Date:	Friday, May 27, 2011
Sheet	1 of 1
Rev	v1.0

E. Bluetooth to SimpliciTI gateway layout

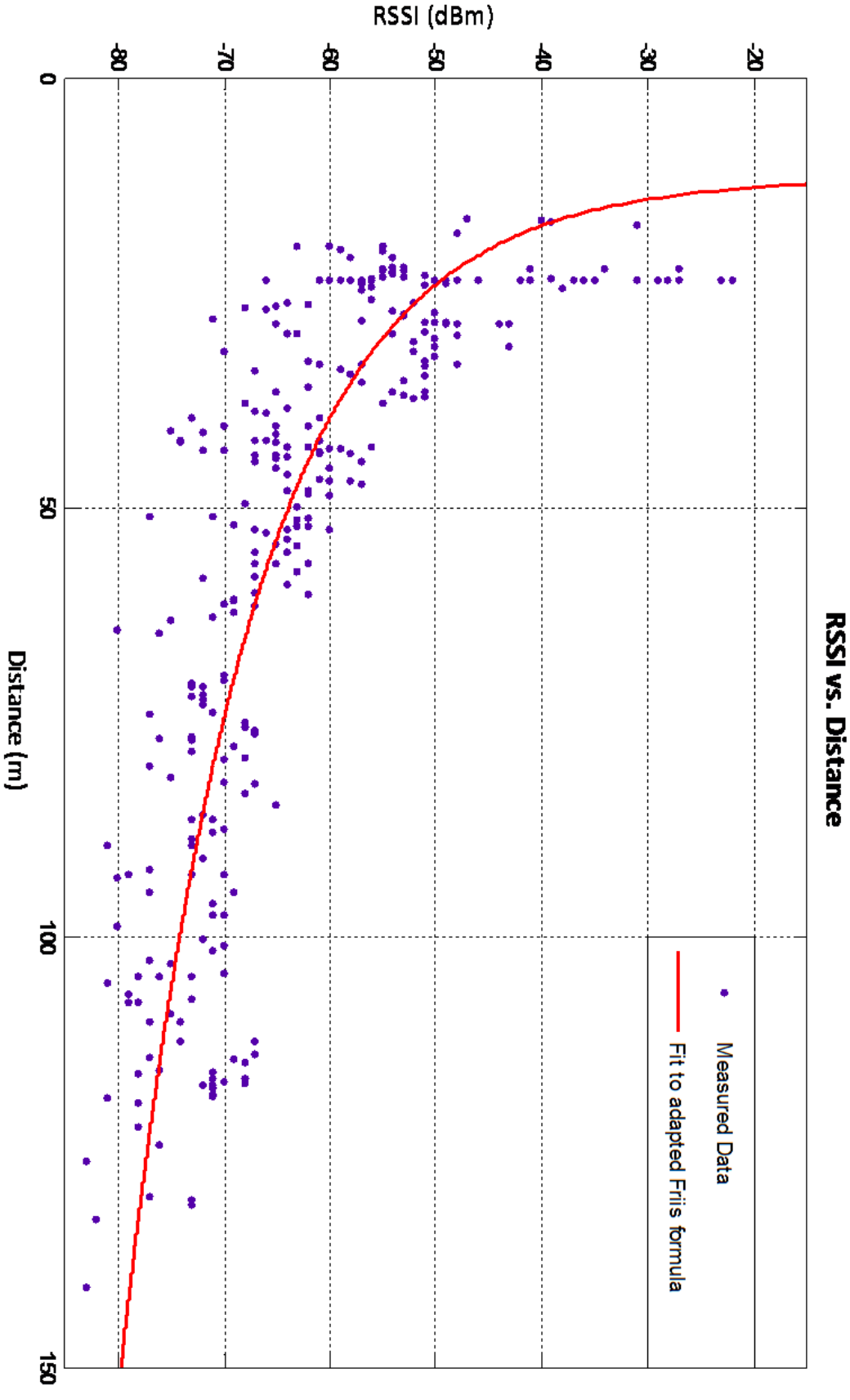


(Layout top view)



(Layout bottom view)

F. Measured RSSI in terms of the distance between modules



BIBLIOGRAPHY

[1] (2009). *HardwareZone*. Available: http://2009.hardwarezone.com/articles/print.php?cid=14&id=2654

[2] L. Greenemeier. (2009). *Apple's Mac turns 25*. Available: http://www.scientificamerican.com/blog/post.cfm?id=apples-mac-turns-25-2009-01-23

[3] N. Baiao. (2008). *Senha de proteção do iPhone pode ser driblada facilmente*. Available: http://snnangola.wordpress.com/2008/08/29/senha-de-protecao-do-iphone-pode-ser-driblada-facilmente/

[4] A. El-Rabbany, *Introduction to GPS: the Global Positioning System*: ARTECH HOUSE, INC., 2002.

