Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Final Master Thesis

Development of tools for the use of Android cell-phones to recognize user activities

Course: ***Master of Siense in Information communication and Technology***

Student: **Nooshin Abdollah**

Director: **Juan Luis Gorricho**

Year: December 2015

Contents

Table of Pictures

Session 1

Introduction

In this fast pace technology driven world, access to the Internet anywhere and anytime is playing a key role in human's life, Moreover, most of the businesses attempt to consider the value of internet and its mobility in their provided market.

The client architecture of Internet mobility devices includes of smartphones, tablets and laptops. Based on a research in 2012 in EU, using Internet by movable devices such as portable computer was 36% and by tablet 7% and the rest has belonged to the smartphone users. As is mentioned, popularity of access to Internet by smartphones and mobile phones has significant increase over recent years. Meanwhile, based on the "Global Smartphone Market Share 2012" research, 66.6% of smartphone clients use Android operating system devices. [10]

The goal of this study is developing the fundamental techniques to define activity of the users like walking, running, standing or cycling while the user is using one of two different types of Network connectivity, GPRS or Wi-Fi. This method is knows as Activity recognition that it is one the new researches and reachable by embedded sensors of the smart phone devices such as GPS sensors, vision sensors, direction sensors and acceleration sensors. Recently this subject of user activities is taken into the consideration of some mobile applications which are used for health care especially for daily lives of older people more than the alarm when the person falls because of the heart attack.

"Therefore by analyzing data is possible to define the action or state of the user with simple human activities such as walking, running, driving, bicycling and other classification methods." [6]

This study is not about using accelerometer sensor embedded in device, but calculating the distance of the user in a period of time by the longitude and latitude of the user´s location which has reached in the location recognizing. Therefore, the main goal of the study to retrieve user's activity is achieved by requesting location of user for each period of time, from current and previous location of user.

In order to accomplish the project some steps have to be done. The first step would be identifying if the device is connected to either Wi-Fi or GPRS connection. When the network is Wireless it means the connection is indoor and when the user is outdoor the type of connection would be GPS. The second step would be getting the valid GPS location of the user. When it is outdoor, is needed to track the user location along the time. Afterward, in the third step, the user activity is concluded which is walking, running, driving or standing. When the user is outdoor, it must save the sequences of GPS locations along the daytime, so it will discover the user´s daily pathway.

Session 2

Abstract: *In the session 2, I have explained Android programming language is used for creating and developing this study. So after an introduction about Android operating system, in the second part, I have sentenced the environment and its necessary tools for creating and developing an android project. In the third part of this session, I have described how to create an android project, then android project content components and entailed of the project.*

Android

Android is a comprehensive and open source operating system framework for intelligent smart phone devices which has been created by Andy Robin, the manager of Android Company, then has been bought by Google Company in 2005. The word of Android means "Humanoid Robot". In 2007, some proposed groups of producers in electronic equipment like cell phones and wireless equipment beside of some software companies including Google made a consortium with the goal of creation an Open Source Operating System of Cell phones in order to reduce expenses and costs of Mobile's Applications.

Therefore, this consortium with the name of Open Handset Alliance, has published Software Development Kit (Acronym SDK) consists of required software tools of developing Android Applications. The first cell-phone based on Android OS introduced in October of 2008 with the name of T-Mobile G1 by HTC Company. From that year onward, beside of cell-phones, these companies used the OS for other electronic tools such as Tablet, watch, electronic reader and even television.

Android software has utilized the power of JAVA for constructing its applications with the format of apk for a package of an application. The most famous development environments of Android are Open Source programing environments named as Android studio, NetBeans IDE and Eclipse beside of ADT and ADK.

This operating system has been optimized based on Virtual Machine Linux OS that is embedded as Dalvik for cell-phones devices. This virtual machine has several functions such as memory management, ease of use of Sandbox in application producing, more compression of applications and compatibility with variety of CPUs in different devices without the need if rewrite application for each specific CPU.

Surfing on the is by Web-Kit Open Source web browsing engine, OpenGL ES is used in order to process two-dimensional and three-dimensional images, and SQLite is used as data storage database.

Tools need to start

Before developing an android application, we need to install all of its necessary tools. This part is explaining all the important and required tools for obtaining Android environment to develop an android project. This application is run by Dalvik virtual machine, so each operating system that supports developer tools is able to support android's application. These operating systems could be one of the followings:

- Microsoft Windows (XP or later)
- Mac OS X 10.4.8 or later (Intel chips only)
- Linux

On one of these operating systems, we need to download and install a language programming environment and Java Development Kit(JDK). They are all explained in following.

Integrated Development Environment

In following some different types of development environment are emphasized for creating and handling an android project.

- **Android Studio** is the official IDE for Android application development, based on IntelliJ IDEA

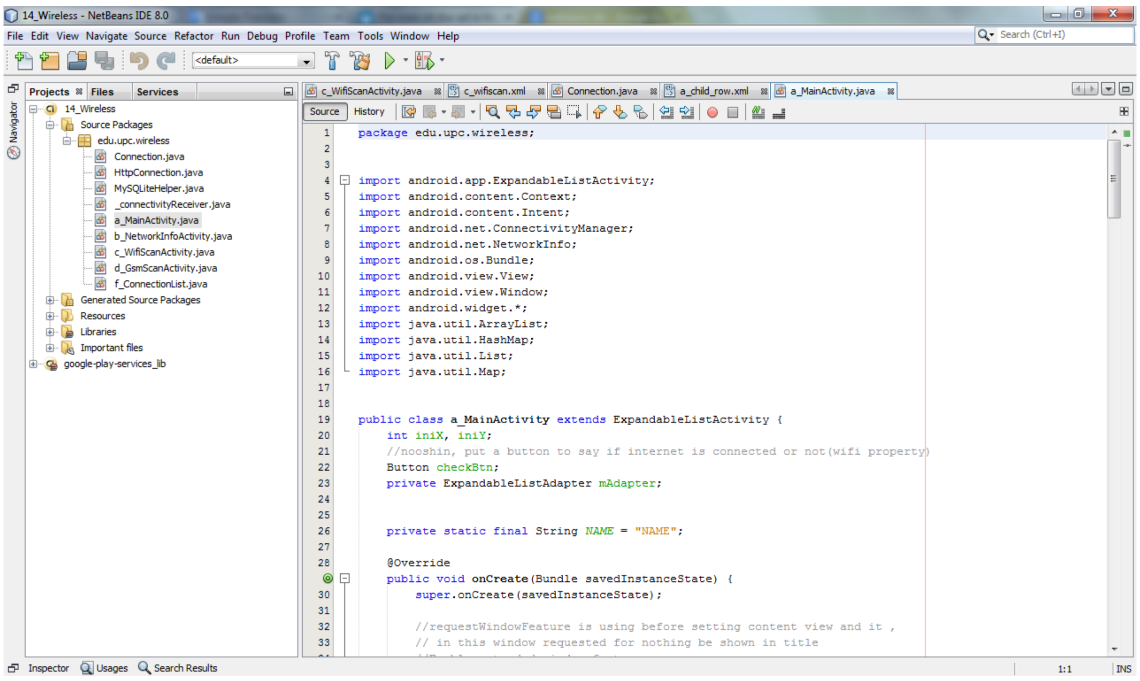
- **Eclipse** an open-source IDE (integrated development environment) particularly popular for Java development
- **NetBeans IDE** is an open source java based environment.

NetBeans IDE is the environment that I have used for this study. This IDE is a java based software development platform contains of modules as a software component to develop an application. In addition, the open source integrated development environment (IDE) is a platform in NetBeans for extending application by third party developers like video game industries. Beside of Java-Based platform, it can handle other programming languages such as C/C++, PHP and HTML5. This platform is declared as a cross-platform that runs on Microsoft Windows, Mac OS X, Linux, Solaris and other platforms supporting a compatible JVM.

One of the important modules presented by NetBeans IDE is NBAndroid module which is included of developing modules of Android. It can make a connection between NetBeans IDE and Android SDK in order to create, edit and execute android projects.

Properties provided by NBAndroid for android application are:

- Supporting core Android SDK
- Building, running and debugging of android projects that are supported by Android SDK.
- Containing LogCat viewer to tracking what is going on while the code is running.
- Containing Android XML files editors.
- GUI layout preview to design UI and interface of android application.



```java
package edu.upc.wireless;


import android.app.ExpandableListActivity;
import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.widget.*;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;


public class a_MainActivity extends ExpandableListActivity {
    int iniX, iniY;
    //nooshin, put a button to say if internet is connected or not(wifi property)
    Button checkBtn;
    private ExpandableListAdapter mAdapter;


    private static final String NAME = "NAME";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //requestWindowFeature is using before setting content view and it ,
        // in this window requested for nothing be shown in title
```

Figure 1. NetBeans IDE environment

Android SDK

The android software development kit (SDK) is developed by Google based on Java language. This software contains extensive set of android development tools such as debuggers, libraries and handset emulator, sample codes and tutorials. This is also cross-platform software supported by Linux, MAC OS and Windows XP and later.

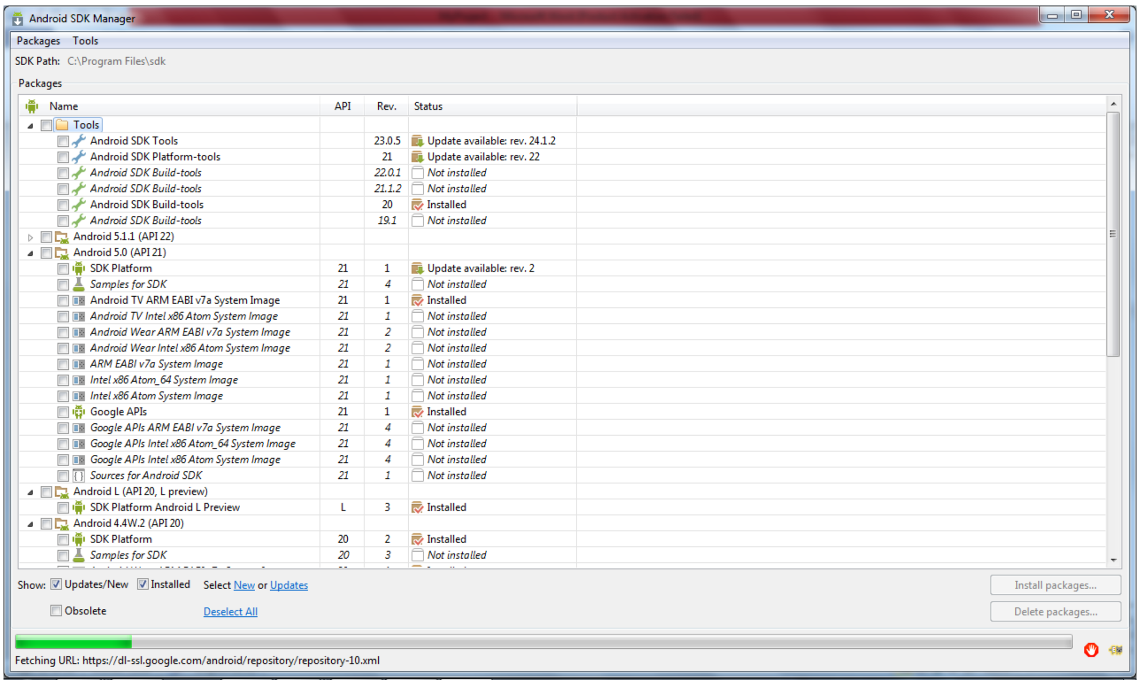
This software is an important and necessary platform for developing android applications that could be supported by NetBeans via setting plugins. In order to set its plugins to NetBeans IDE, have to follow tools/android SDK manager.

Android SDK doesn't include everything, so it needs to download each required package of android SDK separately based on needs of the application by Android SDK Manger. In following required packages for the project are named:

- The latest SDK tools those are necessary to be downloaded.
    - Android SDK Tools

- Android SDK Platform-tools

- Android SDK Build-tools
- SDK Platform of Android API 22 or the latest version of API
- Support library for additional APIs
- Android Support Repository
- Android Support Library
- Google Play Services for APIs related to Google play services
- Google Play Services
- Google Repository



SDK Path: C:\Program Files\sdk

Packages

Name	API	Rev.	Status
Tools			
Android SDK Tools		23.0.5	Update available: rev. 24.1.2
Android SDK Platform-tools		21	Update available: rev. 22
Android SDK Build-tools		*22.0.1*	*Not installed*
Android SDK Build-tools		*21.1.2*	*Not installed*
Android SDK Build-tools		20	Installed
Android SDK Build-tools		*19.1*	*Not installed*
Android 5.1.1 (API 22)			
Android 5.0 (API 21)			
SDK Platform	21	1	Update available: rev. 2
Samples for SDK	*21*	*4*	*Not installed*
Android TV ARM EABI v7a System Image	21	1	Installed
Android TV Intel x86 Atom System Image	*21*	*1*	*Not installed*
Android Wear ARM EABI v7a System Image	*21*	*2*	*Not installed*
Android Wear Intel x86 Atom System Image	*21*	*2*	*Not installed*
ARM EABI v7a System Image	*21*	*1*	*Not installed*
Intel x86 Atom_64 System Image	*21*	*1*	*Not installed*
Intel x86 Atom System Image	*21*	*1*	*Not installed*
Google APIs	21	1	Installed
Google APIs ARM EABI v7a System Image	*21*	*4*	*Not installed*
Google APIs Intel x86 Atom_64 System Image	*21*	*4*	*Not installed*
Google APIs Intel x86 Atom System Image	*21*	*4*	*Not installed*
Sources for Android SDK	*21*	*1*	*Not installed*
Android L (API 20, L preview)			
SDK Platform Android L Preview	L	3	Installed
Android 4.4W.2 (API 20)			
SDK Platform	20	2	Installed
Samples for SDK	*20*	*3*	*Not installed*

Show: [✓ Updates/New] [✓ Installed] Select New or Updates

Obsolete Deselect All

Fetching URL: https://dl-ssl.google.com/android/repository/repository-10.xml

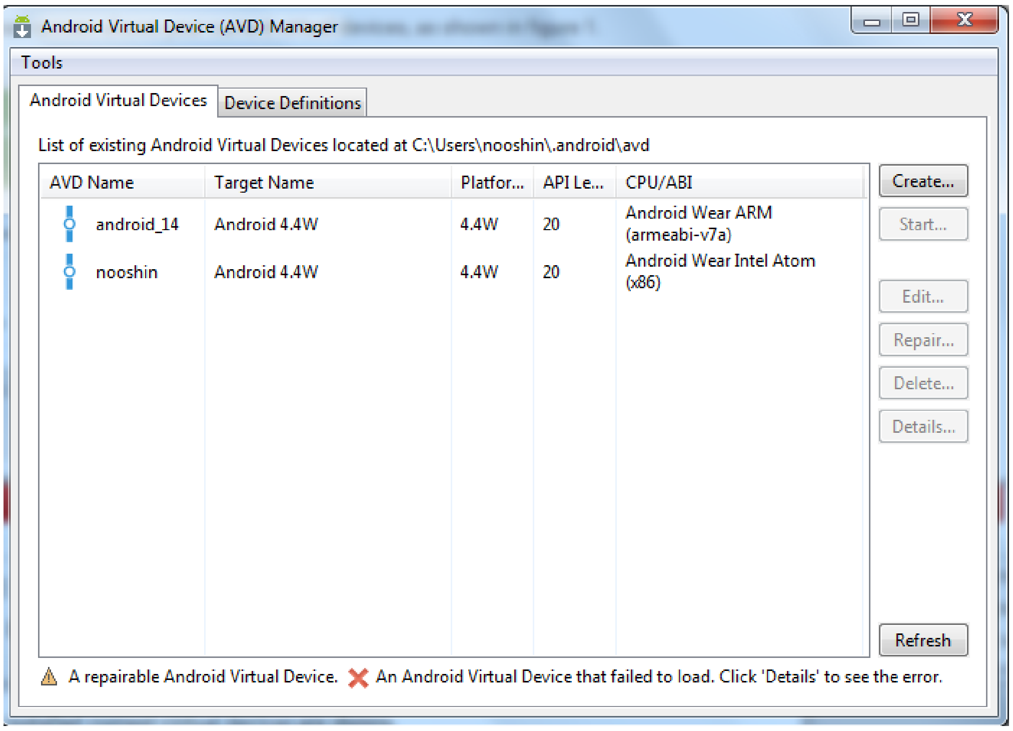
Figure 2. Android SDK manager interface

After installing these tools and APIs we are able to develop an Android Application.

AVD Manager0

This tool is used for graphical user interface to simulate the application on an Android emulator. To access AVD manager by NetBeans need to go tools/AVD manager from toolbar of software.

All the current installed virtual devices are shown in the main screen of AVD manager as shows in figure 3. We can have as many AVDs as we would like to have as Android Emulator. This emulator provided by AVD manager is used to test the application and see the process of application on computer.



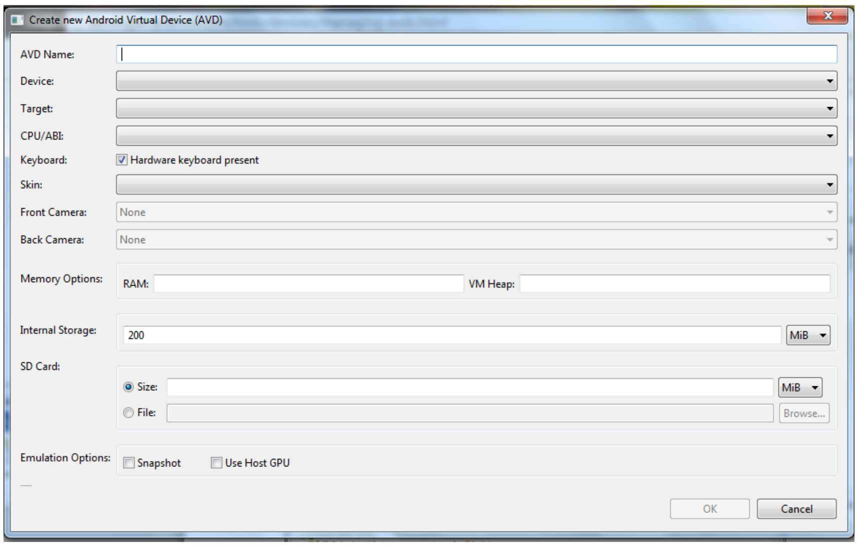
List of existing Android Virtual Devices located at C:\Users\nooshin\.android\avd

AVD Name	Target Name	Platfor...	API Le...	CPU/ABI
android_14	Android 4.4W	4.4W	20	Android Wear ARM (armeabi-v7a)
nooshin	Android 4.4W	4.4W	20	Android Wear Intel Atom (x86)

⚠ A repairable Android Virtual Device. ✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

Figure 3. AVD manger main screen

The figure 4 illustrates the process of creation a new virtual device by custom by hardware options exist in the window such as Device ram Size, Size of screen, Defining the device name, supporting landscape or Portrait view or both, accessible sensors and so on.


Create new Android Virtual Device (AVD)

AVD Name:	
Device:	
Target:	
CPU/ABI:	
Keyboard:	☑ Hardware keyboard present
Skin:	
Front Camera:	None
Back Camera:	None
Memory Options:	RAM: VM Heap:
Internal Storage:	200 MiB
SD Card:	● Size: MiB ○ File: Browse...
Emulation Options:	☐ Snapshot ☐ Use Host GPU

Figure 4. The Configure Hardware window when creating a custom device configuration

Emulator

Two ways for testing an Android project are defined, one by Android Simulator and the other using hardware device.

- Android Emulator

This virtual android device is supplied by Android SDK in order to test an android application on the computer. So it possible to prototype, test and develop the application without need of hardware device.

The Android emulator mimics all of the hardware and software features of a typical mobile device, except that it cannot place actual phone calls. It provides a variety of navigation and control keys, which you can "press" using your mouse or keyboard to generate events for your application. It also provides a screen in which your application is displayed, together with any other active Android applications.

As is indicated, the android emulator is set by AVD manager configurations, so we can have different emulators with different configuration to test Android platform. Once your application is running on the emulator, it can use the services of the Android platform to invoke other applications, access the network, play audio and video, store and retrieve data, notify the user, and render graphical transitions and themes.

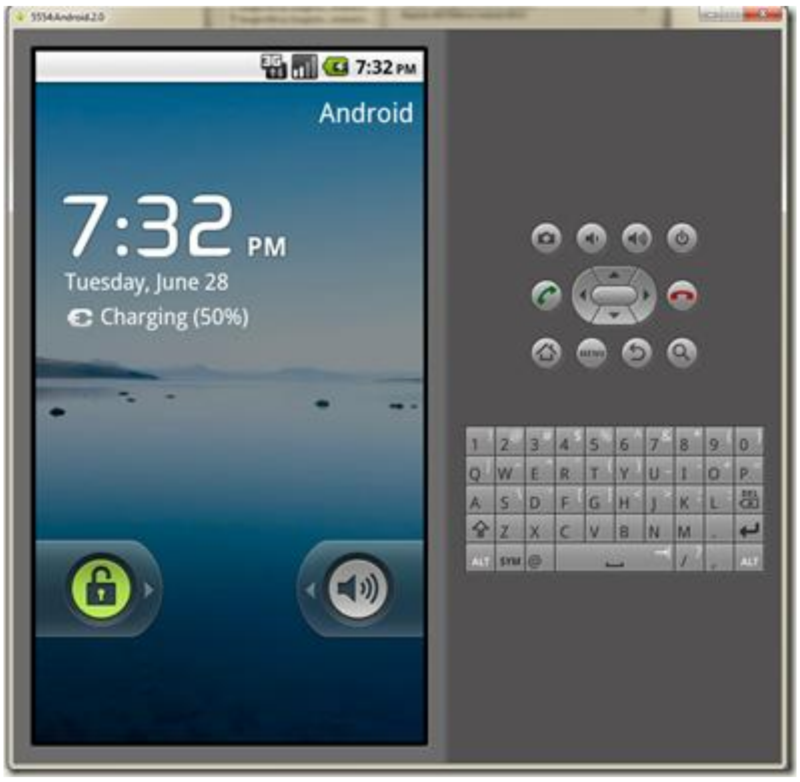


Figure 5: Android Emulator

The emulator also includes a variety of debug capabilities, such as a console from which you can log kernel output, simulate application interrupts (such as arriving SMS messages or phone calls), and simulate latency effects and dropouts on the data network.

