

**Indoor Positioning System**

**LO53 –Spring 2012**

**Aurélien SIGNE – Thibaut ACKERMANN**

User guide 3

Access Point (AP) 3

Server 4

Mobiles 4

List of the key tasks required in the project description 5

Access Point (AP) 5

Server 6

Mobiles 7

Self-analysis of your work 8

# User guide

The aim of this part is to explain how to use our programs. Firstly there are three different device types and each of them requires to be setup with special applications installed.

## Access Point (AP)

On the Access Point, there is special software to install and run, through SSH. This software is actually a kind of “sniffer” which scans and stores received Rssi values. In addition, the program is able to answer to the Tomcat Server requests. For example, if the server asks for an average Rssi value for a Mobile, the AP is able to respond.

*How to do it:*

First you have to copy the executable file on the AP. (our program is called “pcap”)

|  |
| --- |
| $ scp pcap [root@192.168.1.1](mailto:root@192.168.1.1):. |

Once it’s done, you still have to launch the program, through SSH

|  |
| --- |
| $ ssh [root@192.168.1.1](mailto:root@192.168.1.1)  # ./pcap |

You should start to see information in the console. When a new device is detected, you will see its MAC address and when an UDP request is proceed, you should see it. Finally, you can quit at anytime through the “Ctrl+C” shortcut. Of course you can use the command “nohup” to let the program run even if you exit the ssh session.

## Server

On the server, a Tomcat Application Server coupled with a PostgreSQL database has to be installed. With the start of this Tomcat Server, 2 servlets are launched waiting the request of mobiles, and a listener is launched to communicate with all APs.The first servlet is used to manage mobile devices calibration requests and the second to manage mobile devices positioning requests.

Each access point has to be added manually in the table “AccessPoint” of the database.

The first servlet charged to manage mobile devices calibration requests store information in the tables “Location” and “Rssi”.

The second charged to manage mobile devices positioning requests, stores temporary information in the table “TempRssi”. It uses the tables “Rssi” and “Location” to compute the mobile position.

## Mobiles

In the project description, 2 applications were needed for the mobiles. We decided to develop the features in only one application called “LO53 – IPS” for Indoor Positioning System.

This application has to be installed on each Android mobile. So each mobile may generate "burst" in the offline mode or asks to the server its position in the online mode.

# List of the key tasks required in the project description

We had to develop special features for each device type, here is a description of what is done and what is partially done.

## Access Point (AP)

The software is actually divided into 3 different threads. Each of them is launched at the same time, and none of them dies until the end of the program (You can properly stop the software at any time with the “Ctrl+C” shortcut).

*Thread 1: the “sniffer”*

This thread run indefinitely and when it receives a new Rssi values (through pcap and the “prism0” interface) it stores this value into a Double Linked List structure.

Figure : The Double Linked List System

*This part is totally implemented.*

*Thread 2: the cleaner*

This thread flushes the old values in the Linked List. As we need to compute an average value of the Rssi for each device, and as this average is based on the values received during the last second, we need to put a timestamp on each Rssi and flush the old values (older than 1 second). This thread goes through all the Lists and removes the Rssi if it’s required.

*This part is totally implemented.*

*Thread 3: The UDP Server*

This thread creates an UDP server and answer to all the requests received. It handles both the Offline and Online modes. We decided to modify a bit the UDP Frames.

Here is the Offline UDP Frame, sent by the Tomcat server.

|  |
| --- |
| GETOFF; posx; posy; map\_id; android\_mac\_addr |

The AP responds with:

|  |
| --- |
| RSSIO; posx; posy; map\_id; android\_mac\_addr; ap\_mac\_addr; avg\_value |

And for the online mode:

|  |
| --- |
| GET; android\_mac\_addr |

The AP responds:

|  |
| --- |
| RSS; android\_mac\_addr; ap\_mac\_addr; avg\_value |

*This part is totally implemented.*

## Server

We had to develop the 2 servlets and the listener. Below, you will find the detail of each feature developed and the status of their development.

The first servlet waiting the mobile devices calibration requests is called ***MobileCalibrationListener.*** When a mobile access to the server with this url :

http://ipServer:8080/ServerPositionning/MobileCalibrationListener?x=value\_x&y= value\_y&map=value\_map

a thread is launched to treat the request, in others words to ask the RSSI average of each AP and store information in the database.

*This part is totally implemented.*

The second servlet waiting the mobile devices position requests is called ***MobileLocationListener.***When a mobile access to the server with this url :

http://ipServer:8080/ServerPositionning/MobileLocationListener

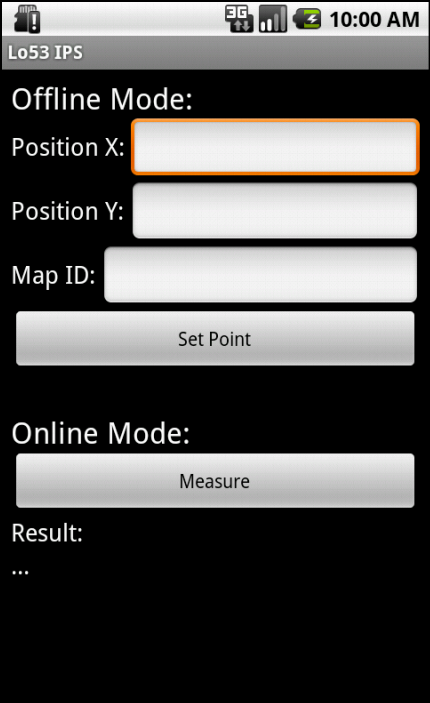
a thread is launched and asks to each AP the RSSI received for this mobile address, computes the mobile position and returns this position to the mobile.

*This part is partially implemented. We didn’t have the time to develop the complex algorithm to compute the mobile position but we developed a simplified algorithm, which find in the database the nearest position.*

The listener called ***APListener*** creates a thread, whichis charged to listen Aps responses on UDP port. Indeed each time a mobile send a request to the server, the server send a package to all APs to know the RSSI of the mobile. The server obtains his response thanks to this listener.

*This part is totally implemented.*

## Mobiles

 Two applications were needed for the mobiles. We decided to develop only one application. The main screen of this application is divided in two parts.

On the top, there is the calibration part and on the bottom we have the possibility to ask to the server our position.

The delays being relatively short, we did not have the time to develop all features such as the display of the map…

*This part is partially implemented.*

# Self-analysis of your work

This project was really interesting. During the development, we used our knowledge in different programming languages. We discovered Tomcat Application Server coupled with a PostgreSQL database. We also used different connection protocols (HTTP, UDP…) between different devices (Access Points, Server & Mobiles).

Concerning our development, we have always split the work into 2 parts, between both of us. For example Aurélien handled the Tomcat Server part, while Thibaut worked on the Android Application and we both worked on the AP software.

Even if all features have not totally implemented and if the project needs some improvements, we have a global project, which works.