[5] Wi-Fi. (2011). *Wi-Fi Alliance*. Available: http://www.wi-fi.org/

[6] Bluetooth. (2011). *About the Technology*. Available: http://www.bluetooth.com/Pages/About-the-Technology.aspx

[7] (2011). *GeekProject*. Available: http://geekproject.com.br/2011/04/a-evolucao-dos-celulares/

[8] T. Instruments. (2011). *SimpliciTI Compliant Protocol Stack*. Available: http://focus.ti.com/docs/toolsw/folders/print/simpliciti.html

[9] H. A. Asoke K. Talukdar, Roopa R. Yavagal, *Mobile Computing 2E*: Tata MacGraw Hill, 2010.

[10] Apple. (2011). *iPod*. Available: http://www.apple.com/pt/ipod/

[11] J. P. Cunha, "Apontamentos de Computação Móvel: Mobility History," ed, 2010.

[12] N. A. o. P. A. National Research Council (U.S.). Committee on the Future of the Global Positioning System, *The global positioning system: a shared national asset : recommendations for technical improvements and enhancements*: National Academies Press, 1995.

[13] A. P. Subramanian*, et al.*, "Drive-By Localization of Roadside WiFi Networks," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, 2008, pp. 718-725.

[14] M. Gauger, "Integration of Wireless Sensor Networks in Pervasive Computing Scenarios," Berlin, 2010.
[15] E. Chan*, et al.*, "Using Wi-Fi Signal Strength to Localize in Wireless Sensor Networks," in *Communications and Mobile Computing, 2009. CMC '09. WRI International Conference on*, 2009, pp. 538-542.
[16] Google. (2011). *Latitude*. Available: https://www.google.pt/latitude
[17] O. H. Alliance. (2007). *FAQ*. Available: http://www.openhandsetalliance.com/oha_faq.html
[18] J. V. Camp. (2011). *Honeycomb screenshot*. Available: http://www.digitaltrends.com/mobile/android-3-0-honeycomb-new-feature-breakdown-with-screenshots/
[19] Microsoft, "Windows Phone 7: A New Kind of Phone (36:47 min)," ed, 2010.
[20] Microsoft. (2010). *Windows Phone 7: A Fresh Start for the Smartphone*. Available: http://www.microsoft.com/presspass/features/2010/oct10/10-11wp7main.mspx
[21] Microsoft. (2011). *APP HUB*. Available: http://create.msdn.com/en-us/home/membership
[22] Apple. (2011). *iOS 4*. Available: http://www.apple.com/iphone/ios4/
[23] Blackberry. (2011). *Blackberry devices*. Available: http://worldwide.blackberry.com/pt/
[24] (2010). *Symbian OS*. Available: http://pt.wikipedia.org/wiki/Symbian_OS
[25] Nokia. (2011). *All models*. Available: http://www.nokia.pt/produtos/todos-os-modelos
[26] bada. (2010). *What is bada?* Available: http://www.bada.com/whatisbada/index.html
[27] Samsung. (2011). *bada devices*. Available: http://www.bada.com
[28] L. Guenda. (2009). *IPIS - Indoor Positioning Identification System*. Available: http://ipis.av.it.pt
[29] A. S. Tanenbaum, *Computer Networks*, Fourth ed. Amsterdam, Netherlands: Pearson Education International, 2003.

[30] R. M. A. F. Palha, "Interacção entre um dispositivo móvel e um ecrã de grandes dimensões," Master, Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro, Aveiro, 2010.
[31] T. Instruments, *SimpliciTI - Application Programming Interface*. San Diego, California, USA, 2009.
[32] (2011). *Mobile Operating System Share*. Available: http://connect.icrossing.co.uk/mobile-market-share_6301#
[33] D. Ionescu, "Mobile OS Smackdown: Windows Phone 7 vs. iOS vs. Android," *PC World,* 2010.