When you run your app from Android Studio, it installs and launches the app on your connected device or emulator (launching the emulator, if necessary). You can specify emulator startup options in the Run/Debug dialog, in the Target tab. When the emulator is running, you can issue console commands.

- Android Hardware Device

Before releasing an application to users, it is important to test on a real device to see its performance clearly. For running, debugging and testing an application any device equipped by android Operating system is suitable. The tools included in the SDK make it easy to install and run your application on the device each time you compile. It is possible to install the application on device by command lines of ADB or using USB drivers to run the application on the device.

Android-powered devices have a host of developer options that you can access on the phone, which let you:

- Enable debugging over USB.
- Quickly capture bug reports onto the device by help of adb logcat.
- Show CPU usage on screen.
- Draw debugging information on screen such as layout bounds, updates on GPU views and hardware layers, and other information.
- Plus many more options to simulate app stresses or enable debugging options.

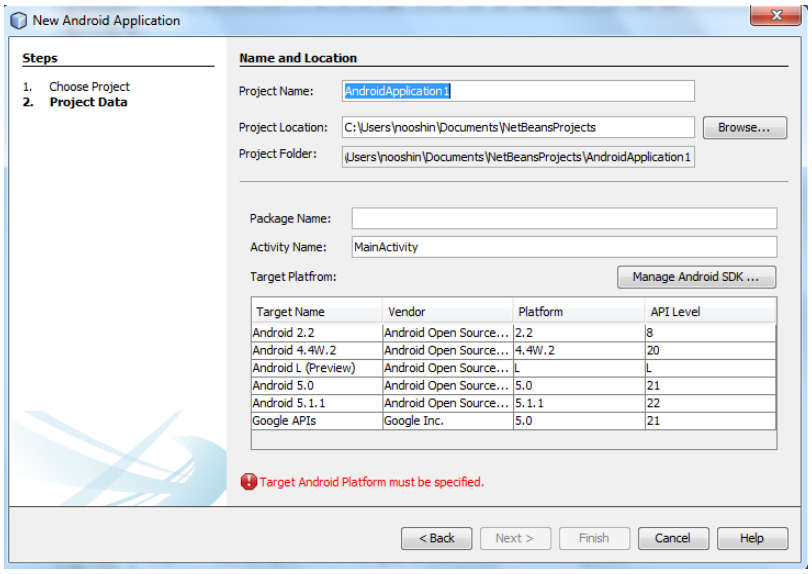
Summary

Setting up a development environment of android basically needs SDK and a java IDE.

Android Structure

Once we have plugged the SDK in the plug-in and install NetBeans IDE, we are able to start and create an android application, develop it and set it up, run and debug configurations.

Now we can start to create a new android project from file/new project, then in the appeared dialog, choose Android project and then click next button. In the new window, it will ask about the project name as the specific name of our project, project location and its folder. Package Name specifies its java package; the field of Activity name is to clarify of our initial and main activity. At end in Target Platform we can choose minimum SDK version that our application will be run on it.



Project Name: AndroidApplication1

Project Location: C:\Users\nooshin\Documents\NetBeansProjects

Project Folder: \Users\nooshin\Documents\NetBeansProjects\AndroidApplication1

Package Name:

Activity Name: MainActivity

Target Platfrom:

Target Name	Vendor	Platform	API Level
Android 2.2	Android Open Source...	2.2	8
Android 4.4W.2	Android Open Source...	4.4W.2	20
Android L (Preview)	Android Open Source...	L	L
Android 5.0	Android Open Source...	5.0	21
Android 5.1.1	Android Open Source...	5.1.1	22
Google APIs	Google Inc.	5.0	21

Target Android Platform must be specified.

Figure 6: creating an android project

In this part, a main structure of an android's package is defined. There are five folders in the left side of environment

- Source Packages
- Generated Source Packages
- Resources
- Libraries

- Important Files

Source packages

The source package of android project contains main Android components to build an application that includes all java source codes of application. In this part we have an overview on these components to understand how they are related to a real-world application. These components embedded in source package are main building blocks of an application. These perceptual components should be put and interact together to make a bigger whole. First we should have a conceptual draw of what we aim to create by lines and circles to have a general view of the idea and how we desire to relate these components together to make sense.

Main building blocks of android are upstanding requirement of an application that is broken into conceptual units to work independently on each of them and then by putting they together reach to a complete package. These components are named as activities, Intents, services, broadcast receivers and content providers. In following we have a short overview on each of these components.

Activities

An activity as one of the distinguishing features of the Android framework is the interface of application that provides a screen for user to see the application in order to do something and interact with the application and user. So we can say they are visible part of application. Usually applications have more than one activity that one of them is the main activity which appears when we launch the application, like the home page of a website, and then we can navigate among other activities.

Services

Services work like activities with a difference of their interface that services perform in background of device and are hidden from sights of users. They are used in order to run an activity invisibly for a while like playing music while user is working with another application. The point about services is that they have simpler structure than activities because of its requirement methods. Another point is its lifecycle which is handled by developer more than system. But we have to be aware about its shared resource consuming such as CPU or battery.

Content Providers

Content providers are data stores that facilitate applications to use, share and manage its stored sharable data. Because the nature of each application in android is to be isolated and work separately from other applications, so it means their data is not accessible by other application. In order to solve this, content providers are developed to pass through an application. For example, Contact Data is accessible by other application because of passing content providers through this application. Therefore, this application lets application to use its content by connecting to Contact Data application. The content providers use standard database methods of insert (), update (), delete (), and query () to have relatively simple interface.

Broadcast Receivers

The broadcast receivers are components of android that they are actively working in memory and background, like services are without interfaces and visual representation. They are mechanism of publish, subscribe, sending notifications in the device like sending notification when a message is received, when a call comes in or when the battery runs low.

Additional Components

There are some additional components of Android in source packages in the construction of above mentioned components. These are listed below:

Fragments

They represent a behavior or a portion of user interface in an Activity. It means that when we want to create a multi segment in the user interface of an activity, we need to use fragments to encapsulate UI components and activity behaviors inside of a module to swap into and out of the activity. So we can have a nested activity that each activity has its lifecycle and own layout.

Intents

Intents are messages that are sent among the major building blocks. They trigger an activity to start up, tell a service to start or stop, or are simply broadcasts. Intents are asynchronous; meaning the code that sends them doesn’t have to wait for them to be completed. Intent could be *explicit* or *implicit*. In an explicit intent, the sender clearly spells out which specific component should be on the receiving end. In an implicit intent, the sender specifies the type of receiver. For example, your activity could send an intent saying it simply wants someone to open up a web page. In that case, any application that is capable of opening a web page could “compete” to complete this action. When you have competing applications, the system will ask you which one you’d like to use to complete a given action. You can also set an app as the default. This mechanism works very similarly to your desktop environment, for example, when you downloaded Firefox or Chrome to replace your default Internet Explorer or Safari web browsers. This type of messaging allows the user to replace any app on the system with a custom one. For example, you might want to download a different SMS application or another browser to replace your existing ones.

In following of this part, I go through the components in details that I have used in my project. Hitherto I have explained the important components of an android project that they have to be put together for constructing an application and run perfectly. In following, I want to explain more about building block components that I've used in my project to have better cognition of these components, how they work together and what they have inside.

Activities

Usually an application has more than one activity, the visible part of an application that user can flip back and forward among them like websites. An activity could contain one button to start another activity of application like choosing sending message when you are in contacts list then a new activity of composing message appears. The method that is using in activities in order to navigate between activities of an application is "Back Stack" or in another word "last in, first out", that pushes an element to the collection of elements and pops to remove the last element that was added. Therefore when an application is launched, main activity of the application starts and then it interact with other bounded activities of the application. By starting new activity of application, the previous activity stops but maintains in the stack of the system. And when an activity starts it is pushed onto the back stack. Therefore, when user uses Back button, it will be destroyed and previous activity resumes.

An activity has callback methods to perform specific work appropriate to the changed state such as resume, stop, create or destroy. These callback methods are essential in developing a strong and flexible application and managing the lifecycle of the application. For example when the resume method is called, all the necessary resources that were interrupted should be resumed. By these callback methods the lifecycle of application is provided.

An activity interface is a subclass of Activity with implemented callback methods for different transitions in the application. The figure 7 lifecycle callbacks are shown:

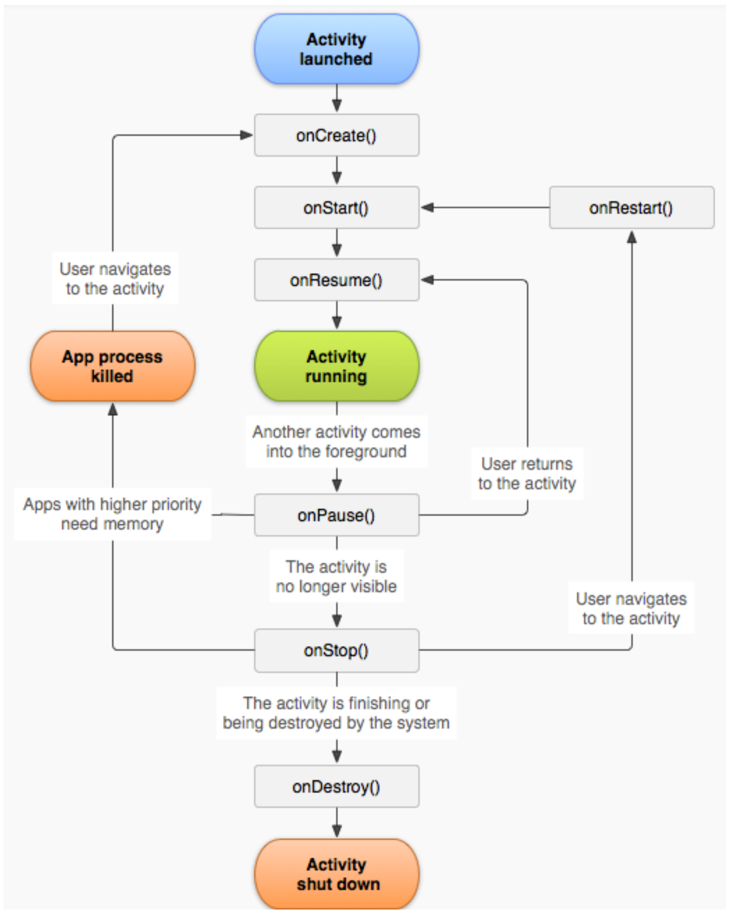

Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Figure 7: The activity lifecycle

Starting state

First we have starting state or activity launched when the activity of application does not exist in the memory. When we launch the program, the activity goes through all callback methods of application to run it. This is one of the most expensive operations in terms of computing time and battery's life. The application leads to "running state" from "starting state" by three different kinds of callback methods.

- **onCreate():** In this method all the essential components of that interface of application such as buttons, textboxes, and views are implemented and called.

- **onStart():** Called when the activity is becoming visible to the user. Followed by onResume ().
- **onResume():** refers to when application is running in foreground lifetime of an activity that starts by this method to interact with user.

Running state

The state of application that is visible and intractable by user like typing, touching screen or clicking buttons. When application is in this state it is using memory, resources and has to run as quickly as possible. Application can goes to stop state or paused state by onPause method.

- **onPause():** This method is used when user is leaving the application, but it doesn't mean destroying the application, the application is still running this activity is still visible and alive, partially transparent or doesn't cover the entire screen.

Stopped state

The application comes to this state by onPause method that the activity is not visible but is in memory. And activity could be brought back to in front by calling onResume() method or could be destroyed by onStop() callback method and be removed from memory of device. Also Stopped activities can be removed from memory at any point. We can use onResume, onCreate or onStart() methods for backing to Running State or use onDestroy() method to finish the application and go to Destroy State.

- **Destroyed state:** The activity could be destroyed by onDestroy() callback method from stopped state then the activity is in background or destroyed when the system needs its memory. A destroyed activity is no longer in memory.

Fragment activity

This component is responsible to present a behavior or part of user interface like maps in the interface of activity. It is kind of modular section of an activity with its own lifecycle that receive s its input events. Fragments have to be embedded in an activity and their lifecycle depended on the lifecycle of its host activity. This means when activity starts, they can be started too, but when activity is stopped they will be stopped too.

The reason that I used fragment activity in the project is for embedding the Google map in an activity. The Google map activity has its own behavior and structure and it needs to be implemented in own activity. Another point about this component is that they have to be imported from extra libraries “google-play-services-lib”, I have explained more about this library in following pages (page 27).

For managing the fragments, fragmentManager from the library of Android is utilizing. This should be called from getFragmentManager() from the activity. To define the location of fragment in the application, it uses findFragmentById(), to put take the place of fragment in an activity. The following figure shows lifecycle of fragment activities.

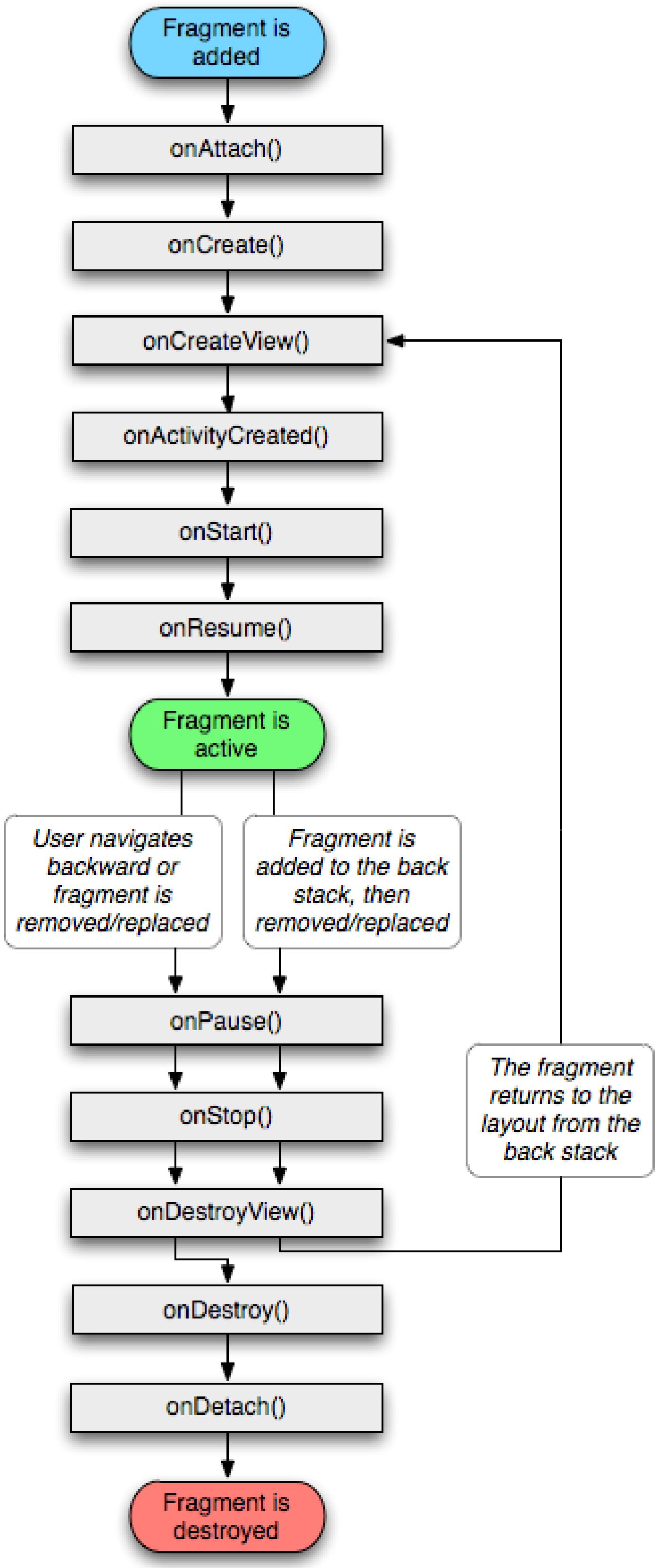


Figure 8. The lifecycle of Fragment

Broadcast Receivers

Broadcast receiver is a component of android with responsibility of responding to system-wide broadcast announcements. This class receives intents sent by sentBroadcast() and is implemented as a subclass of BroadcastReceiver. The object include of Broadcast Receiver starts by calling onReciver(Context, Intent) method.

The lifecycle of the broadcast receiver considered as foreground process that keeps running while the device is on until pressure of lack of memory prevents its running. Most of broadcast receivers are generated by system like when the battery of device is low. This component doesn't have an interface to show user, they work like a notification or alert when an event happens. They are kind of "gateway" to other components of application.

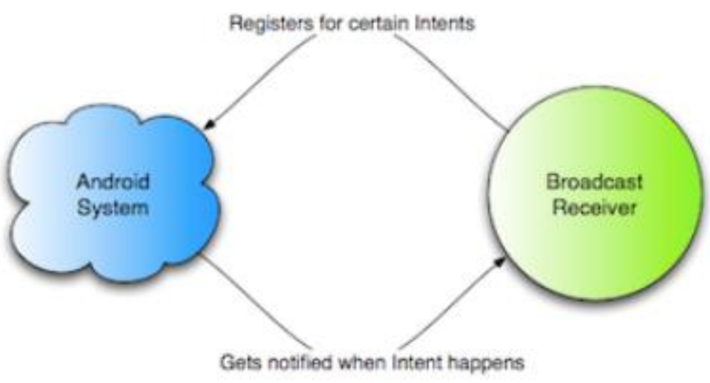

Registers for certain Intents
Android System
Broadcast Receiver
Gets notified when Intent happens

Figure 9: lifecycle of broadcast receivers

Heretofore in this chapter, I have explained some of the most important components, and what I have used in my application that they should put together to make an application. In following I will demonstrate other packages of an android application.

Generated Source Packages

- This package includes of dynamically generated R.java class that is created during built of code and before complete compiling of code to define all assets of the program. This class contains public contains of program to define value of resources from string to android widgets and to layout. This java class aids to find errors of code earlier than compiling time.
- In the other meaning, the automatically generated R.java connects the java classes to resources of application and makes a relation between them. We should never modify it, because it can be updated automatically due to any changes in the resource directory.

Resources:

This directory is made of all User Interface components that user can see and interact with, such as Drawable files, structured layouts objects, String values, static content that is used in code, animation instructions, bitmaps and some special modules like menus. These resources are always maintained separately in various sub-directories under res/ directory of the project. The User Interface allows us to build the graphical part of application by variety of pre-build UI provided by Android. In order to use these elements we have to import View or ViewGroup class in the application. Android provides a collection of both View and ViewGroup subclasses that offer you common input controls (such as buttons and text fields) and various layout models (such as a linear or relative layout). Also each type of resource has its own specific subdirectory (res/ directory). For example, here's the file hierarchy for a simple project:

```
MyProject/
         src/
             MyActivity.java
         res/
            drawable/
                         icon.png
```

```
        layout/
                activity_main.xml
                info.xml
        values/
                strings.xml
```

As it is mentioned, the res/ directory contain all the resources in various subdirectories. In Following I give you more detail about the resource directories supported inside my project's res/ directory.

- **Drawable** resources are using for graphical concepts of the Project like bitmap files with format of JPEG, GIF, PNG or XML files that should be drawn in the screen and they are accessible into code by R.drawable class.
- **Layout** files consist of XML files that define screen layouts of the application. In fact these XML files are User Interface architecture of an activity or a component. They saved in res/layout/filename.xml that they are provided by a hierarchy of views. A layout defines and organizes widgets like buttons, text filed, checkbox, image and other type of views to use. This is a way to separate design of user interface from codes of application. By passing the ID of layout in setContentView () of activity a layout is set to an activity java code. They are accessed in the code by R.layout class.
- **Menu** file defines XML files related to menus of the application store in res/menu/filename.xml and are accessed by R.menu class, includes of Option Menu, Context Menu or submenu that can be inflated with MenuInflater. This xml file also contains of some elements and attributes. <menu> is required element in the menu xml file as the root node. It contains two elements itself <item> and/or <group>.
- **Values** include of other xml structure resources to define simple values such as strings, integers and colors.

- **Colors** is defined in the tag of <resources> <color name :””> and in the tag of color, the RGB value of color with # should be defined, like #FFFFFF.

- **Styles** is a resource to describe UI’s format and look. So it should be defined in the <resources> element as the root node, and <style> element to define a single style that contains <item> element. Style.xml can be made of two attributes, name as a necessary attribute for the name of style and parent as a style resource. Another way to describe <style> element is to use <item> element for defining a special properties for a style also each item of style has name as required attribute to define the name of each item in style element.

Libraries

Each new project has accessibility to the default libraries of android such as Activity, broadcast, Service, Intent, View, Application and others. But some packages need extra libraries that are not defined in the default Android library. So the un-default libraries have to be embedded to project manually.

Google-Play-Services-lib

This library is used to **develop an app using the Google Play services APIs.** To develop an application using the Google-Play-Services-lib, first we need to set up the project with the Google Play services by SDK manager. This library gives the application more features that are not defined in default library of android for attracting users on a wider range of devices.

I have used this library to take advantage of the latest Google-powered feature, GoogleMap to download the map tiles on demand. In addition, this Google-Based feature could be updated automatically through Google Play Services and will send newest Google offers to programmer. In following I have explained about Google play services that need this library.

-

Google Play Services

Google play service is a private background service that covers a large various Google's services such as Google Play Game Services, Location APIs, Maps, Ads, Google+ and other APIs used by Android. The process of this service is to establish communication between Google applications and third-party applications. The advantage of Google Play Services is automatically updating its Google Powered features and propagating the application experience.

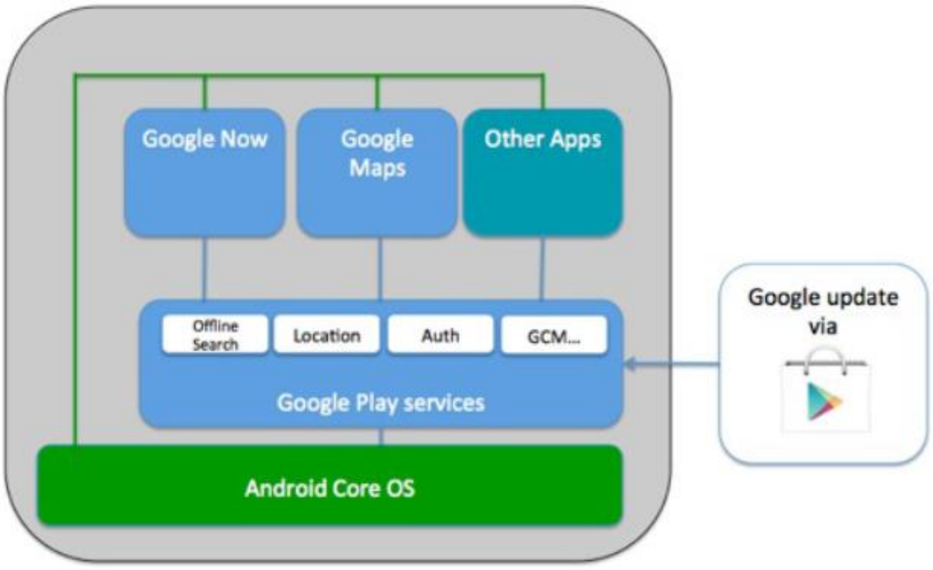


Figure 10. Process of Google Play Services

Google Play Services are considered as a channel of background services for Google, this means it updates its features without getting user's permission or manufactures. This indicated updates by Google Play Services don't need to push large updates to operating system.

Furthermore, a common Standard authorization is shared by Google Play Services for all products and applications of a device.

Google-Support-v4

The Android Support Library package is a set of code libraries that provide backward-compatible versions of Android framework APIs as well as features that are only available through the library APIs. Each Support Library is backward-compatible to a specific Android API level. This design means that your applications can use the libraries' features and still be compatible with devices running Android 1.6 (API level 4) and up.

Google map needs to be implemented through a separate activity, so the most suitable component to use in an activity to have Google map beside of other elements, is fragment activity. This component should be imported by Google-Support-v4.

Important files

In this directory one of the most important components of android is placed to configure files of the application.

Manifest File

Android has dedicated specific permissions for certain dangerous operation; so by this way has possibility of managing security of application installed on device. In addition, the application asks user for grant or ignore all these permissions as soon as application is installed on device.

Accessing to system of device needs to define AndroidManifest.xml in the root directory file to declare the activity of the application. This file prepares important information of application like the number of activities of program. Its responsibilities are:

- Specifying the name of application.

- Specifying components that are used in application like activity, broadcast and which component is a host of application.
- The important role is specifying permissions of application. The permission has ability of accessing to protected parts of API and interacting with other applications of the device and the application's components.
- The system will understand each application is launched under which conditions.
- Also it can specify the minimum level of API that application requires.

The manifest.xml has its unique structure and legal elements. So we are not able to add new attributes or elements in its structure. The manifest file starts with a tag <manifest> and all elements are defined inside of this tag and then <application> tag. They are unique required elements that should be called but the other elements can be used more than one time or not be used.

The other elements could be used more than one time or not be used are <activity>, <provider> and <service>. Therefore, inside attributes of each element are defined. Attributes are used to assign properties of an element like defining a theme for an activity. Although attributes are optional, some of them are used for specifying purpose of an element such as android: name to introduce the class name of the activity.

One of the important elements of manifest is Permission elements to define limiting access of a part of code to data on the device. They are used for protecting critical data and code of application. Each label of permission demonstrates the restricted feature action. The element to declare a permission is <uses-permission> in the manifest. So when the application installed on the device, the installer will check the authorities of the application that if they are signed by the application certificate or not.

System Services

In this part I go through explaining services and important classes that I needed to use for developing the project. Android has a number of system services that are always on, always running and readily available for developers to tap into. Some notable services are Location service, Sensor service, Wi-Fi service, Alarm service, Telephony service, Bluetooth service, and so on. System services are started at boot time and are guaranteed to be running by the time your application launches.

In this part, I'll explain some of the system services that I used in my project. First, I'll take a look at the Location service to demonstrate some of the concepts that are applied to most of the location services. Then, I'll explain Wi-Fi services information to our status updates via the Wi-Fi service.

To use Android system services we need to access to their own manager. These managers of each service use getSystemService method for passing in Context of service name.

Location Services

In order to construct a location-aware application, Google play Services of a standard Android platform provides us to access location data from available location providers by implying classes of location services in Android. Some classes to retrieve information of location are described in following:

- LocationManager

This class provides accessing to other system location services to retrieve periodic updates of geographical locations. This class retrieves data from all local GPS and network location providers of the device's area.

- LocationListener

It can receive notifications of the updates and changes of the location, which is used in the method of LocationManager, requstLocationUpdatses (String, long, float, LocationListener). So it provides callback methods which call when location is changed.

- LocationProvider

It is an abstract superclass for location providers for periodic reports of geographical position of the devices.

- Criteria

This is the class help to choose the best and suitable Location Provider, and it prepares a set of required properties of the LocationProvider. It can provide accuracy, power stage, ability to report altitude, speed, bearing and monetary cost.

- GeoCoder

It is a class to retrieve geocoding and reverse geocoding to transform the longitude and latitude coordinates to the street address, postal code, city and other information of the location by reverse geocoding. This amount could be different for different location, that means for one location maybe it could show all information and for another just the name of city.   In the GeoCoder a java.lang.object class, locale is called to specify language/country/variant combination of the location. And by the getDefault () method, have an appropriate and preferred locale of the user.

Wi-Fi Services

The service has some classes that provide us information about Wi-Fi connections, monitoring Wi-Fi connectivity and provides the primary API for managing all aspects of Wi-Fi connectivity. The manager element that Wi-Fi services pass through is Wi-Fi Manager.

- WifiManager

WifiManager is a class manger that monitors Wi-Fi connectivity and manages them. Also does broadcast Intents whenever the connectivity status of the Wi-Fi network changes, using an action. For example when an active Wi-Fi connection has been established, we can use the getConnectionInfo method on the Wi-Fi Manager to find information on the connection's status.

Also we can get a list of current network configurations such as ID, SSID and other details.

- WifiInfo

WifiInfo is another sub-class of Wi-Fi Services to provide us all information about device's active or in process Wi-Fi connection. The features such as Wi-Fi Frequency, return the BSSID and so on.

Connectivity Services

This service provides us the internet connection of device and its status. This service is used to monitor the states and configure failover settings, and control the network radios and set our preferred network connection. Like other services, this service needs to be implemented by its own manager and pass it in Context.CONNECTIVITY_SERVICE.

- ConnectivityManager

The ConnectivityManager represents the Network Connectivity Service to answer queries about the state of network connectivity. At the other hand, it notifies applications when network connectivity changes. The duties of this manager is monitoring network connections, Send broadcast intents when network connectivity changes, attempt to "fail over" to another network when connectivity to a network is lost, Provide an API that allows applications to query the coarse-grained or fine-grained state of the available networks and Provide an API that allows applications to request and select networks for their data traffic.

Google Map

The Google map object prepares an object in application to show the Google map in the application with the same appearance of the Google map application and some of its features. But there are some differences between the main application of Google map and its object in an application.

One of them is this API is not included personalized contents and icons of places. Another, if it shows, all of them are not clickable.

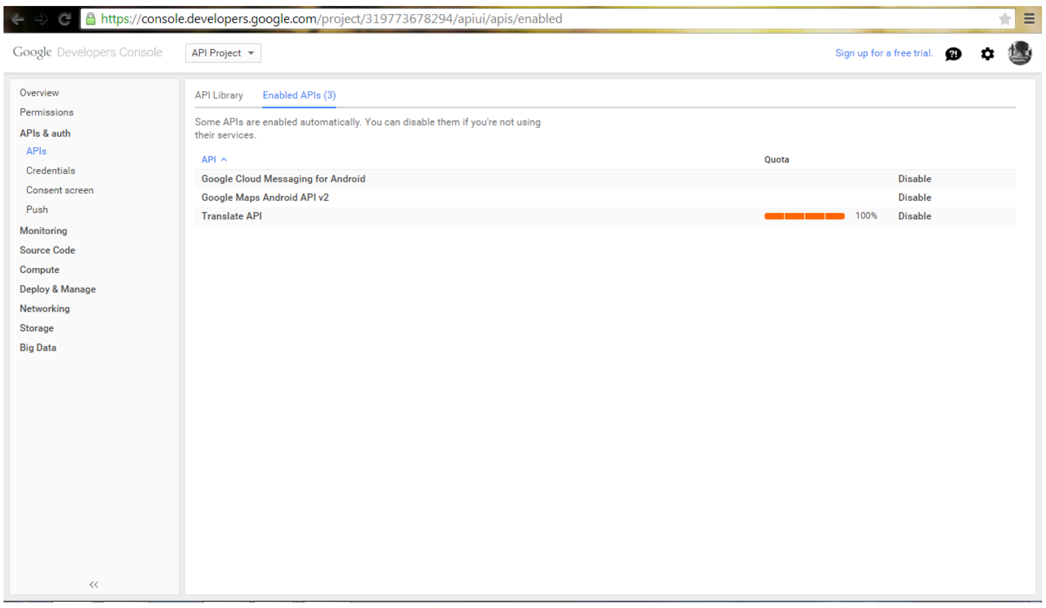
The object of “com.google.android.gms.maps.GoogleMap” implements a main class of Google Map Android API that includes all methods of the map. So it provides automatically access to Google Maps servers, data downloading, map display, and response to map gestures. In order to implement the Google map object, MapFragment or MapView are required to be added in the main threat application. MapView or MapFragment can display the map by the provided objects in location based services of Android. [13]

The important is for developing a Google map in the application needs to get a certificate from android and Google map API key and add this key to the application. The point is this key is free and can support an unlimited number of users to use the Map API in their application. Therefore, we have to prepare a Maps API key from the console of Google developers by following link https://console.developers.google.com and then add the key to the AndroidManifest.xml. This key is considered as a specific certificate/package pairs. The application with same certificate can use the same API too.

```
<meta-data android:name="com.google.android.maps.v2.API_KEY"

             android:value="AIzaSyDiNg51xYmg_330JV5aSYxIAJ4rGK4CLb8"/>
```

The Google map API key is a SHA-1 fingerprint key which is a cryptographic hash function to create a key based on the short form of the application digital certificate because of its uniqueness.

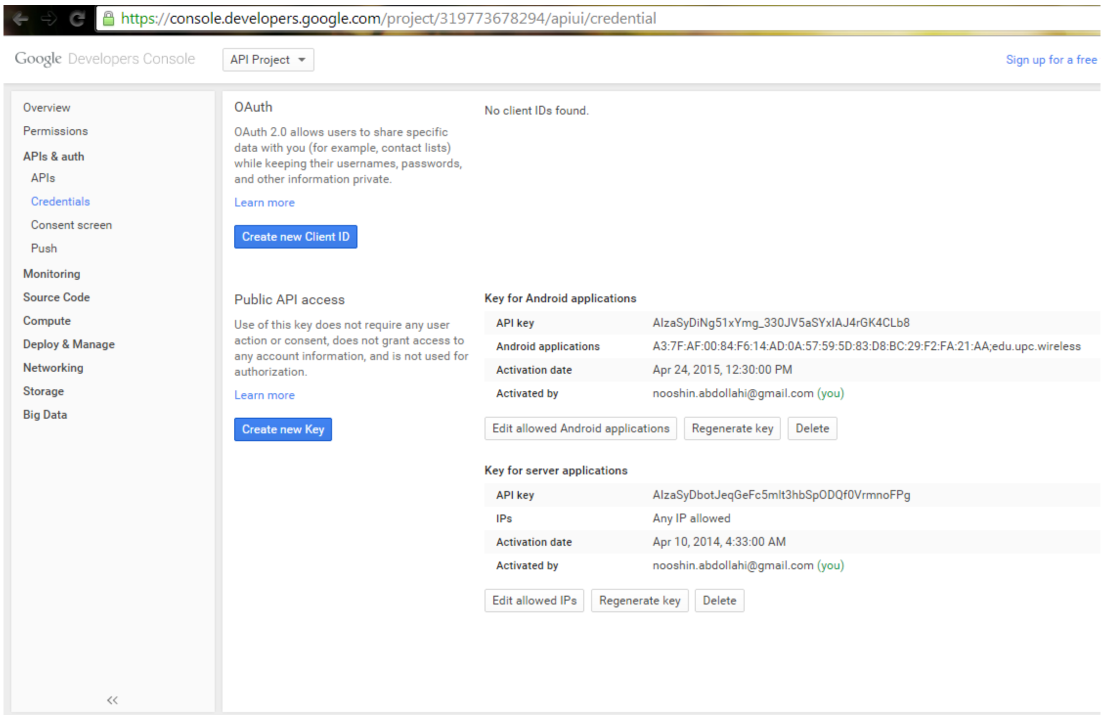
Therefore, we have to create AN API object in the Google Developer Console. To do that at first needs https://console.developers.google.com/ link to go to which is the Google Developers Console and then after choosing the API project, create the project in this console. To see the list of our enabled APIs, we have to follow APIs & auth/APIs/Enabled APIs that we need Google Map API.



Some APIs are enabled automatically. You can disable them if you're not using their services.

API	Quota	
Google Cloud Messaging for Android		Disable
Google Maps Android API v2		Disable
Translate API	100%	Disable

Figure 11: Console of Enable APIs for the application



OAuth

OAuth 2.0 allows users to share specific data with you (for example, contact lists) while keeping their usernames, passwords, and other information private.

No client IDs found.

Public API access

Use of this key does not require any user action or consent, does not grant access to any account information, and is not used for authorization.

Key for Android applications

API key	AlzaSyDiNg51xYmg_330JV5aSYxIAJ4rGK4CLb8
Android applications	A3:7F:AF:00:84:F6:14:AD:0A:57:59:5D:83:D8:BC:29:F2:FA:21:AA;edu.upc.wireless
Activation date	Apr 24, 2015, 12:30:00 PM
Activated by	nooshin.abdollahi@gmail.com (you)

Key for server applications

API key	AlzaSyDbotJeqGeFc5mlt3hbSpODQf0VrmnoFPg
IPs	Any IP allowed
Activation date	Apr 10, 2014, 4:33:00 AM
Activated by	nooshin.abdollahi@gmail.com (you)

Figure 12: Credentials page in the Google Developers Console

Then in the Credentials in the left side, we can see the API key of android application that as has been mentioned is a SHA-1 fingerprint code. Thus, the API key of application has to be added in manifest of the application. So, when the application runs, from manifest, Google Maps Android API v2 will read the key of application, and sends it to the servers of the Google Maps to receive confirmation of Google Maps data. This key should be added in the <application> element of manifest, that I have more specified it in the chapter 3.

Now that the permission and confirmation of using Google Map API is ready, the developing of a way to draw a line is needed. In this class package are some simple ways to draw a shape in the application's basic map.

The three shape types provided by this class are

- A **Polyline** shape draws paths and routs on the Google map by a series of connected lines between points.
- A **Polygon** shape is created to have an enclosed shape on the Google map.
- A **Circle** is show of a circle on the Earth's surface.

A Polyline provides our need of drawing route on the Google map of the application when the user is outdoor and it is using internet connection of mobile.

Debugging

The Android SDK provides most of the tools that you need to debug your applications. You need a JDWP-compliant debugger if you want to be able to do things such as step through code, view variable values, and pause execution of an application. If you are using Android Studio, a JDWP-compliant debugger is already included and there is no setup required. If you are using another IDE, you can use the debugger that comes with it and attach the debugger to a special port so it can communicate with the application VMs on your devices. The main components that comprise a typical Android debugging environment are:

Adb

Adb acts as a middleman between a device and your development system. It provides various device management capabilities, including moving and syncing files to the emulator, running a UNIX shell on the device or emulator, and providing a general means to communicate with connected emulators and devices.

Device or Android Virtual Device

Your application must run on a device or in an AVD so that it can be debugged. An adb device daemon runs on the device or emulator and provides a means for the adb host daemon to communicate with the device or emulator.

Session 3

Abstract: *In this chapter, I am going to explain how I have created and developed my project by using android components, tools, methods, classes, libraries and so on. The idea is to define an android tool which recognizes the user's activities such as running, walking, bicycling and driving while the user is connected to Internet. This is done by a simple body position independent algorithm and doesn't need to use embedded Android sensors. In order to fulfill the goal of the algorithm, we need to collect data of device's location regularly. It is mentionable that "location based services" is one the most important services has been imported and applied in my project. This service provides me many benefits for retrieving the information about current location of the device, processes of gathering data and useful information about location nearby. So I have proposed a development for location based of Android services which gives us required information based on device's location.*

The Project definition

An android project is an appearance collocation of activities, services and broadcast receivers to work together and their incorporation. The project covers of two main building blocks of android, Activity and Broadcast Receiver, and other additional components Fragment activity and Intent. As it has been mentioned in previous chapter, each component works individually and works together by fitting them properly.

The figures, 13 to 14 show what looks like the project at end to achieve the goal.

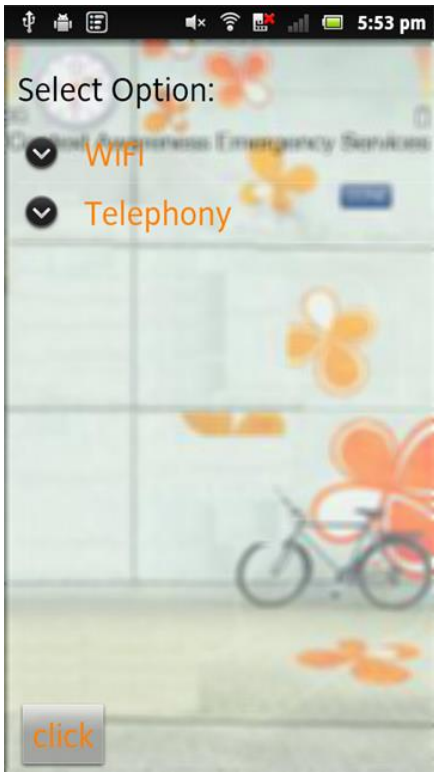
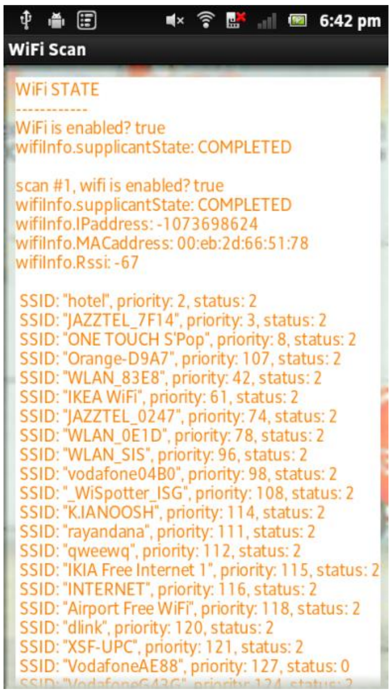


Figure 13: The main screen of application



```
WiFi STATE
------------
WiFi is enabled? true
wifiInfo.supplicantState: COMPLETED

scan #1, wifi is enabled? true
wifiInfo.supplicantState: COMPLETED
wifiInfo.IPaddress: -1073698624
wifiInfo.MACaddress: 00:eb:2d:66:51:78
wifiInfo.Rssi: -67

 SSID: "hotel", priority: 2, status: 2
 SSID: "JAZZTEL_7F14", priority: 3, status: 2
 SSID: "ONE TOUCH S'Pop", priority: 8, status: 2
 SSID: "Orange-D9A7", priority: 107, status: 2
 SSID: "WLAN_83E8", priority: 42, status: 2
 SSID: "IKEA WiFi", priority: 61, status: 2
 SSID: "JAZZTEL_0247", priority: 74, status: 2
 SSID: "WLAN_0E1D", priority: 78, status: 2
 SSID: "WLAN_SIS", priority: 96, status: 2
 SSID: "vodafone04B0", priority: 98, status: 2
 SSID: "_WiSpotter_ISG", priority: 108, status: 2
 SSID: "K.IANOOSH", priority: 114, status: 2
 SSID: "rayandana", priority: 111, status: 2
 SSID: "qweewq", priority: 112, status: 2
 SSID: "IKIA Free Internet 1", priority: 115, status: 2
 SSID: "INTERNET", priority: 116, status: 2
 SSID: "Airport Free WiFi", priority: 118, status: 2
 SSID: "dlink", priority: 120, status: 2
 SSID: "XSF-UPC", priority: 121, status: 2
 SSID: "VodafoneAE88", priority: 127, status: 0
 SSID: "VodafoneG43G", priority: 124, status: 2
```

Figure 14: the second activity

My project is consist of eight java classes, six xml files of the layout and one manifest to organize all the activities of the program. From 8 java classes 4 of them are activities as interface of the project to interact with the user, one of them is Broadcast receiver to work in background. And two other java classes are developed in order to have database.

From xml files of layout, 4 of them are the layout of the project related to each activity and other two xml files are embedded in the main activity of the program.

As this program is working as background and foreground, is included both activity intents and broadcast intent. When the android device is on, its connectivity to internet is checking by the broadcast intent to retrieve if the user is connected or not and if is connected, which type of internet connection is.

The List of activities in the source package is listed below:

- a_MainActivity.java
- c_WifiScanActivity.java
- d_GsmScanActivity.java
- f_ConnectionList.java

Other .java classes of the source package are:

- Connection.java
- HttpConnection.java
- MySQLiteHelper.java
- _connectivityReceiver.java

In addition in the resources/layout route of the application, four xml layout with the name of:

- main.xml
- c_wifiscan.xml
- d_gsmscan.xml

- f_connectionlist.xml

They are embedded in a_MainActivity.java activity for representing the list of Wi-Fi and GPS.

So, by running the application, a_MainActivity.java as the main activity of the program appears which includes of a list by the names of Wi-Fi and Telephony that by clicking on each one goes to their related activity.

Both activities have same structure included of the five textboxes. That they show the requested information of connection, a textbox to show the Location of the user, another to show the activity of the user and at least a textbox which declares distance of user from its previous location that was connected to internet and the velocity.

After these textboxes is a button linked to the f_ConnectionList.java that is a page to show all the connections saved in the database. And at the end a Google Map fragment is added to draw a line between points that user is connected.

_connectivityReceiver.java is a BroadcastReceiver that it doesn't need the user's interaction with application. While the cell phone is running, it works to retrieve the requested data.

In this chapter, that all the progress of the program is included, at first all the algorithm, classes, libraries and activities are presented. Afterward, the use of these in each activity and java class is defined.

Part 1: Android User Interface

In the first part I am going to introduce the UI module of my project that has focused on developing the first component of application for user to interact with and the unique feature that it is dual approach to UI via both Java and XML. Here I explained how this user interface works when it is launched by user. Through this activity, we need to use Java and XML together and we have to know how to connect them to each other.

Beside these two most important elements of building activity, I have explained about layouts, views, menus, intents, Google map, how to handle Java events such as button clicks, and other widgets.

Two Ways to Create a User Interface

There are two ways to create user interface of project. One is declarative, and the other one is programmatic. They are quite different but often are used together to get the job done. The best way is to use declarative approach which means applying XML to declare all the static interface elements such as layout of screen, and all widgets like a button that we declare how looks like. Then for interaction of various widgets we use programmatic way by Java to say what a defined button does.

Views and Layouts

Layouts as the screen of an activity includes of views, everything that we can see in a layout like button, label, textbox. They are organized by layout for grouping together or group some elements together to make a complex user interface structure. It means a layout can contain other children and allocate them space that each child is a layout itself. The layouts that I have used in my project are LinearLayout and RelativeLayout.

- **LinearLayout**

It is simplest and common layout that its children are set next to each other horizontally or vertically. In the LinearLayout the order of setting element are important that allocates required space that that they need and allocates desired space to each child based on their adding order.

- **RelativeLayout**

RelativeLayout lays out its children relative to each other. As such, it is very powerful because it doesn't require you to nest unnecessary layouts to achieve a certain look. At the same time, using RelativeLayout can minimize the total number of widgets that need to be drawn, thus improving the overall performance of your application. Having it said that, RelativeLayout requires each of its child views to have an ID set so that we can position it relative to other children.

Building Project's Activities

Firstly, for building the project we should define a Name for the project that the entire project will be organized under it. Then we should define the Android system will to run the application on that could be any platform, either standard or proprietary.

Then have to indicate the package name of project that designates a java package. Then we are able to create an activity that is one of the principle parts of this project. So the project is started by building the first activity name as MainActivity and developing it.

Part 1: Main Activity

My goal of this activity is to define the below figure 15 that runs when we lunch the application.

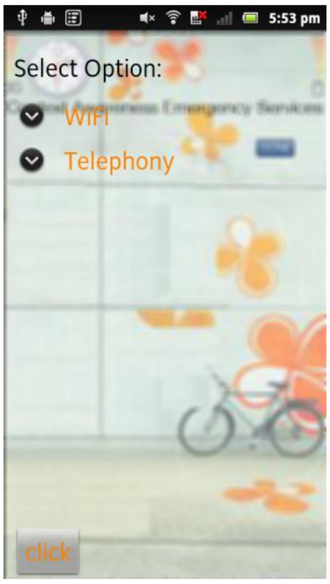


Figure 15: First snapshot of A_MainActivity

The Main Activity layout

I have started my project by designing its main screen layout with the name of a_main.xml in the res/layout folder. This layout consists of a menu and its submenus and it launches as main screen when the application is run.

The first screen of project shows a list that it is included of a TextView and an ExpandableListView widgets. The TextViews shows the title of menu and the ExpandableListView widget is used for having a vertical expandable list, Wi-Fi and Telephony and their submenus. These widgets are defined inside of a LinearLayout that it is a View Group tag and then it is connected to its related java code by referring to R class in its related java code.

Each widget has some properties that the important widget properties have to be specified in this xml file. Two properties that I have used are layout_width and layout_height to determine how much space belongs to each widget by asking from its parent layout. Although I could define a specific amount for width and height, because the application will be run on different devices with different screen sizes, is better to choose fill_parent or wrap_parent to have a relative size for each component. Fill_perent means that the widget will allocate all the available screen size for itself. I gave this fill_parent property to RelativeLayout screen that it covers all the screen of device with any kind of screen size. In the other hand, in some widgets like TextView, ExpandableListView, Button, I have used wrap_content for their width and height that means it requests for as much as possible space to display its own content.

One of the important and necessary properties is ID that should be identified for each widget. It is the unique identifier for a widget in a layout for calling the widget in the java code. It has to be defined in the in this way "@+id/menuexpandable". The other property is text property which it is used for some kind of widgets such as Button or TextView and they can specify a text inside a widget.

```
<?xml version="1.0" encoding="UTF-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/menuexpandable"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="@drawable/fons_blur">
   <TextView
      android:id="@+id/menuexpandableTextView"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginTop="20dip"
      android:layout_marginBottom="10dip"
      android:layout_marginLeft="10dip"
      android:textSize="24sp"
      android:textColor="@color/black"
      android:text="Select Option:"/>
   <ExpandableListView
      android:id="@id/android:list"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_below="@id/menuexpandableTextView"
      android:layout_marginLeft="10dip"/>
</RelativeLayout>
```

Strings Resource

In order to applying any changes in resources of a project, that applies on whole the package, Android has a policy to keep data in separate files. A separate file to save all the layouts of project, or defines another file name as string.xml to save values to hold text values such as button text or title text. Later in this chapter has been explained more about this file that includes of name/value.

The MainActivity Java Class

The layout of the main activity is ready, now I can define its java codes. Its java class, a_mainActivity.java is already created in com.wireless.edu java package.

An activity needs to start needs to start by defined sub-classes in the Android's framwork. So we should override the defined inherited methods of used sub-class. So firsly, we should use onCreate() necessary method to override all the widgets, describe the screen and starts its state machine. In this screen, onCreate() method has responsibility of set up the expandable list menu that response to clicks and connect us to other activities.

When we use public void onCreate (Bundle savedInstanceState), we need to pass a small amount of data by Bundle through onCreate() method which this bundle is prepared by super.onCreate(savedInstanceState). Also, this method provides the original small amount of parents into the activity by the intent that started it.

The first thing after calling the super method is to load the UI xml file in the activity and inflate it into the java memory space that for this activity it is called "inflating from XM". This is done by setContentView (R.layout.main) method. So java will be able to open its related XML layout, parses it and prepares a space for each element that is defined in the XML.

As I have mentioned in session 2, R class is an automatically generated class that connects Java and XML and all the other resources together to communicate. As you can see in the figure, there are two elements, one TextView filled by Select

Option and an Adapter List with sub-list of Wi-Fi and Telephony. In order to preview the list I have used ExpandableListAdapter.

ExpandableListAdapter Function

In the main activity we need to create a list group with its children. In order to create a list, adapter is the class to be used, it can map static data defined in xml to group and child view for initiating a list. In the separate xml file the views to display in a group are defined, then they map from keys of each element. This Process is same for a child too with one-level deeper. The first List corresponds to the group of the child, the second List corresponds to the position of the child within the group, and finally the Map holds the data for that particular child.

After defining an object of adapter, we fill it with SimpleExpandableListAdapter (Context context, List<? extends list <? extends Map < String, ? >>> childData, int childLayout, String[] childFrom, int[] childTo).

```
mAdapter = new SimpleExpandableListAdapter(
            this,
            createGroupList(),
            R.layout.a_group_row,
            new String[] { NAME },
            new int[] { R.id.menuexpandibleGroupTextView },
            createChildList(),
            R.layout.a_child_row,
            new String[] { NAME },
            new int[] { R.id.menuexpandibleChildTextView }
            );
      setListAdapter(mAdapter);
  }
```

The CreateGroupList() function, contains of an object of List<Map<String, String>> and a menuItem object of CharSequenc[] array filled with its correspond defined elements in xml view. The CharSequence is an interface that represents an ordered set of characters and defines the methods to probe them.

```
private List createGroupList() {
List<Map<String, String>> groupList = new ArrayList<Map<String, String>>();

CharSequence[] menuItems = getResources().getTextArray(R.array.menuItems);

    for(CharSequence item : menuItems){
Map<String, String> groupMap = new HashMap<String, String>();
      groupMap.put(NAME, item.toString());
      groupList.add(groupMap);
    }
    return groupList;
  }
```

At least I had to mention that because of using Expandable List, we need to extend the main activity class to ExpandableListActivity as follow.

```
public class a_MainActivity extends ExpandableListActivity {}
```

The createGroupList() is defined by a list of grouplist that gets items for *menuItem* from /res/values/array.xml to returns the group list. The code snip of array.xml is placed here.

```
 <string-array name="menuItems">
    <item>WiFi</item>
    <item>Telephony</item>
  </string-array>
```

By applying these codes we will have the following screen shot. In addition, I have defined a Sub-List for both of options to achieve the figure 16.

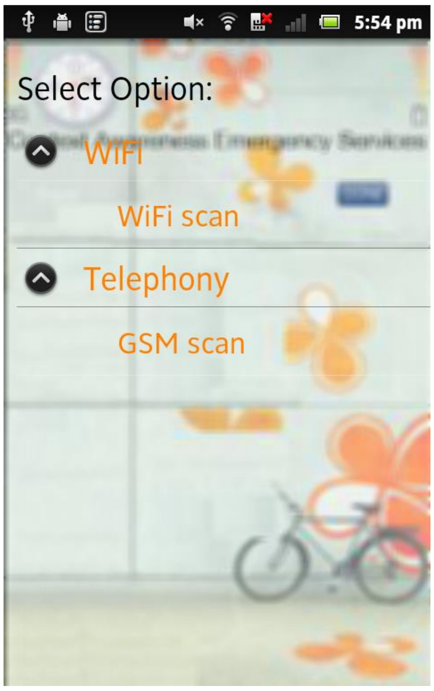
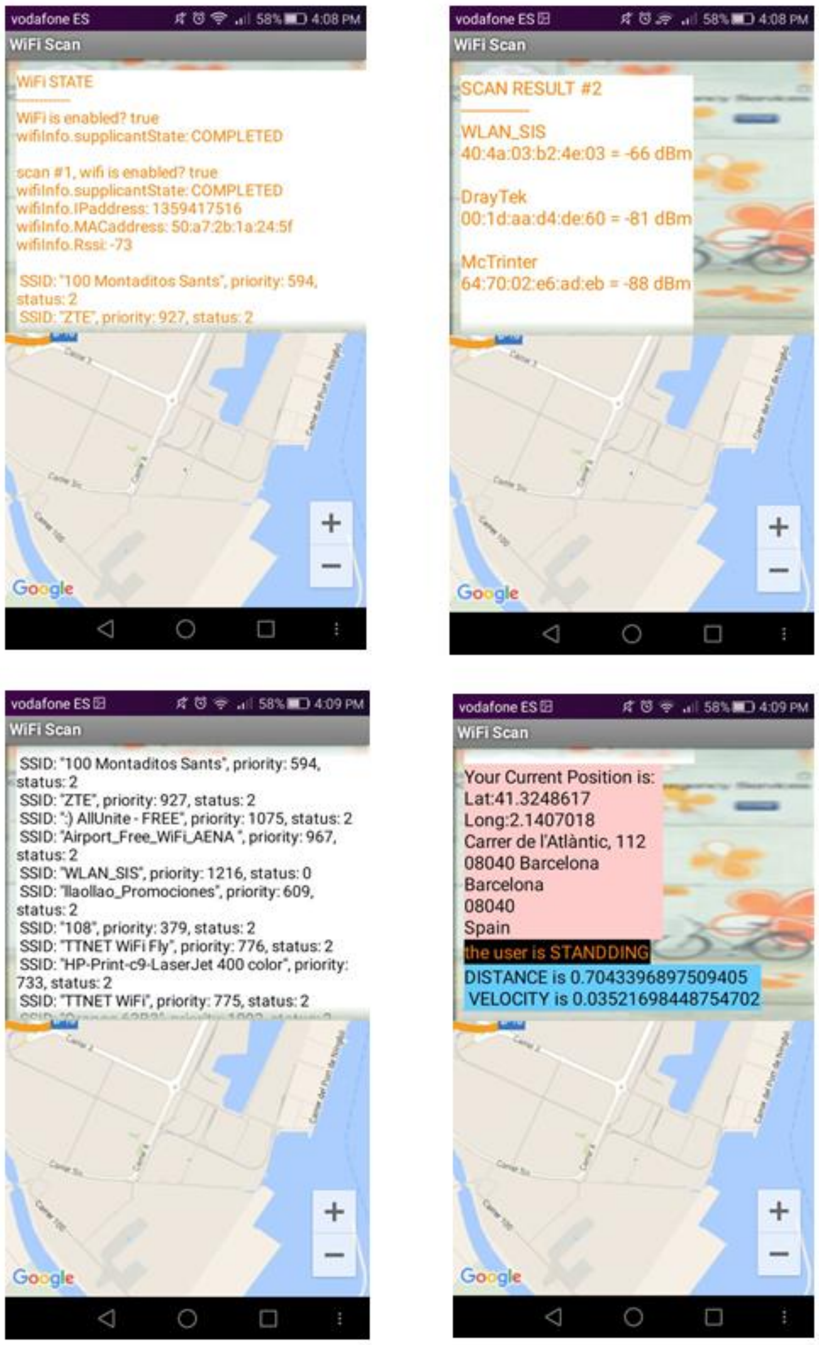


Figure 16: Second snapshot of a_ MainActivity

Part 2: c_wifiScanActivity

After doing click on Wi-Fi we will be navigated to this activity. In this part I explain the layout of this activity, and then its java class and other necessary components to implement this activity. The figure 17 presents whole the layout of c_wifiScanActivity.


WiFi Scan

WiFi STATE

WiFi is enabled? true
wifiInfo.supplicantState: COMPLETED

scan #1, wifi is enabled? true
wifiInfo.supplicantState: COMPLETED
wifiInfo.IPaddress: 1359417516
wifiInfo.MACaddress: 50:a7:2b:1a:24:5f
wifiInfo.Rssi: -73

SSID: "100 Montaditos Sants", priority: 594, status: 2
SSID: "ZTE", priority: 927, status: 2

WiFi Scan

SCAN RESULT #2

WLAN_SIS
40:4a:03:b2:4e:03 = -66 dBm

DrayTek
00:1d:aa:d4:de:60 = -81 dBm

McTrinter
64:70:02:e6:ad:eb = -88 dBm

WiFi Scan

SSID: "100 Montaditos Sants", priority: 594, status: 2
SSID: "ZTE", priority: 927, status: 2
SSID: ":) AllUnite - FREE", priority: 1075, status: 2
SSID: "Airport_Free_WiFi_AENA ", priority: 967, status: 2
SSID: "WLAN_SIS", priority: 1216, status: 0
SSID: "llaollao_Promociones", priority: 609, status: 2
SSID: "108", priority: 379, status: 2
SSID: "TTNET WiFi Fly", priority: 776, status: 2
SSID: "HP-Print-c9-LaserJet 400 color", priority: 733, status: 2
SSID: "TTNET WiFi", priority: 775, status: 2

WiFi Scan

Your Current Position is:
Lat:41.3248617
Long:2.1407018
Carrer de l'Atlàntic, 112
08040 Barcelona
Barcelona
08040
Spain
the user is STANDDING
DISTANCE is 0.7043396897509405
VELOCITY is 0.03521698448754702

Figure 17. the c_wifiScanActivity screen

I am going to create a new activity and build it based on below steps to develop all the elements that are shown in the figure 17.

- First defining its layout
- Importing services that we need to use
- Defining Wi-Fi information and ….

- Location detection
- Updating location
- Defining the user activity
- Refer to a database that reserve our desired data
- Importing Google map

The c_wifiScanActivity layout

As you can see in the figure 17 this activity is consist of five TextViews and a Google map and one Button that all the elements are embedded in a RelativeLayout view group with fill_parent property to covers all the screen of device with any kind of screen size. In this activity I have a quite bit of data which has to be shown in TextViews and Button and all of these widgets have to be fit in a half of the screen, so I needed to use ScrollView to wrap them in a part of screen and provide us a scroll bar to up and down. This view works like the window that by scrolling we will be able to have a larger screen more than space which device allows us to use.

Inside of ScrollView, a LinearLayout is solicited again to ask fill_parent properties for layout_height and layout_width and fill all the space that ScrollView can provide us.

To refer the Google map instance as a fragment class in the activity, I had to use fragment to set the inside of it in XML layout.

One notable property that is used in this layout is layout_weight equals to 1. This property is defined in ScrollView and fragment tags to imply weight for layout requirements and we are requesting half of the screen for ScrollView and half of the screen for the Google Map.

Another widget that is utilized in this layout is button. For button we have to use wrap_content for its layout_height and layout_width because it won't need more space to take. In following you can find the layout of this activity.

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

   android:layout_width="fill_parent"

   android:layout_height="fill_parent"

   android:orientation="vertical">

   <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"

         android:id="@+id/ wifiscan"

         android:layout_width="fill_parent"

         android:layout_height="0dip"

         android:layout_weight="1"

         android:background="@drawable/fons_blur">

      <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"

          android:layout_width="fill_parent"

          android:layout_height="fill_parent"

          android:orientation="vertical">

         <TextView

            android:id="@+id/GsmScanTextView1"

            android:layout_width="wrap_content"

            android:layout_height="wrap_content"

            android:layout_marginLeft="10dip"

            android:layout_marginTop="10dip"

            android:layout_marginBottom="10dip"

            android:textSize="12sp"

            android:textColor="@color/black"

            android:background="@color/white"

            android:text="To show Cell Info press button"/>
```

```
        <Button
          android:id="@+id/button1"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="SHOW CONNECTIONS"
        />
      </LinearLayout>
    </ScrollView>
    <fragment xmlns:android="http://schemas.android.com/apk/res/android"
       android:id="@+id/mapview"
       android:layout_width="match_parent"
       android:layout_height="0dp"
       android:layout_weight="1"
       class = "com.google.android.gms.maps.SupportMapFragment"/>
</LinearLayout>
```

Part 2: The c_wifiScanActivity Java Class

The layout of this activity is ready, so we can define its related java class, **c_wifiScanActivity**. This activity is a main building class like the previous activity so firstly it is needed to define its provided subclasses of android framework. So it starts with overriding onCreate() method to define elements which should be implemented. This activity extends FragmentActivity to support the android.support.v4.app library and provide some useful features and ensure us about compatibility of application with older versions of Android.

```
public class c_WifiScanActivity extends FragmentActivity {….. }
```

In this activity, I need to use Wi-Fi service to give me information about the Wi-Fi connection of device. So I should import its java class WifiManager by getSystemService () method, and import all the Wi-Fi services in a WifiManager object.

```
wifiManager = (WifiManager) getSystemService(Context.WIFI_SERVICE);
```

The forth step of this part is device's location detection while it is using Wi-Fi. In order to achieve this idea we need to apply location services of android and import them in a location manager that is able to manage all the information that is provided by location services. I will explain about it in following of this part.

Afterward as I have mentioned, the seventh step of this activity is to save all information that we requested in the database that we defined. In onCreate() method we have to pass the database through the onCreate(). This would be discussed in following parts completely. After passing the database, we have a button that should be defined in onCreate() method that is related to our database. So we need to set up the button to respond to clicks and connect us to the application's database activity that shows a table of data. This is a user interface object which has its own defined methods to do an action. The method that help us to execute certain code when the button is pushed is onClick() method has to pass it through the setOnClickListener method on the Button.

```
myButton = (Button)findViewById(R.id.button1);

        myButton.setOnClickListener(new View.OnClickListener() {

         public void onClick(View v) {

         Intent i=new Intent(c_WifiScanActivity.this, f_ConnectionList.class);

         startActivity(i);}});}
```

The second snip is onResume() method that runs while application in this activity is running, and this method is calling every time while the activity is brought up front. It retrieves information about the Wi-Fi states. It used a Wi-Fi timer to update searching about states of Wi-Fi.

In onResume() overriding method, I have defined a WifiInfo object that is an Android class that is able to retrieve the state of any active Wi-Fi or being set up connections by getConnectionInfo() method of WifiManager.

WifiManager of Wi-Fi Services can provide us some information as follow:

- isWifiEnabled(), a Boolean method to return if the Wi-Fi is enabled or not.
- setWifiEnabled(), to enable or disable Wi-Fi.

WifiInfo has following methods

- wifiinfo.getBSSID(),
- wifiinfo.getSSID(),
- wifiinfo.getIpAddress(), To obtain IP address.
- wifiinfo.getMacAddress(), To obtain MAC address.
- wifiinfo.getNetworkId(), Access network ID.
- **wifiinfo.getRssi(),** Access to RSSI, RSSI to receive signal strength indicator.
- **Wifiinfo.getSupplicantState(),** to return the detailed state of the supplicant negotiation with an access point in the form of a SupplicantState object.

```
public void onResume(){
        super. OnResume();
        WifiInfo wifiInfo = wifiManager.getConnectionInfo();
        stateTextView.setText("WiFi STATE\n------------\nWiFi is enabled?
"+wifiManager.isWifiEnabled()+"\nwifiInfo.supplicantState:"+wifiInfo.getSupp
licantState());
```

In the figure 17, is showing that we set in the TextView to show if the Wi-Fi is enabled? By calling isWifiEnabled() method of WifiManager and shows SupplicantState that is completely accessed to access point or has another enum values of SupplicantState such as ASSOCIATED, DISCONNECTED, SCANNING and so on.

In following I want the application to scan its Wi-Fi connectivity status and to get all of its information regularly in each 20 seconds. So as you can see in the snip code, I have registered a receiver in onResume() method that means, whenever the c_wifiScanActivity is running, recall the wifiScanReciever function that this function is defined separately as a private class that extends BroadcastReceiver. I will explain this function in the page 94 after introducing BroadcastReceiver. This BroadcastReceiver function needs to be registered in the Activity to affect the UI while it is updating. So the wifiScanReciever function should be registered in the onResume() handler and be unregistered in onPause() by IntentFilter class.

Broadcast Receivers are used to listen for Broadcast Intents. Now I need to explain about Filter and an instance of IntentFilter which it is used when we registered the wifiScanReciever by registerReceiver() method. This IntentFilter simply indicates which intent actions we want to be notified about. In this case, we want to run the wifiScanReciever through an IntentFilter when we have completed access to Access Point of Wi-Fi and we can have a result from supplier. So we add SCAN_RESULTS_AVAILABLE_ACTION method of WifiManager as an action to IntentFilter. We should keep this in mind that Receivers need to be registered in code or in the manifest file.

```
IntentFilter intentFilter = new IntentFilter();
intentFilter.addAction(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION);
registerReceiver(wifiScanReceiver, intentFilter);
wifiTimer = new Timer();
wifiTimer.schedule(new wifiTask(), 0, TIME_BETWEEN_SCANS);   }
```

In following snip code is showing onPause() handler that activates when the application is paused. In this part if the wifiTimer is not null, it stops listening and unregisters the wifiScanReceiver function and cancels the wifiTimer. Also if the

Boolean object of wifiWasNotEnabled be equals to false, the WifiManager service will be disabled.

```
public void onPause(){
      super.onPause();
      if(wifiTimer !=null){
         wifiTimer.cancel();
         unregisterReceiver(wifiScanReceiver);
      }
      if(wifiWasNotEnabled){
         wifiManager.setWifiEnabled(false);
         wifiWasNotEnabled = false;
      }
   }
```

Location Service

Before explaining about the code I'm going to explain about two methodologies of implementing location based services, first is a device-based application that user directly requests for finding the location for example using Google Map, and second is processing the location of user as a service in the application and deliver information to the user. The second methodology has been considered for the project.

Location Based Services of Android are accessible information services as long as the device is connected to the Internet. They can be used in different areas and conditions such as work, routine life and lots of other cases [13]. This is a real time service to find out the geographical position in spatial terms or text description. A spatial term means longitude, latitude and altitude of the user on the coordinate system.

The **latitude (φ)** is x-axis of user angle of device base on north or south from equatorial plane. With connection of all same width points together, a parallel line to the equator will be obtained and also all lines are parallel to each other. It varies between 0-90 degrees. The 0^o parallel line is considered the equator.

The **Longitude (λ)** is the west or east angle of the device based on meridian which is zero degree. And it varies from 0º – 180º to east or west. The zero point is located in the Gulf of Guinea and the antipodal meridian of Greenwich is both 180ºW and 180ºE. The composition of these longitude and latitude deduce the exact location of anything on the earth without considering its altitude that can present the height in meters above the sea. The text description represents string of data about address, street location, postal code, city and country.

To catch the current location is possible to use:

i) Mobile Phone Service Provider Network

By this service retrieve information about the current Base Transceiver Station (BTS) that the user is located in its area and is interacting with, which uses the radio base station of the connected station? The radios of a GSM cell could be from 2 – 20 kilometers. This service is cheap and doesn't add costs to the handset.

ii) Satellites (write and find better and more about, this is copy and paste)

The Global Positioning System (GPS) uses a constellation of 24 satellites orbiting the earth. GPS finds the user position by calculating differences in the times the signals, from different satellites, take to reach the receiver. GPS signals are decoded, so the smart phone must have in-built GPS receiver. This service provides us more accuracy of 5 – 10 meters. [8]

The Network Location Provider Service of Android defines the location of user by Wi-Fi Signals or cell tower information to determine that cell phone is indoor or outdoor.

As has been mentioned Location based services are important keys to find the user's current location and be updated periodically and retrieve their longitude, latitude and radius.

Now I can continue the project and explain how to imply location based services of Android. In the previous step application lets us to know about Wi-Fi connection, its status and all information that we willing to know. In this step I want to use location service the system service of Android. The location API of android is supported by the location manager.

After accessing to location services we need to register a location listener with location manager that location service would be able to call back when device's location has been changed. The Location Manager uses location providers, such as GPS or Network, to figure out our current location. In addition, to retrieve the most appropriate information about of accuracy, cost, and other requirements of providers, I have used Criteria.

I am going to give a short explanation about these four elements of location based services, Location manager, Location Listener, Location Provider and Criteria.

Location Manager

Location Based services of Android are services to be used for location detection. In order to establish a Location based Services system in a project, Location manager is the main class to utilize for communicating with other services and manage all the other components of LBS [13]. So we need to pass location services through a location manager by calling getSystemService () and register for location updates to use the location information. The major responsibility of this class is to implicate location-based services and retrieve periodic updates of geographical locations.

Location services should be passed through a location manager and implemented through onCreate() handler. So in following of the onCreate() in the previous part, we will add it for accessing to Location Services.

The other responsibilities of this class are:

- Obtain your current location
- Follow movement
- Set proximity alerts for detecting movement into and out of a specified area

- Find available Location Providers
- Monitor the status of the GPS receiver

The most useful method of Location manager are:

- **getAllProviders** : That provides a list of all known location providers around the cell phone.
- **getBestProvider**: It used Criteria class, to define the best provider between all providers.
- **getLastKnownLocation**: Retrieve data of the last known location from the given provider.
- **getProvider**: It presents information of the indicated location provider.
- **requstLocationUpdatses**: It updates the location, by using the provider in a specific time distance and a pending intent.

LocationListener

Also I am going to implement the LocationListener interface, which means adding a number of new callback methods to this activity. When the location changes, updates the location object, and next time around when we update our status online, we'll have the proper location information. It can receive notifications of the updates and changes of the location, which is used in the method of LocationManager, requstLocationUpdatses (String, long, float, LocationListener). This class includes of four methods to call by changes of location as below:

- onLocationChanged
- onProviderDisabled
- onProviderEnabled
- onStatusChanged

LocationProvider

It is an abstract superclass for location providers for periodic reports of geographical position of the devices. This class helps the application to get a list of available location providers on the device.

The methods used in the project related to LocationProvider are:

- **getAccuracy***()* to retrieve horizontal accuracy of the provider.
- **getName***()* for receive name of provider.

Criteria

This is the class help to choose the best and suitable Location Provider, and it prepares a set of required properties of the LocationProvider. It can provide accuracy, power stage, ability to report altitude, speed, bearing and monetary cost.

- ***setAccuracy***() :It represents the amount of accuracy of longitude and latitude that could be ACCURACY_FINE that desired location is fine, or ACCURACY_COARSE that has more accuracy with constant value of 2 (0x00000002) that is two times more than ACCURACY_FINE.
- ***setAltitudeRequired***(): It is a Boolean method that indicated if Altitude is needed or not by demanding setAltitudeRequired(false) or setAltitudeRequired(true).
- ***setBearingRequired***() : A Boolean method to indicate that if the bear information is needed or no.
- ***setCostAllowed***(): This is also another Boolean method to say if is allowed to incur monetary cost.
- ***setPowerRequirement***(): The desired level of power, that could be defined in three ways, ACCURACY_LOW, ACCURACY_HIGH or NO_REQUIERMENT. In the project POWER_LOW is used because of reducing the consumption of power and taking time.

Geocoder

A class to transform longitude and latitude coordinates of the location to return street address, postal code, city and country of the location. Is possible that for a location it be able to return all information about the location but for another location couldn't retrieve all information.

A Geocoder class is called in this order Geocoder (Context context, Locale locale). The locale is user to present language, country, and variant combination. It is used to notice about existence of information such as numbers, dates of a region.

After defining the Geocoder object, we try to prepare a list by getFromLocation (double latitude, double longitude, int maxResults) for obtaining an array of known addresses and getting the accurate or best guess information of location based on longitude and latitude. After filling the address list object with gc.getFromLocation(), we can retrieve address line, locality, postal code and the country name of location.

Finding Location Providers by implying Criteria

I have used Criteria class of Android to provide the best provider of the user based on our defining, setting requirements and choosing Contexts for using each one of its methods for a location provider in terms of accuracy, power and abilities of returning altitude, and speed. In the following snip code you can see what values are set for the provider.

Here I have a short definition for methods and the context that I have used to get the best location provider.

- **setAccuracy***()* that can be set by one of following accuracy contexts.
  - ACCURACY_FINE
  - ACCURACY_COARSE
- **setAltitudeRequired**() a Boolean method for defining if we need altitude or not.

- setBearingRequired()
- setCostAllowed()
- **setPowerRequirement***()* that can be set by one of three below contexts to set the power amount which is received by provider.
- ACCURACY_LOW
- ACCURACY_HIGH
- NO_REQUIERMENT
- POWER_LOW

```
public void onCreate(Bundle icicle) {
locationManager=
(LocationManager)getSystemService(Context.LOCATION_SERVICE);
        criteria = new Criteria();
        criteria.setAccuracy(Criteria.ACCURACY_FINE);
        criteria.setAltitudeRequired(false);
        criteria.setBearingRequired(false);
        criteria.setCostAllowed(true);
        criteria.setPowerRequirement(Criteria.POWER_LOW);
        String provider = locationManager.getBestProvider(criteria, true);
        location = locationManager.getLastKnownLocation(provider);
        locationManager.requestLocationUpdates(provider, 2000, 10,
locationListener);
}
```

As you can see in the snip code, the criteria are called by getBestProvider of LocationManager service to give us a name of criteria.

After finding the location of device and getting to know about the location provide based on our setting, we need to request from application to do this regularly. So we have to use requstLocationUpdates() method of location manager to register for location updates by getting, provider, minimum time interval between location updates, in milliseconds, minimum distance between location updates in meter and a pending intent of location listener.

Update Location by LocationListener

A LocationListener class is a callback method called by the location manager to return or do an action due to any changes or updates of location.

As I have mentioned, the location Listener is an important service for listening to the changes and sends or receives notifications when the location is changed, the provider is enabled or disabled and when the status is changes. These methods are named as onLocationChanged(), onProviderDisabled(), onProviderEnabled() and onStatusChanged().

```
private final LocationListener locationListener = new LocationListener() {
        public void onLocationChanged(Location location) {
          updateWithNewLocation(location);
        }
        public void onProviderDisabled(String provider){
          updateWithNewLocation(null);
        }
        public void onProviderEnabled(String provider) {}
        public void onStatusChanged(String provider, int status, Bundle extras){} };
```

The important method that is used by location listener in onLocationChanged() callback method, is *updateWithNewLocation(location)* method. That I can say that it is a milestone of my project, and by this method I have lots of progress. This method besides registering updates by location manager provides me current and previous location of device and so on, I was able to find distance of previous and current location and then Activity Recognition that all of them are explained in following.

UpdateWithNewLocation() method for Finding the Current Location and Previous Location of device

In the onCreate() override method we defined an object of location that gets the last known location by calling getLastKnownLocation method of location manager for retrieving the last location that device could distinguish and we should pass it through updateWithNewLocation(location). So the method gets the location object that is filled with all solicited information.

First of all function updateWithNewLocation(Location location) provides us the of longitude and latitude double amounts of the current location by two location methods of getLatitude() and getLongitude(). Then these values are set to show on a TextView on the screen.

```
public static double nlat;

public static double nlng;

nlat = location.getLatitude();

nlng = location.getLongitude();

latLongString = "Lat:" + nlat + "\nLong:" + nlng;

myLocationText.setText("Your Current Position is:\n" + latLongString);
```

Now we can continue the updateWithNewLocation method after extracting the latitude and longitude of the current location and having location’s string address to show it on myLocationText TextView element of the screen.

Then we have utilized GeoCoder class of location based services to show the street address, city and postal code of the current location of device because longitude and latitude are geographical values of location and user has no idea about them.

This class can transform longitude and latitude coordinates to real-world addresses.

Geocoder has two responsibilities, one is "Forward Coding" which detects the longitude and latitude of a partial address, and the other is reverse coding which acquires us the street address of device. If we don't specify a locale[1], it assumes the device's default, by the way I have indicated to return the default locale of user.

```
Geocoder gc = new Geocoder(this, Locale.getDefault());
```

Both geocoding functions return a list of Address objects. Each list can contain several possible results up to what is specified. To define a string street address, we have to use Reverse Geocoding and pass its getFromLocation method through a List<Address> with this format (double latitude, double longitude, int maxResults). This method will return a list of possible address matches to the specified properties. If the GeoCoder could not resolve any addresses for the specified coordinate, it returns null. For the result of this method we have an array of addresses that describes the area. Then we modified the updateWithNewLocation method to instantiate a list of addresses and passed in it the newly received location and limiting the results to a single address.

In an address object all the required information are defined by the following methods:

- getMaxAddressLineIndex(): To return the largest index currently used to show a address line.
- getAddressLine(): This method can specify a line of the address numbered by given index i.
- getLocality(): Defines locality of the address or null if is not defined.

1 - Locale represents a language/country/variant combination. Locales are used to alter the presentation of information such as numbers or dates to suit the conventions in the region they describe.

- getPostalCode
- getCountryName

After all of these processes we will have the exact location of the user in terms of geographical and real-world values. I have added the following snip code to the updateWithNewLocation() method.

```
private String updateWithNewLocation(Location location) {
        try {
              List<Address> addresses = gc.getFromLocation(nlat, nlng, 1);
              StringBuilder sb = new StringBuilder();
              if (addresses.size() > 0) {
                    Address address = addresses.get(0);
                    for (int i = 0; i < address.getMaxAddressLineIndex(); i++)
                    sb.append(address.getAddressLine(i)).append("\n");
                    sb.append(address.getLocality()).append("\n");
                    sb.append(address.getPostalCode()).append("\n");
                    sb.append(address.getCountryName());
              }
              addressString = sb.toString();
           } catch (IOException e) {}
      } else {
           latLongString = "No location found";
      }
      myLocationText.setText("Your Current Position is:\n" + latLongString + "\n"
+ addressString);
      return addressString;}
```

Before it is discussed about calculating the distance of user by retrieving longitude and latitude from its previous location and current location while is using internet. Although updateWithNewLocation() method provide us longitude and latitude of current location, it won't keep in its memory the previous location of user. So the solution for having both current and previous locations and their geographical values is to define two double values with name of plat and plng and at first define them with zero value.

While the activity is running and getting current user's location information, these two values are filled too. When the first location has value of location.getLatitude() for current location latitude as defined nlat and nlng is also equal to location.getLongitude(), plng and plat are amounted to zero. But when the location changes, we contextualize nlat and lng to nlat and nlng that are previous values of location before getting new location latitude and longitude to nlat and nlng. So we need to add them our updateWithNewLocation method.

In next part, I have explained about how the distance between two point is calculated.

Public double calculateDistance (double lat1, double lon1, double lat2, double lon2)

Before explaining the algorithm, I should explain about how the amounts of longitude and latitude that are based on degree, are converted to radians for these calculations. In following I explain about mathematical way of converting Degree to Radian and then the calculateDistance() method of the project.

The formula is used which calculates the distance between two points in a great circle, Earth is **Heversine** formula. It calculates the shortest distance of two points over the earth's surface. Is just about the distance based on sea surface and didn't consider the altitude.

Heversine formula:

- $a = \sin^2(\Delta\varphi/2) + \cos \varphi_1 \cdot \cos \varphi_2 \cdot \sin^2(\Delta\lambda/2)$
- $c = 2 \cdot \text{atan2}(\sqrt{\mathbf{a}}, \sqrt{\mathbf{1 - a}})$
- $d = R \cdot c$ - Distance of two points on the earth
- $\Delta\varphi$ – subtraction of latitude degrees in Radian
- $\Delta\lambda$ – subtraction of longitude degrees in Radian
- R – Radius of the Earth

The following snip code is illustrating how the Heversine formula is implied in the project.

```
double dLat = Math.toRadians(lat2 - lat1);

                double dLon = Math.toRadians(lon2 - lon1);

                double angle = Math.sin(dLat / 2) * Math.sin(dLat / 2) +
Math.cos(Math.toRadians(lat1)) * Math.cos(Math.toRadians(lat2)) *
Math.sin(dLon / 2) * Math.sin(dLon / 2);

                double c = 2 * Math.atan2(Math.sqrt(angle), Math.sqrt(1 - angle));

                double raio = 6378100;

                double distance = raio * c;
```

When we have distance of previous and current location of device, we can define the activity of user by its velocity. This is demonstrated in following part.

Activity Recognition

Now we can get to the aim that is activity recognition based on applying location based services. As is explained, the first step is to collect data from smartphone user by carrying and storing its locations in the memory of mobile device. Second step is to transfer data and use Heversine formula. Third step is the classification process where unknown activity template will be compared in a loop with every activity template from predefined training set. After calculating distances between each type of activity and target activity, algorithm will display as result that activity which will have minimal distance from the unknown one. Thus, the accuracy of results mostly depends on the templates that were chosen for selected activities. [7]

Based on what we have done to now, we have longitude and latitude of two different that maybe it doesn't change while user is standing. Then we calculate distance of user from its previous point, so by retrieving the distance of two points, it is possible to calculate the velocity of the user if we request it for periodic of 20 seconds. Afterward velocity of the user is can conclude the activity of user such as walking, running and driving which is the goal.

```
int timer = 20;
if ( velocity < 5 && velocity >= 1){
          myMotion.setText("the user is WALKING");
          myAction = "walking";
     }
     else if (velocity < 10 && velocity >= 5){
          myMotion.setText("the user is RUNNING");
          myAction = "running";
     }
     else if (velocity < 15 && velocity >= 10 ){
          myMotion.setText("the user is CYCLING");
          myAction = "riding";
     }
     else if (velocity >= 15){
          myMotion.setText("the user is DRIVING");
          myAction = "driving";}

     else if (velocity < 1 ){
          myMotion.setText("the user is STANDDING");
          myAction = "standing";
     }
```

And then the function calculateDistance(double lat2,double lon2,double lat1,double lon1) which need to get four entries is called in the *updateWithNewLocation()* function that already provides us the current and previous location's longitude and latitude. The following snip code shows how we have called it in the updateWithNewLocation().

```
calculateDistance(nlat,nlng,(plat==0?nlat:plat),(plng==0?nlng:plng));
```

In addition, the amount of velocity and distance are set in the myMotion and myDistanceAndVelocity TextViews of the layout.

myDrawLine() function

In following we had in mind to draw a line on a map to track the user's mobility. In order to draw a line on the map of application while user is using Internet specially when is using GPRS of its device, myDrawLine() is designated. It is a map based method that needs four double entries of previous and current longitude and latitude. So before implementing myDrawLine() method, we have to use external Maps library included as part of the Google API package, and create a map-based Activity using Google Maps as a user interface element. So as it is explained in the session 2, the map helps controlling display settings and use functions related to map to do an activity on the map.

Map views can be represented and set in the fragments inside of an activity to include Google map for application. The other reason of extending Activity to FragmentActivity is to include the map view element in the project. So we added this code where activity has is created.

```
public class c_WifiScanActivity extends FragmentActivity{ ...}
```

To run the Google map in the application, we developed a private function named setUpMapIfNeeded() for implementing the Google map inside of Map Fragment. So, first we have to check if the Google Play Services are available in the device or not. The GooglePlayServicesUtil is a package of classes that prepares us this information. This utility class declares the availability or being up-to-date of APK in the device.

The considered method of this class for our application is isGooglePlayServicesAvailable (Context context). The result of this method would equal to one the constants of ConnectionResult form com.google.android.gms.common.ConnectionResult class.

This class includes of all notifications about failing or connecting of device to Google play services. It uses GoogleApiClient.OnConnectionFailedListener to distinguish the errors or success in connection.

Contents of this class that we used to clarify are listed below:

- **SUCCESS** when the connection is achieved and breaks from code to counties setting of Google map.
- **SERVICE_MISSING** notification when device cannot find Google paly services.
- **SERVICE_VERSION_UPDATE_REQUIRED** notifies that we need to update the Google play services.
- **SERVICE_DISABLED** is notifying that we missed the Google Play services on the device.

And in case of three last notifications, the program will send a toast message for informing the user about Google play services connection fails. The below snip code shows how the application checks if the Google Play services is accessible or not.

```
private void setUpMapIfNeeded() {
        int resultCode = GooglePlayServicesUtil.isGooglePlayServicesAvailable(this);
        switch (resultCode) {
          case ConnectionResult.SUCCESS:
            break;
          case ConnectionResult.SERVICE_MISSING:
          case ConnectionResult.SERVICE_VERSION_UPDATE_REQUIRED:
          case ConnectionResult.SERVICE_DISABLED:
            Toast.makeText(getBaseContext(), "error with GooglePlayServices",
Toast.LENGTH_SHORT).show();
```

```
      }
```

In the case of success the function goes to next part and tries to obtain the map from the Support Map fragment. At first we check if the object of Google map is null which if do a null check, it means we don't have instantiated map, and then we fill it with a SupportMapFragment. The **getMap** () is the method to use for providing GoogleMap instance in the map view.

```
  GoogleMap mMap;

      if (mMap == null) {

        mMap = ((SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.mapview)).getMap();

      }

    }
```

After initiating the Google map inside of view, we can define myDrawLine() function to draw a line between previous and current points of the device each 20 second and retrieve a path while user is using Internet.

The tiles of the Google map object should be set by **setMapType**(int type) void method in order to determine which type of map should be initiated. There are three different types of map tile.

- **MAP_TYPE_NORMAL** that is a basic map with roads.
- **MAP_TYPE_SATELLITE** to display satellite view of the earth.
- **MAP_TYPE_TERRAIN** with Terrain view without roads.

Our desire type of Google map is the basic map with road so we used MAP_TYPE_NORMAL.

```
mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
```

For making the map more user friendly, we used UiStting from com.google.android.gms.maps.UiSettings class to set the user interface of GoogleMap by calling **getUiSetting**() method. It includes of some public methods to use in order to define the map. In following indicated methods that are used in the project:

- **setZoomControlsEnabled**, a Boolean value for enabling or disabling the zoom controls.
- **setMyLocationButtonEnabled** to enable or disable the my location button.
- **CameraUpdateFactory** class includes of some methods for setting camera, zoom updating in the map and modify the map's camera. It has some methods for updating camera like animateCamera(CameraUpdate), animateCamera(CameraUpdate, GoogleMap.CancelableCallback)or moveCamera(CameraUpdate).
- **MoveCamera(CameraUpdate)** based on the CancelCallback instructions repositions the camera's position and its zoom. So when we use newLatLngZoom(LatLng latLng, float zoom) method of CameraUpdateFactory, we are defining the zoom level and the center of the screen that is based on the current longitude and latitude.

```
public String myDrawLine(double lat1, double lng1, double lat2, double lng2){

        setUpMapIfNeeded();
        mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
        UiSettings uiSettings = mMap.getUiSettings();
        uiSettings.setZoomControlsEnabled(true);
        uiSettings.setMyLocationButtonEnabled(true);
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(new LatLng(lat1,
lng1), 13.0f));
```

In following, we have used **Polyline** class to draw a line between two points of the map. By the **PolylineOptions** method, the new polyline is drawn by the following properties:

- Draw the line between previous point and current point.
- Width of 4 pixels.
- Color of blue.
- Geodesic and Visibility are true.

```
    PolylineOptions po = new PolylineOptions()
                .add(new LatLng(lat1, lng1), new LatLng(lat2, lng2))
                .width(4)
                .color(Color.BLUE)
                .geodesic(true)
                .visible(true);
     po.add(new LatLng(lat2, lng2));


     Polyline polyline = mMap.addPolyline(po);
     return null;
}
```

Updating the Manifest File for Internet Permission

Before running the application, we should ask the user to grant us the right of using some elements and update our manifest file and request permission.

For this activity, we want to request some different grants in the manifest file. We need Internet access for connecting to online services, Google Play Services and Google map. So we update the Android Manifest.xml by adding following permission code:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Then because of using location based services of android, it needs its own permission. The methods to allow the application having access to location derived from network location sources such as Internet providers, cell and Wi-Fi and provide us accurate information are added in following term.

```
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />

<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Whereof we need permission for location services in manifest.xml to access for reading and writing the network and detect the internet connection and type of connection. As is mentioned in following:

```
<uses-permission
android:name=”android.permission.ACCESS_NETWORK_STATE”/>
```

This permission allows the API to check the connection status in order to determine whether or not data can be downloaded.

```
<uses-permission
android:name=”android.permission.CHANGE_NETWORK_STATE”/>
```

This permission allows the API to cache the map tile data in the device's external storage area.

For Google Map Android API V2 Permission, when the application runs, from manifest, Google Maps Android API v2 will read the key of application, and sends it to the servers of the Google Maps to receive confirmation of Google Maps data. This key should be added in the <application> element of manifest.

Allows the API to check the connection status in order to determine whether data can be downloaded that is added before for accessing to network.

```
<uses-permission android:name =
"android.permission.ACCESS_NETWORK_STATE">
```

After permissions we have to declare building blocks that we are developing them in them project in a <application> tag. Here we have ready an activity element that should be declare in a tag of <activity> and label it as we want, but for name we should write its .java name.

```
<application android:label="@string/app_name" >

      <meta-data android:name="com.google.android.gms.version"

        android:value="@integer/google_play_services_version"

      />

<activity android:name=".c_WifiScanActivity" android:label="WiFi Scan"

       android:screenOrientation="portrait"
android:windowSoftInputMode="stateHidden">

         <intent-filter>

            <action android:name="android.intent.action.MAIN"/>

            <category android:name="android.intent.category.DEFAULT"/>

         </intent-filter>

      </activity>

</application>
```

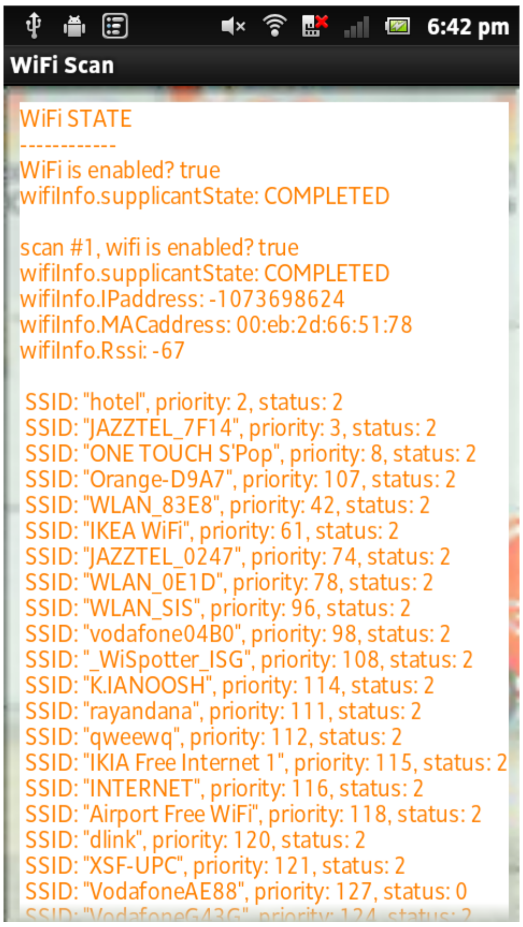
And at the end, for the Google Map we have to use a <meta-data> tag that is a name-value tag for additional data that we want to supply it to the parent component. For Google map we assign a value and name as unique name for this item. Name of this meta data for Google map is the package of API_KEY and for value we have to use the certificate key which Google provided us to use in the application.

```
<meta-data android:name="com.google.android.maps.v2.API_KEY"
```

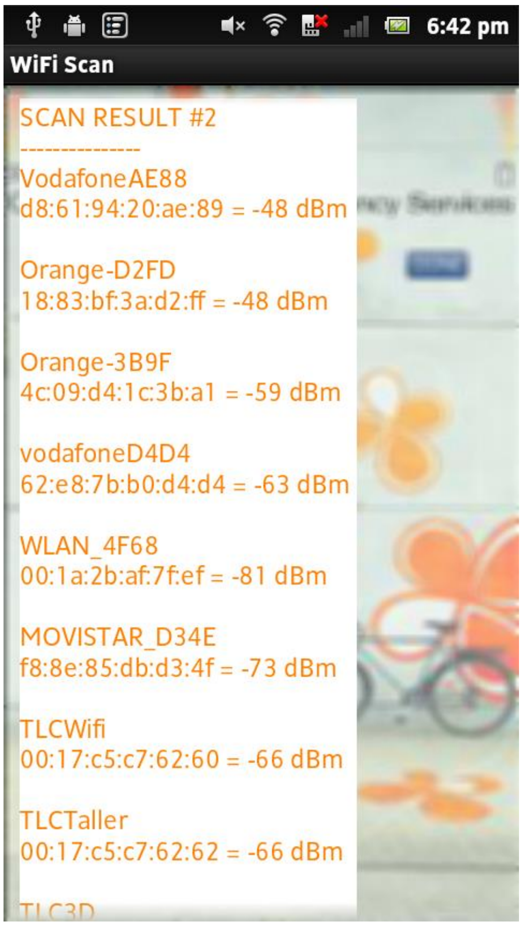
```
android:value="AIzaSyDiNg51xYmg_330JV5aSYxIAJ4rGK4CLb8"/>
```

Summary:

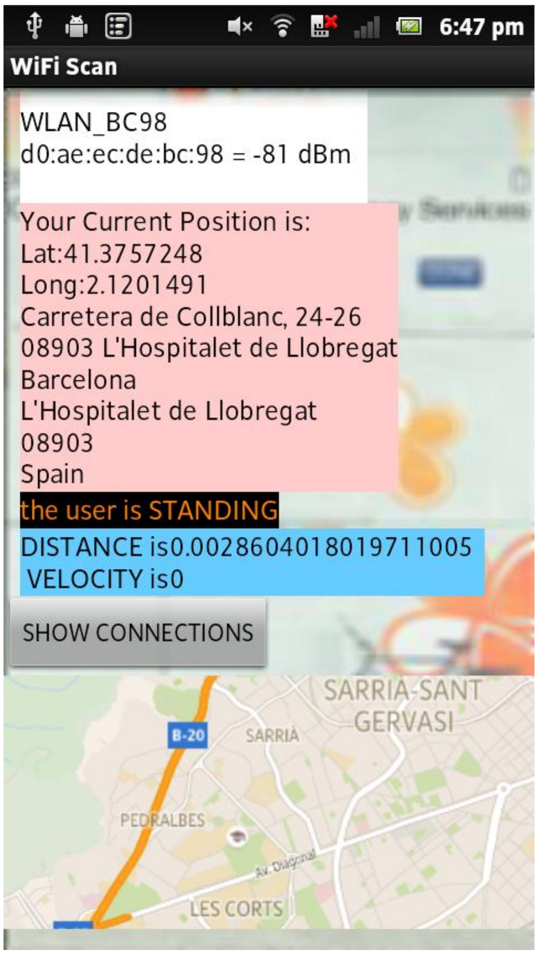
By the end of this part, when this layout of application is run, it looks like Figures 18 - 19. That as you can see it successfully can show the state of Wi-Fi which device is connected and shows the IPaddress, MACAddress and RSSI in first TextView and in second TextView it declares name of router which device is connected and its signal power of the Basic Service Set Identifier(BSSID) of recent Basic Service Sets.



```
WiFi STATE
------------
WiFi is enabled? true
wifiInfo.supplicantState: COMPLETED

scan #1, wifi is enabled? true
wifiInfo.supplicantState: COMPLETED
wifiInfo.IPaddress: -1073698624
wifiInfo.MACaddress: 00:eb:2d:66:51:78
wifiInfo.Rssi: -67

 SSID: "hotel", priority: 2, status: 2
 SSID: "JAZZTEL_7F14", priority: 3, status: 2
 SSID: "ONE TOUCH S'Pop", priority: 8, status: 2
 SSID: "Orange-D9A7", priority: 107, status: 2
 SSID: "WLAN_83E8", priority: 42, status: 2
 SSID: "IKEA WiFi", priority: 61, status: 2
 SSID: "JAZZTEL_0247", priority: 74, status: 2
 SSID: "WLAN_0E1D", priority: 78, status: 2
 SSID: "WLAN_SIS", priority: 96, status: 2
 SSID: "vodafone04B0", priority: 98, status: 2
 SSID: "_WiSpotter_ISG", priority: 108, status: 2
 SSID: "K.IANOOSH", priority: 114, status: 2
 SSID: "rayandana", priority: 111, status: 2
 SSID: "qweewq", priority: 112, status: 2
 SSID: "IKIA Free Internet 1", priority: 115, status: 2
 SSID: "INTERNET", priority: 116, status: 2
 SSID: "Airport Free WiFi", priority: 118, status: 2
 SSID: "dlink", priority: 120, status: 2
 SSID: "XSF-UPC", priority: 121, status: 2
 SSID: "VodafoneAE88", priority: 127, status: 0
 SSID: "VodafoneG43G", priority: 124, status: 2
```



```
SCAN RESULT #2
---------------
VodafoneAE88
d8:61:94:20:ae:89 = -48 dBm

Orange-D2FD
18:83:bf:3a:d2:ff = -48 dBm

Orange-3B9F
4c:09:d4:1c:3b:a1 = -59 dBm

vodafoneD4D4
62:e8:7b:b0:d4:d4 = -63 dBm

WLAN_4F68
00:1a:2b:af:7f:ef = -81 dBm

MOVISTAR_D34E
f8:8e:85:db:d3:4f = -73 dBm

TLCWifi
00:17:c5:c7:62:60 = -66 dBm

TLCTaller
00:17:c5:c7:62:62 = -66 dBm

TLC3D
```


6:47 pm

WiFi Scan

WLAN_BC98
d0:ae:ec:de:bc:98 = -81 dBm

Your Current Position is:
Lat:41.3757248
Long:2.1201491
Carretera de Collblanc, 24-26
08903 L'Hospitalet de Llobregat
Barcelona
L'Hospitalet de Llobregat
08903
Spain

the user is STANDING

DISTANCE is0.0028604018019711005
VELOCITY is0

Figure 18: First and Second Snapshot of c_wifiScanActivity

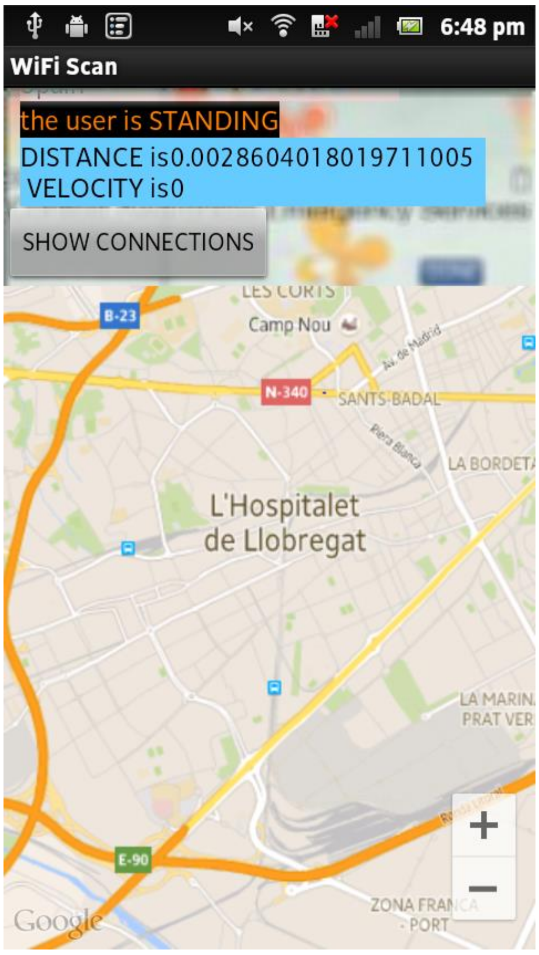

6:48 pm
WiFi Scan
the user is STANDING
DISTANCE is0.0028604018019711005
VELOCITY is0

Figure 19: Third and fourth Snapshot of c_wifiScanActivity

In the third TextView illustrates our Current Position in terms of Human readable and Geographical information latitude and longitude. Then in fourth TextView shows the User's Activity and at least TextView at the end Distance and velocity are displayed.

Show button is related to our database that I have explained about it in following and at the end, in the second part of activity we have Google map that shows our location and the movement of device which is showing by black color.

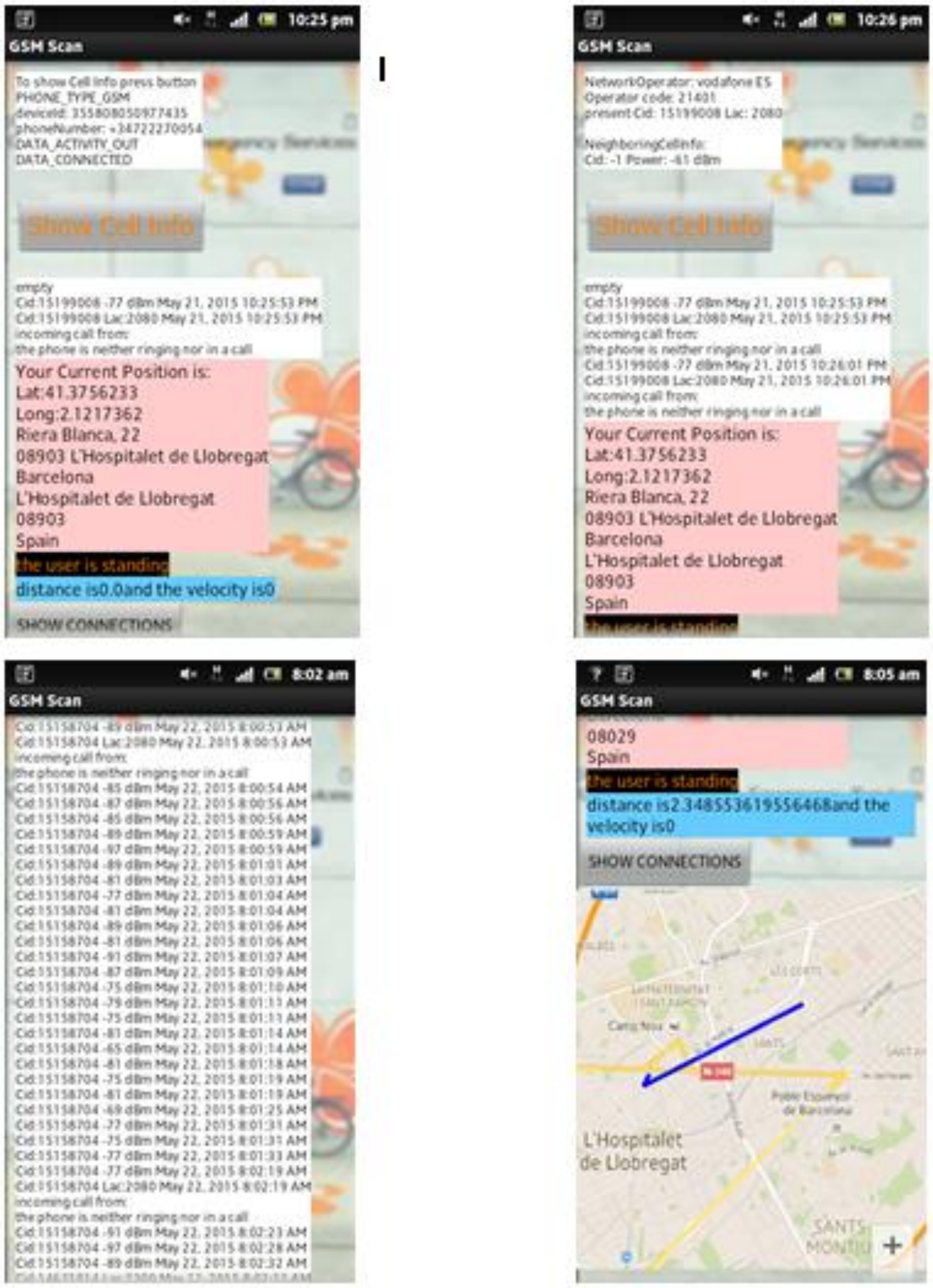
Part 3: d_GsmScanActivity

This activity starts to work by clicking on the Telephony and GSM scan of the list, we will be navigated to this activity. In this part I am going to skip explaining its layout because is similar to c_wifiScanActivity's layout. Also its java classes are similar to c_wifiScanActivity activity with some differences, so I will explain about the utilized additional methods and differences in the activity.

This activity also has similar steps as the previous one, and uses some extra libraries of android. In following I have mentioned the additional methods and libraries that are used in this activity.

- Importing additional services
- Show phone details() method
- MyPhoneStateListener() method

This activity has the whole layout as is presented in the figure 20.



```
To show Cell info press button
PHONE_TYPE_GSM
deviceId: 355808050977435
phoneNumber: +34722270054
DATA_ACTIVITY_OUT
DATA_CONNECTED

empty
Cid:15199008 -77 dBm May 21, 2015 10:25:53 PM
Cid:15199008 Lac:2080 May 21, 2015 10:25:53 PM
incoming call from:
the phone is neither ringing nor in a call
Your Current Position is:
Lat:41.3756233
Long:2.1217362
Riera Blanca, 22
08903 L'Hospitalet de Llobregat
Barcelona
L'Hospitalet de Llobregat
08903
Spain
the user is standing
distance is0.0and the velocity is0
```

```
NetworkOperator: vodafone ES
Operator code: 21401
present Cid: 15199008 Lac: 2080

NeighboringCellInfo:
Cid: -1 Power: -61 dBm

empty
Cid:15199008 -77 dBm May 21, 2015 10:25:53 PM
Cid:15199008 Lac:2080 May 21, 2015 10:25:53 PM
incoming call from:
the phone is neither ringing nor in a call
Cid:15199008 -77 dBm May 21, 2015 10:26:01 PM
Cid:15199008 Lac:2080 May 21, 2015 10:26:01 PM
incoming call from:
the phone is neither ringing nor in a call
Your Current Position is:
Lat:41.3756233
Long:2.1217362
Riera Blanca, 22
08903 L'Hospitalet de Llobregat
Barcelona
L'Hospitalet de Llobregat
08903
Spain
the user is standing
```

```
Cid:15158704 -89 dBm May 22, 2015 8:00:53 AM
Cid:15158704 Lac:2080 May 22, 2015 8:00:53 AM
incoming call from:
the phone is neither ringing nor in a call
Cid:15158704 -85 dBm May 22, 2015 8:00:54 AM
Cid:15158704 -87 dBm May 22, 2015 8:00:56 AM
Cid:15158704 -85 dBm May 22, 2015 8:00:56 AM
Cid:15158704 -89 dBm May 22, 2015 8:00:59 AM
Cid:15158704 -97 dBm May 22, 2015 8:00:59 AM
Cid:15158704 -89 dBm May 22, 2015 8:01:01 AM
Cid:15158704 -81 dBm May 22, 2015 8:01:03 AM
Cid:15158704 -77 dBm May 22, 2015 8:01:04 AM
Cid:15158704 -81 dBm May 22, 2015 8:01:04 AM
Cid:15158704 -89 dBm May 22, 2015 8:01:06 AM
Cid:15158704 -81 dBm May 22, 2015 8:01:06 AM
Cid:15158704 -91 dBm May 22, 2015 8:01:07 AM
Cid:15158704 -87 dBm May 22, 2015 8:01:09 AM
Cid:15158704 -75 dBm May 22, 2015 8:01:10 AM
Cid:15158704 -79 dBm May 22, 2015 8:01:11 AM
Cid:15158704 -75 dBm May 22, 2015 8:01:11 AM
Cid:15158704 -81 dBm May 22, 2015 8:01:14 AM
Cid:15158704 -65 dBm May 22, 2015 8:01:14 AM
Cid:15158704 -81 dBm May 22, 2015 8:01:18 AM
Cid:15158704 -75 dBm May 22, 2015 8:01:19 AM
Cid:15158704 -81 dBm May 22, 2015 8:01:19 AM
Cid:15158704 -69 dBm May 22, 2015 8:01:25 AM
Cid:15158704 -77 dBm May 22, 2015 8:01:31 AM
Cid:15158704 -75 dBm May 22, 2015 8:01:31 AM
Cid:15158704 -77 dBm May 22, 2015 8:01:33 AM
Cid:15158704 -77 dBm May 22, 2015 8:02:19 AM
Cid:15158704 Lac:2080 May 22, 2015 8:02:19 AM
incoming call from:
the phone is neither ringing nor in a call
Cid:15158704 -91 dBm May 22, 2015 8:02:23 AM
Cid:15158704 -97 dBm May 22, 2015 8:02:28 AM
Cid:15158704 -89 dBm May 22, 2015 8:02:32 AM
```

```
08029
Spain
the user is standing
distance is2.348553619556468and the
velocity is0
```

Figure 20: a screen shot of d_GSMScanActivity

The d_ GSMScanActivity Java Class

This activity is started by onCreate() callback method to create all the widget, and Google map of the screen as you can see in the figure 20. In this activity also we will to find information about the cell that device is connected to network. So here one of the differences among these two activities has been appeared, because we don't need to imply Wi-Fi services to get information of location.

The service that we are going to use is TELEPHONY_SERVICE that has been mentioned in previous session. In this activity, we will to retrieve information about the cell that device is currently using its services, so in order to access to the telephony APIs we have to request TELEPHONY_SERVICE context of android and pass it through a Telephony Manager for enabling us to use elements and methods of this service and handling management features of the device. We have used telephony manager to provide us many properties of telephone, the device information, network information and data state details. The applied methods of telephony manager are listed below:

- **getNetworkOperatorName**(): returns the alphabetic name of current registered operator.
- **getNetworkOperator**: returns the numeric alphabetic name of current registered operator.
- **getCellLocation**(): Returns the current location of the device.
- **getNeighboringCellInfo**(): Returns the neighboring cell information of the device.
- **getPhoneType**():Returns a constant indicating the device phone type. This indicates the type of radio used to transmit voice calls. The phone types that are predictable by this method are GSM, CDMA, SIP and none if it doesn't know the type.
- **getDataActivity**(): Returns a constant indicating the type of activity on a data connection (cellular). The constants that can represent are activity in, activity out, activity in-out and dormant.

- **getDataSet**():returns a constant indicating the current data connection state.

So after passing TELEPHONY SERVICES through a telephony manager instance, we are able to have above information. This should be created in onCreate() callback method of this activity. So we have updated in the activity by following snip code:

```
TelephonyManager  telephonyManager =
(TelephonyManager)getSystemService(Context.TELEPHONY_SERVICE);
```

Now we need to instantiate an instance for recognition of device connectivity to network. In order to retrieve this information, we need to use one of the Android's Services with the name of CONNECTIVITY_SERVICE.

In session 2, CONNECTIVITY_SERVICE is also mentioned that this Context is the service of handling management of network connections. Like the other services, also we need to utilize a manager in order to keep all information of the service inside of itself. So an object to keep values of network has to be defined like following:

```
(ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE)
```

onResume()

The active lifetime of application starts with onResume() callback method to receive events. We want to keep the application responsive when the foreground is moving in and out. This method should be light weight and can be used for registering a broadcast receiver or a process that need to be stopped in onPause().

In this override method, I want the activity listens to any changes of the telephony states and receive notifications. For this reason, we need to register a listener object and pass a PhoneStateListener through it and define which contexts of PhoneStateListener are in mind to receive its events. So we have used listen () method of telephony manager and beside of passing myPhoneStateListener as an object of MyPhoneStateListener() through it, we requested to for monitoring the following events.

- LISTEN_SIGNAL_STRENGTH
- LISTEN_CELL_LOCATION
- LISTEN_CALL_STATE

The following snip code shows the way of applying this method.

```
protected void onResume() {

      super.onResume();

      telephonyManager.listen(myPhoneStateListener,

         PhoneStateListener.LISTEN_SIGNAL_STRENGTH |

         PhoneStateListener.LISTEN_CELL_LOCATION |

         PhoneStateListener.LISTEN_CALL_STATE);

   }
```

While the application is listening to events and receiving notifications, it updates information of the cell and shows them in the second TextView.

onPause()

Now we have to end the listening to events which we called in onResume() by onPause () method and unregister it. So the telephony manager should stop listening updates and notifications by unregistering listener and LISTEN_NONE event pass through listen() method.

```
   protected void onPause() {
      super.onPause();
      telephonyManager.listen(myPhoneStateListener,
PhoneStateListener.LISTEN_NONE);
   }
```

In this activity, I have defined two additional methods to work with telephony manager. They named as show_phone_details() and MyPhoneStateListener() that in following of this part, I am going to briefly outline them.

show_phone_details ()

This method focused on demonstrating information about the cell by utilizing telephony manager as soon as this activity is run, so this method should be called in onCreate() overriding method to show us basic information of cell in the first TextView of screen as you can see in the first figure of figure xxxx. It uses getPhoneType() method of telephony manager to retrieve type of phone such as SIP, CDMA and GSM.

And then it shows the device ID by getDeviceId() method and get number of cart by getLine1Number() method of telephony manager. And at least getDataSet() method to returns a constant indicating the current data connection state.

```
private void show_phone_details(){
    int phoneType = telephonyManager.getPhoneType();
    switch (phoneType) {
      case (TelephonyManager.PHONE_TYPE_CDMA):
        TextView1.append("\nPHONE_TYPE_CDMA");
         …
    }
    String deviceId = telephonyManager.getDeviceId();
    String phoneNumber = telephonyManager.getLine1Number();
    int dataActivity = telephonyManager.getDataActivity();
    int dataState = telephonyManager.getDataState();
    switch (dataActivity) {
      case TelephonyManager.DATA_ACTIVITY_IN:
        …..
    }
    switch (dataState) {
      case TelephonyManager.DATA_CONNECTED:
        TextView1.append("\nDATA_CONNECTED");
…
    } }
```

At the bottom of this TextView, we have a button that if user clicks on this button, content of TextView will be changed and show us other requested information of cell as you can see in the second shape of figure 21. Then again, the TextView will be filled by new contents, such as name of network operator that is the result of using getNetworkOperatorName() method of telephonyManager, and then some additional information such as ID of network operator.

In addition by retrieving location of user, it calculates GSM the signal strength. To calculate the signal strength, it needs to retrieves received signal strength indicator (RSSI) of a cell by getRssi() method of NeighboringCellInfo.

Received Signal Strength Indicator (RSSI) is a definition in telecommunication for power measurement based on received Radio Signals of IEEE 802.11 systems. The signal strength provided by getRssi() method is "asu" that it is undefined for GSM, so we need standard calculation of dBm = -113 + 2*asu in order to have a signal strength based on dBm for GSM.

```
public void onClick(View arg0) {
if(arg0==cellInfoButton){
        TextView1.setText("NetworkOperator: "+telephonyManager.getNetworkOperatorName()+
                "\nOperator code: "+telephonyManager.getNetworkOperator());
        GsmCellLocation location = (GsmCellLocation) telephonyManager.getCellLocation();
        if(location!=null)
```

```
            TextView1.append("\npresent Cid: "+location.getCid()+" Lac:
"+location.getLac()                              );

         TextView1.append("\n\nNeighboringCellInfo:");

         List<NeighboringCellInfo> llista =
telephonyManager.getNeighboringCellInfo();

         for(NeighboringCellInfo cellInfo : llista){

            int power = -113 + 2 * cellInfo.getRssi();

            TextView1.append("\nCid: "+cellInfo.getCid()+" Power: "+power+"
dBm");

         }          }}
```

MyPhoneStateListener()

The private class of MyPhoneStateListener is extended from PhoneStateListener for monitoring and listening to changes of telephony states such as service state, signal strength, message waiting indicator and others. Listening to the telephony information needs a permission that should to be requested by manifest file of application. By the flags defined in this listener we can retrieve our desired information about any cell.

The considered method to use by this listener is onSignalStrengthChanged(int asu) that callbacks when the strength of the signal changes. So after retrieving gsm location of device it is able to provide us the cell ID, and calculate the power in dBm for GSM connections.

```
private class MyPhoneStateListener extends PhoneStateListener {

      private ServiceState latestServiceState;

      private boolean stop;

public void onSignalStrengthChanged(int asu){

         int power = -113 + 2 * asu;

         GsmCellLocation location = (GsmCellLocation)
telephonyManager.getCellLocation();

            Calendar calendar = Calendar.getInstance();

            if (location!=null)

                TextView2.append("\nCid:"+location.getCid()+" "+power+" dBm
"+calendar.getTime().toLocaleString());

            else

                TextView2.append("\nCid:"+" unknown"+" "+power+" dBm
"+calendar.getTime().toLocaleString());

        }

        public void stopMeasuring() {

stop = true;

}
```

In following, onCellLocationChanged (CellLocation location) is developed in the listener in order to get the new Cell ID when the location of cell is changed and appends on TextView of application Cell ID and GSM Location Area code by getLac() method.

```
@Override
public void  onCellLocationChanged(CellLocation location){
GsmCellLocation gsmCell = (GsmCellLocation) location;
Calendar calendar = Calendar.getInstance();
     if(gsmCell.getCid()==-1)
                return;
TextView2.append("\nCid:"+gsmCell.getCid()+" Lac:"+gsmCell.getLac()+"
"+calendar.getTime().toLocaleString());
        }
```

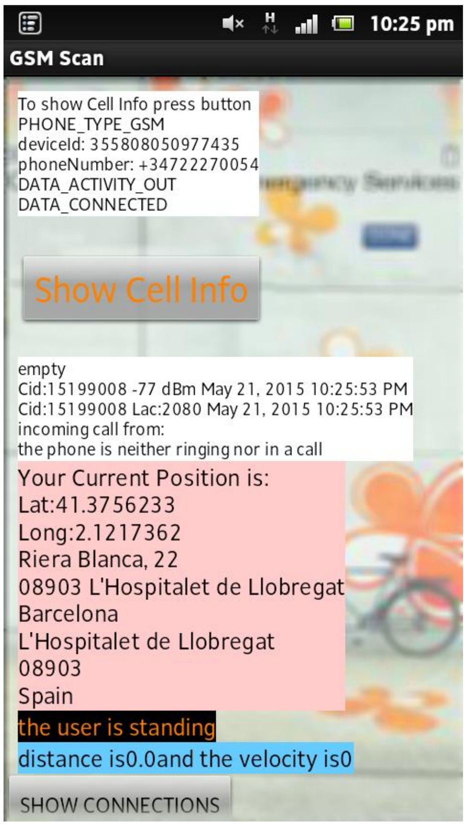
Summary:

The result of this activity looks like Figure 21 - 24. That as you can see it successfully can show the basic information of device such as type of phone, device ID and telephone number and its connection to GSM in first TextView, then by pressing the Show Cell Info button, its information changes to show information of network connection such as name, code and ID of network operator and signal strength of its neighbor cells.

Then when you scroll down, in second TextView it declares the location signal power and date.

In the third TextView illustrates our Current Position in terms of Human readable and Geographical information latitude and longitude. Then in fourth TextView shows the User's Activity and at least TextView at the end Distance and velocity are displayed.

Show button is related to our database that I have explained about it in following and at the end, in the second part of activity we have Google map that shows our location and the movement of device which is showing by black color.



To show Cell Info press button
PHONE_TYPE_GSM
deviceId: 355808050977435
phoneNumber: +34722270054
DATA_ACTIVITY_OUT
DATA_CONNECTED

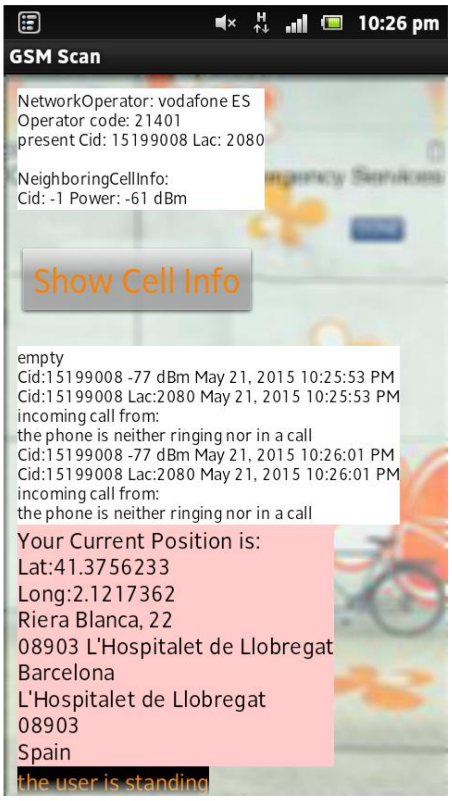
empty
Cid:15199008 -77 dBm May 21, 2015 10:25:53 PM
Cid:15199008 Lac:2080 May 21, 2015 10:25:53 PM
incoming call from:
the phone is neither ringing nor in a call

Your Current Position is:
Lat:41.3756233
Long:2.1217362
Riera Blanca, 22
08903 L'Hospitalet de Llobregat
Barcelona
L'Hospitalet de Llobregat
08903
Spain

the user is standing

distance is0.0and the velocity is0

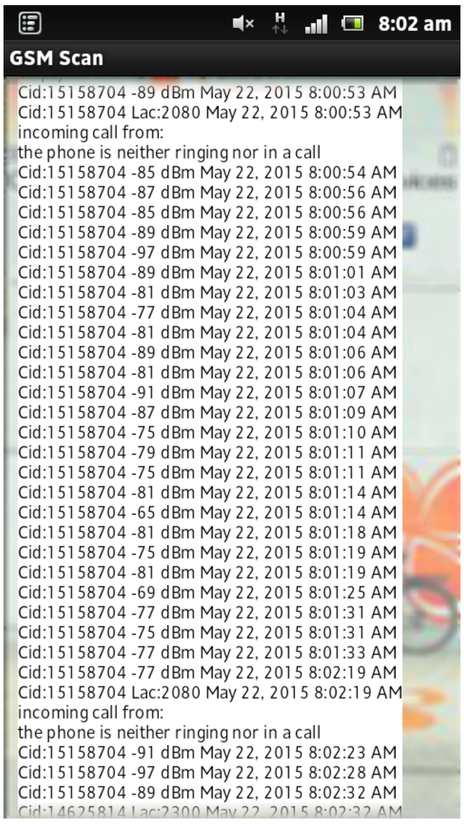
Figure 21: first snapshot of d_GsmScanActivity


10:26 pm
GSM Scan

```
NetworkOperator: vodafone ES
Operator code: 21401
present Cid: 15199008 Lac: 2080

NeighboringCellInfo:
Cid: -1 Power: -61 dBm
```

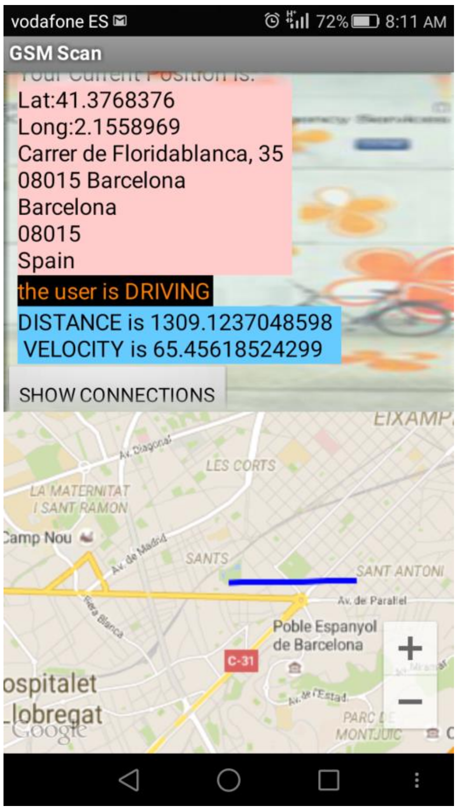
```
empty
Cid:15199008 -77 dBm May 21, 2015 10:25:53 PM
Cid:15199008 Lac:2080 May 21, 2015 10:25:53 PM
incoming call from:
the phone is neither ringing nor in a call
Cid:15199008 -77 dBm May 21, 2015 10:26:01 PM
Cid:15199008 Lac:2080 May 21, 2015 10:26:01 PM
incoming call from:
the phone is neither ringing nor in a call
Your Current Position is:
Lat:41.3756233
Long:2.1217362
Riera Blanca, 22
08903 L'Hospitalet de Llobregat
Barcelona
L'Hospitalet de Llobregat
08903
Spain
the user is standing
```

Figure 22: *First snapshot of d_GsmScanActivity after pressing Show Cell info Button*


8:02 am
GSM Scan

```
Cid:15158704 -89 dBm May 22, 2015 8:00:53 AM
Cid:15158704 Lac:2080 May 22, 2015 8:00:53 AM
incoming call from:
the phone is neither ringing nor in a call
Cid:15158704 -85 dBm May 22, 2015 8:00:54 AM
Cid:15158704 -87 dBm May 22, 2015 8:00:56 AM
Cid:15158704 -85 dBm May 22, 2015 8:00:56 AM
Cid:15158704 -89 dBm May 22, 2015 8:00:59 AM
Cid:15158704 -97 dBm May 22, 2015 8:00:59 AM
Cid:15158704 -89 dBm May 22, 2015 8:01:01 AM
Cid:15158704 -81 dBm May 22, 2015 8:01:03 AM
Cid:15158704 -77 dBm May 22, 2015 8:01:04 AM
Cid:15158704 -81 dBm May 22, 2015 8:01:04 AM
Cid:15158704 -89 dBm May 22, 2015 8:01:06 AM
Cid:15158704 -81 dBm May 22, 2015 8:01:06 AM
Cid:15158704 -91 dBm May 22, 2015 8:01:07 AM
Cid:15158704 -87 dBm May 22, 2015 8:01:09 AM
Cid:15158704 -75 dBm May 22, 2015 8:01:10 AM
Cid:15158704 -79 dBm May 22, 2015 8:01:11 AM
Cid:15158704 -75 dBm May 22, 2015 8:01:11 AM
Cid:15158704 -81 dBm May 22, 2015 8:01:14 AM
Cid:15158704 -65 dBm May 22, 2015 8:01:14 AM
Cid:15158704 -81 dBm May 22, 2015 8:01:18 AM
Cid:15158704 -75 dBm May 22, 2015 8:01:19 AM
Cid:15158704 -81 dBm May 22, 2015 8:01:19 AM
Cid:15158704 -69 dBm May 22, 2015 8:01:25 AM
Cid:15158704 -77 dBm May 22, 2015 8:01:31 AM
Cid:15158704 -75 dBm May 22, 2015 8:01:31 AM
Cid:15158704 -77 dBm May 22, 2015 8:01:33 AM
Cid:15158704 -77 dBm May 22, 2015 8:02:19 AM
Cid:15158704 Lac:2080 May 22, 2015 8:02:19 AM
incoming call from:
the phone is neither ringing nor in a call
Cid:15158704 -91 dBm May 22, 2015 8:02:23 AM
Cid:15158704 -97 dBm May 22, 2015 8:02:28 AM
Cid:15158704 -89 dBm May 22, 2015 8:02:32 AM
Cid:14625814 Lac:2300 May 22, 2015 8:02:32 AM
```

Figure 23: *Second snapshot of d_GsmScanActivity to show all the GSM connections*



vodafone ES 72% 8:11 AM

GSM Scan

Your Current Position is:
Lat:41.3768376
Long:2.1558969
Carrer de Floridablanca, 35
08015 Barcelona
Barcelona
08015
Spain

the user is DRIVING

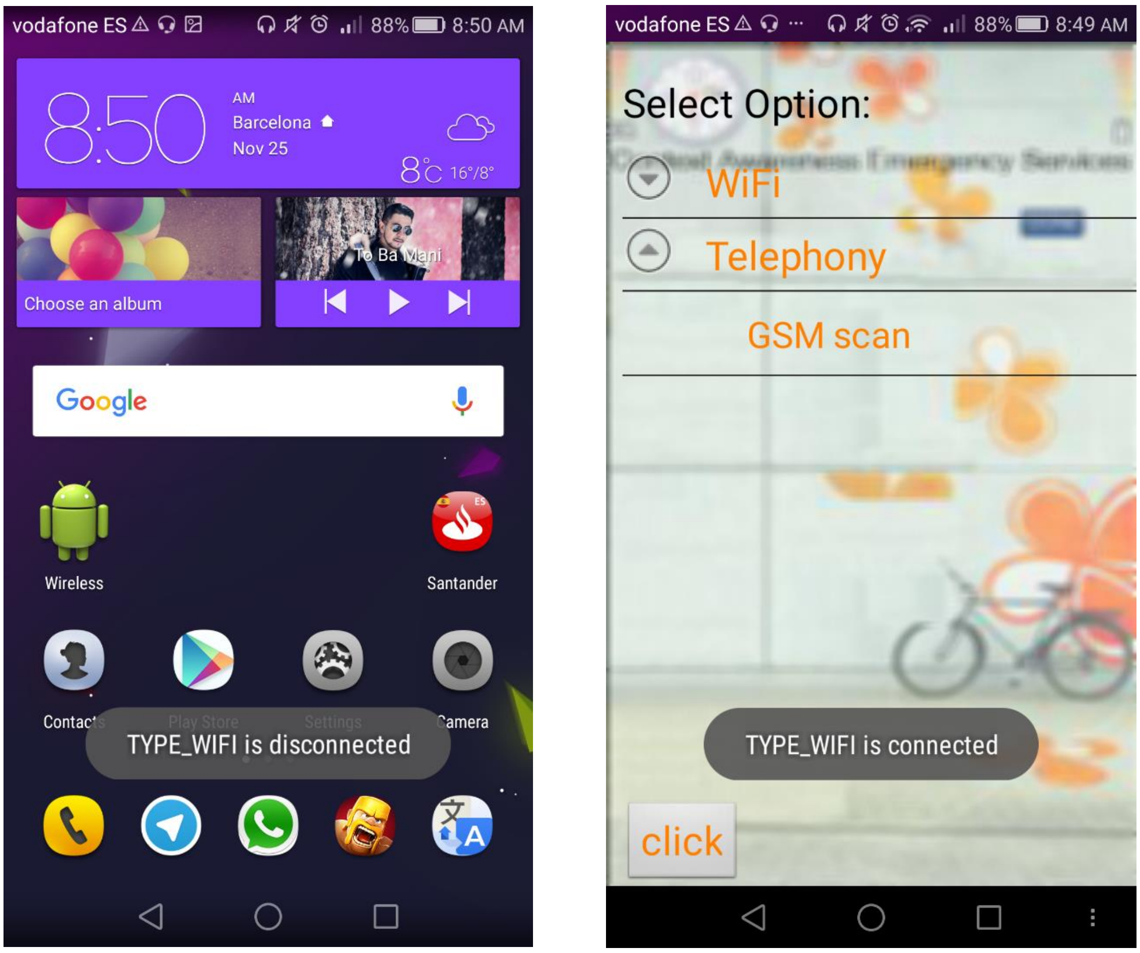
DISTANCE is 1309.1237048598
VELOCITY is 65.45618524299

SHOW CONNECTIONS

Figure 24: Forth snapshot of d_GsmScanActivity

Part 4: Broadcast Receiver

In this part, I want to develop broadcast receivers to alert application the connection type of device. The first scenario of this Broadcast receiver is when the device is powered up and working, it starts to detect and monitor the Network connection and send a toast message. And the second scenario is to save all types of connection after checking, by a method that is defined in this Broadcast Receiver. First we have an overview on how broadcast receivers work.



TYPE_WIFI is disconnected

TYPE_WIFI is connected

Figure 25 – The broadcast Toast message

About Broadcast Receivers

As I have explained in session 2, Broadcast receivers are one of the building blocks of Android for implementation of the Publish/Subscribe messaging pattern, or more precisely, the Observer pattern. We are using this element for working in background without taking in account that if anyone will get them to send events. So BroadcastReceiver is an element of application in order to get notified and send notifications when an action happens. This action is a type of intent broadcast that when it happens the receiver wakes up and does something. The onReciver () is its callback method that do something when application receives an event.

_connectivityReceiver

In the application the broadcast receiver is responsible for monitoring network connectivity and announce about the internet connection of mobile based on any modification in the Internet connection, and alert with four different sentences by any changes.

This broadcast receiver is divided into main two parts

- OnReciver callback Method
- Log method

As we want the broadcast starts to monitor the system automatically while the device is working, we create _connectivityReceiver. So we Create connectivityReceiver by extending it from its base class BroadcastReceiver.

```
public class _connectivityReceiver extends BroadcastReceiver {
  public void onReceive(Context context, Intent intent) {}
  public void log(Context context, String sentence){}
}
```

onReceive() callback Method

As you can see it has two methods inside of the BroadcastReceiver. So we start with overriding onReceive (Context context, Intent intent) callback method. This method gets calls when intent matches this receiver. For my application, first it tries to figure out if we are connected to Wi-Fi or disconnected from Wi-Fi, also if we are connected to mobile/GPRS or disconnected. By any changes it the connection it will send a toast message to inform user about connections.

To define the type of connection, first should recognize if the device is connected to Internet or not, so it is necessary to use two service managers of android with name of WifiManager and ConnectivityManager. There are services for Wi-Fi or GPRS connectivity, monitoring and modifying the network settings and accessing to the point scans. [1]

In this case, Connectivity Service provides us a high level view on monitoring and availability of network. By utilizing its manager, besides of providing queries about the network connectivity states can send a notification in order to changes in the connectivity type or state.

To notify us when any event happens, the broadcast receiver should pass through Intent for listening to connectivity of device and any changes of connection therefore does a predefined action. It is done by ConnectivityManager. CONNECTIVITY_ACTION broadcast as is shown in the following manifest snip code. The Intent contains extras to provide us additional information about changes in connectivity.

```
<intent-filter>
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
 </intent-filter>
```

After that, in order to describe the status of network interface and see what kind of connectivity event occurred, the NetworkInfo class is utilized.

The NetworkInfo class is used to find the connection status, network type, and detail the state information of the returned network. We can define status of the network by a NetworkInfo object that it has responsibility of describing the status of a network interface. Its methods to use are as below:

- getTypeName(): Which will return a human-readable name which clears is WIFI or MOBILE.
- getState(): to return the current coarse-grained state of the network.
- isAvailable()
- isConnected()
- getSubTypeName(): a human-readable name of subtype of network.

The following snip code shows the onReciver() method that is portioned into four parts to check the connection and its type. Before that I have to mention that getParcelableExtra() method extends data from intent and gives us extra information such as reason of failing and other information of the mentioned network.

```
NetworkInfo networkInfo =
intent.getParcelableExtra(ConnectivityManager.EXTRA_NETWORK_INFO);
if (networkInfo.getType() == ConnectivityManager.TYPE_WIFI
&& !networkInfo.isConnected()) {}
if (networkInfo.getType() == ConnectivityManager.TYPE_WIFI
      && networkInfo.isConnected()) {}
if (networkInfo.getType() == ConnectivityManager.TYPE_WIFI
      && networkInfo.isConnected()) {}
if (networkInfo.getType() == ConnectivityManager.TYPE_MOBILE
&& !networkInfo.isConnected()) {
...}
```

By distinguishing any kind of these conditions, The onReciver() method will toast a message, then will retrieve date of changes and save them in history of device by the method defined in the second part of this building block.

```
if (networkInfo.getType() == ConnectivityManager.TYPE_MOBILE
          && !networkInfo.isConnected()) {
          // TYPE_MOBILE is disconnected
          Toast.makeText(context, "TYPE_MOBILE is disconnected",
Toast.LENGTH_LONG).show();
```

```
            Date date = new Date();
            log(context, "TYPE_MOBILE is disconnected at: "+ date.toLocaleString()
                  + "\n NetworkInfo:" + String.valueOf(networkInfo));
          }
```

Log

The log class is used in order to send an output of logs. The point of developing the log class is, saving all information of connectivity in a file that can be executed from device by referring to its direction. The logfile.txt texts is saved in ...\Internal Storage\Android\data\edu.upc.wireless\files directory of device. To return an absolute directory path for an external file, getExternalFilesDir(String type) method of Context class helps to find package's directory. This method allocates an own external path where the application is placed. Then for writing bytes on a file that we already defined it as logfile.txt, FileOutputStream() method is applied.

```
File dir = context.getExternalFilesDir(null);
     File file = new File(dir, "logfile.txt");
     FileOutputStream fos = new FileOutputStream(file, true);
     PrintWriter pw = new PrintWriter(fos);
     pw.println(sentence);
     pw.close();
     fos.close();
```

Registering the _connec tivityReceiver in the Android Manifest File

The following snip code declares connectivityReceiver registration in the manifest file. In addition, an intent filter is added to this file. This intent filter specifies which broadcasts trigger the receiver to become activated.

```
<receiver android:name="._connectivityReceiver"
               android:enabled="true" android:label="ConnectivityReceiver">
          <intent-filter>
             <action android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
            </intent-filter>
```

```
</receiver>
```

Testing the connectivityReceiver

At this point, we tested the application on the device. When it comes back up, it monitors connections and notifies by a toast message that device is connected or not. In addition, we can verify its performance by looking at LogCat of output or by ...\Internal Storage\Android\data\edu.upc.wireless\files route and see all the connections are kept in this logfile.txt.

Summary

At this point, we tested the application on the device. When it comes back up, it monitors connections and will notify by toast message that device is connected or not. In addition, we can verify its performance by looking at LogCat of output or by Computer\XPERIA sole\Internal Storage\Android\data\edu.upc.wireless\files route and you can see all the connections are kept in this logfile.txt.

In following a part of WifiScanActivity which is related to broadcast receivers is explained.

C_WifiScanReceiver Broadcast Receivers

The wifiScanReceiver is a broadcast receiver applied in C_wifiScanActivity.java for listening to all the Wi-Fi connections of device and return the required information about state, IP address Mac address RSSI and others.

```
private class WifiScanReceiver extends BroadcastReceiver{
      public void onReceive(Context c, Intent i){
 WifiInfo wifiInfo = wifiManager.getConnectionInfo();
increaseCounter();
stateTextView.setText(headline+ "\n\nscan #"+counter+", wifi is enabled?
"+wifiManager.isWifiEnabled()+
            "\nwifiInfo.supplicantState: "+wifiInfo.getSupplicantState()+
            "\nwifiInfo.IPaddress: "+wifiInfo.getIpAddress()+
            "\nwifiInfo.MACaddress: "+wifiInfo.getMacAddress()+
            "\nwifiInfo.Rssi: "+wifiInfo.getRssi());
         stateTextView.append(Show_configurations());

         if(!wifiManager.isWifiEnabled() ||
wifiManager.getScanResults().isEmpty()) return;

         StringBuilder sb = new StringBuilder("SCAN RESULT #"+counter+"\n--------
-------\n");
         for(ScanResult scanResult: wifiManager.getScanResults()){
            sb.append(scanResult.SSID+"\n");
            sb.append(scanResult.BSSID+" = "+scanResult.level+" dBm\n\n");
         }
         sb.delete(sb.length()-1, sb.length());
         scanShotTextView.setText(sb.toString());
      }

      private void increaseCounter(){
         counter++;
         if(counter%2==0){
            scanShotTextView.setTextColor(ORANGE);
            stateTextView.setTextColor(BLACK);
         }
         else{
                scanShotTextView.setTextColor(BLACK);
                stateTextView.setTextColor(ORANGE);
            }
        }
    }
```

Show_configurations() method prepares us a list of network configurations and keep information such as SSID, status of the Wi-Fi and priority in a String Builder.

```
private String Show_configurations(){
      List<WifiConfiguration> configurations = 
wifiManager.getConfiguredNetworks();
      StringBuilder sb = new StringBuilder();
      sb.append("\n");
      for(WifiConfiguration conf : configurations){
        sb.append("\n SSID: ").append(conf.SSID).append(", priority: 
").append(conf.priority).append(", status: ").append(conf.status);
      }
      return sb.toString();
    }
```

The Database

The database of Android is used in order to store solicited and required information of an application even if the user kills the application or shut it down. A database in a mobile device is very useful as a supplement to the online world. We can have quick access to the database of an application without need of Internet connectivity. So we will use a database as a cache.

So in this part, we need to create a database and table then use it inside of application to store our status, actions and locations and update them.

About SQLite

SQLite as an open source database is using for a long time for projects. In addition, it is popular to apply for many small devices. There are some reasons to consider about its merit to use for Android. First is its Zero-configuration database that makes it simple to use. Second is its independency from server and its set of libraries that provide database functionality. Third point is its open source property and can be adopted with Android's framework easily. And at least is single-file database that provides us security straightforward.

MySQLiteHelper

Android provides an elegant interface for applications to interact with a SQLite database. To access the database, we need a helper class that provides a "connection" to the database, creating the connection if it doesn't already exist. This class, provided by the Android framework, is called SQLiteOpenHelper. The database class it returns is an instance of SQLiteDatabase.

The Database Schema and Its Creation

First of all, the content's database table is created to show the schema of the database. For this project, database is building to store and retrieve types of connection, locations of the user and his activities

As it is shown in the following table.

Table name: Connection

Fields (columns)	**Data type**	**Key**
Id	INTEGER	Primary Key
Connection type	TEXT	
Location	TEXT	
Activity	TEXT	

Each row has a unique ID in the table, and it will contain data from connections. In all database we need a primary key, so SQLite like most databases, allows us to declare the ID as a primary key and even assigns a unique number automatically to each query.

To build this table, an Object model of connection.java is created. That has two methods of setting and getting for each cell of the table.

```
public Connection(String connection, String location, String action)
 {
        super();
        this.connection = connection;
        this.location = location;
        this.action = action;
   }
```

In the other hand, the schema of database has to be established when the application starts, so should be created on onCreate() method of MySQLiteHelper. After creating the schema we need onUpgrade() method for calling to alter the schema.

The onCreate() and onUpgrade() are two method that I needed to use in SQL. I have executed the connection list table in onCreate() method for creating the table. Then asked to drop older connections table if existed in and create fresh connections table by onUpgrade() method. The DROP TABLE that is defined in onUpgrade(), destroys any data currently in the table.

Four Major Operations

The MySQLiteHelper class prepares us a high-level interface that is much simpler than SQL. The MySQLiteHelper is used to do four major operation named in following:

- insert(): Inserts one or more rows into the database
- query(): Requests rows matching the criteria you specify
- update(): Replaces ones or more rows that match the criteria you specify
- delete(): Deletes rows matching the criteria you specify

These statements are supported by Android framework that they should be passed directly to MySQLiteHelper. That's why we used execSQL() to run the code to CREATE TABLE.

Cursors

Curser is a pointer for applying along with a set of query rows. We can retrieve results one time from the cursor, and move the cursor around the result set. In general, anything is done with SQL could lead to a SQL exception because it is interacting with a system outside of our direct control.

MySQLiteHelper

Now I have created MySQLiteHelper java class to help me open the database of application. It will create the database file if one doesn't exist or will upgrade it.

The following snip code shows the overriding subclasses of class framework and then implementing class's constructor.

```
public class MySQLiteHelper extends SQLiteOpenHelper {
….
}
```

Therefore we should call name of database which is named connection, and make its instance. Then we need to define the DATABASE VERSION. This is an important element that is used in following code to provide users a way to upgrade the database to the latest schema when is changed.

```
public MySQLiteHelper(Context context) {
      super(context, DATABASE_NAME, null, DATABASE_VERSION);
   }
```

In previous code I have overridden SQLiteOpenHelper by passing the constants to super and retaining the local reference to the context.

Now we have to override onCreate() callback method of database by using actual SQL statements to create appropriate SQL schema of connection table. And then with execSQL() method we have created the table by passing it through onCreate().

```
 public void onCreate(SQLiteDatabase db) {
      String CREATE_CONNECTION_TABLE = "CREATE TABLE connections ( " +
             "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
             "connection TEXT, "+
             "location TEXT ,"+
             "action TEXT )";
       db.execSQL(CREATE_CONNECTION_TABLE);
   }
```

Then the onUpgrade() method is overridden to call whenever the user's database version is different from the application database version. It will drop the older connection table and creates a fresh connection table as following.

```
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
      db.execSQL("DROP TABLE IF EXISTS CONNECTION");
```

```
        this.onCreate(db);
    }
```

Then we are going to use following five methods in application.

- getConnection(): this method is to build a query and return it
- getAllConnections(): this method also builds the query and show us all queries by making a list of connections
- updateConnection(): this method creates Content/Values to add key "column"/value and updates rows
- deletConnection(): is using for deleting a value
- addconnection(): is explained in following.

For example, the addConnection() method is applying to add new connection in our database, so it starts with Log.d() method for logging and show us result later on logCat when we run the application. Then after getting a reference to writable database we created value instance of Content Values class to add key "column" based on each value. Then we insert them to the TABLE_CONNECTION.

```
public void addConnection(Connection connection){
        Log.d("addConnection", connection.toString());
         SQLiteDatabase db = this.getWritableDatabase();
         ContentValues values = new ContentValues();
        values.put(KEY_CONNECTION, connection.getConnection());
        values.put(KEY_LOCATION, connection.getLocation());
        values.put(KEY_ACTION, connection.getAction());
         db.insert(TABLE_CONNECTIONS,      values);
         db.close(); }
```

Connection.Java

The class of Connection.java is developed for creating desired table of database in order to save connections, locations and activities of the user and returns "Connection [id=" + id + ", type of connection is =" + connection + ", in the =" + location+ "]" + "and the usual motion is" + action" string. In this class all the setting values and getting them for each element of table has been defined. The following code shows a part of this java class.

```
public class Connection {
   public Connection(){}
   public Connection(String connection, String location, String action) {
      super();
      this.connection = connection;
      this.location = location;
      this.action = action;
   }
     public void setId(int id) {
      this.id = id;
   }
      public long getId() {
      return this.id;
   }

   public String toString() {
      return "Connection [id=" + id + ", type of connection is =" + connection + ",
in the =" + location
            + "]" + "and the usual motion is" + action;  } }
```

Update C_wifiScanActivity

Now it is time to perform the database in the activities. If you remember we had a button in layouts which has responsibility of showing into the local database.

The following code illustrates the way of adding db and dbHelper objects globally throughout the java class. It is include of an addressString for location and myAction string for Action column of database.

Then need to create an instance of MySQLiteHelper and pass this as its context. This works because the Android Service class is a subclass of Context. MySQLiteHelper will figure out whether the database needs to be created or upgraded.

```
 MySQLiteHelper db = new MySQLiteHelper(this);
      db.addConnection(new Connection("WIFI connection",addressString,
myAction));
```

And then by our button the f_ConnectionList.java will be loaded. The button has a setOnClickListener() method that we have to pass our action through this method.

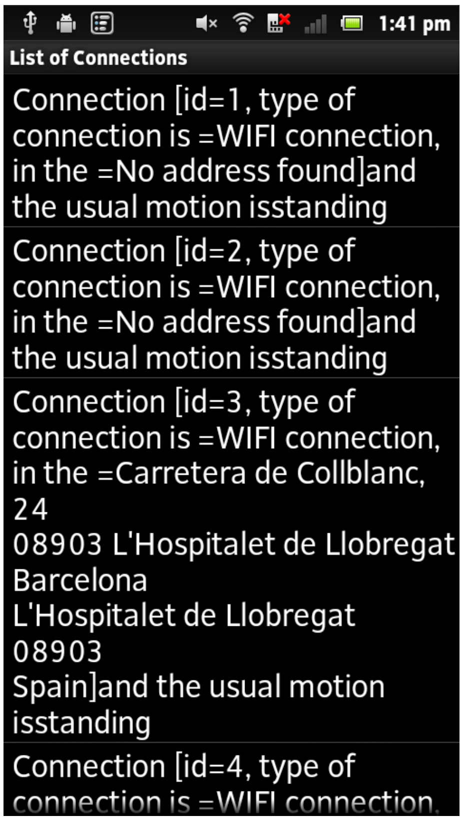
So as this is a listener element, it can apply by an OnClickListener in the activity or allocate an attribute of android: onClick in its XML layout.

```
 myButton = (Button)findViewById(R.id.button1);
        myButton.setOnClickListener(new View.OnClickListener() {
         public void onClick(View v) {
         Intent i=new Intent(c_WifiScanActivity.this, f_ConnectionList.class);
         startActivity(i);
             }
         });
```

The getAllConnection() is function includes a list object with name to show all the connections.

Test the application

The figure 26 is what we got from data base, as you can see with the specific ID it shows type of connection and location address if is defined by provider and the motion of user.



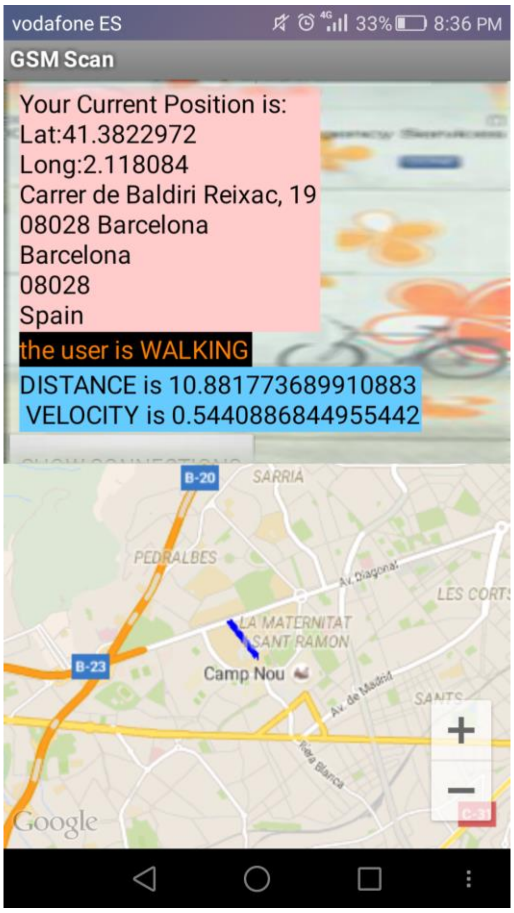
```
1:41 pm
List of Connections
Connection [id=1, type of
connection is =WIFI connection,
in the =No address found]and
the usual motion isstanding
Connection [id=2, type of
connection is =WIFI connection,
in the =No address found]and
the usual motion isstanding
Connection [id=3, type of
connection is =WIFI connection,
in the =Carretera de Collblanc,
24
08903 L'Hospitalet de Llobregat
Barcelona
L'Hospitalet de Llobregat
08903
Spain]and the usual motion
isstanding
Connection [id=4, type of
connection is =WIFI connection,
```

Figure 26: snapshot of database

Session 4

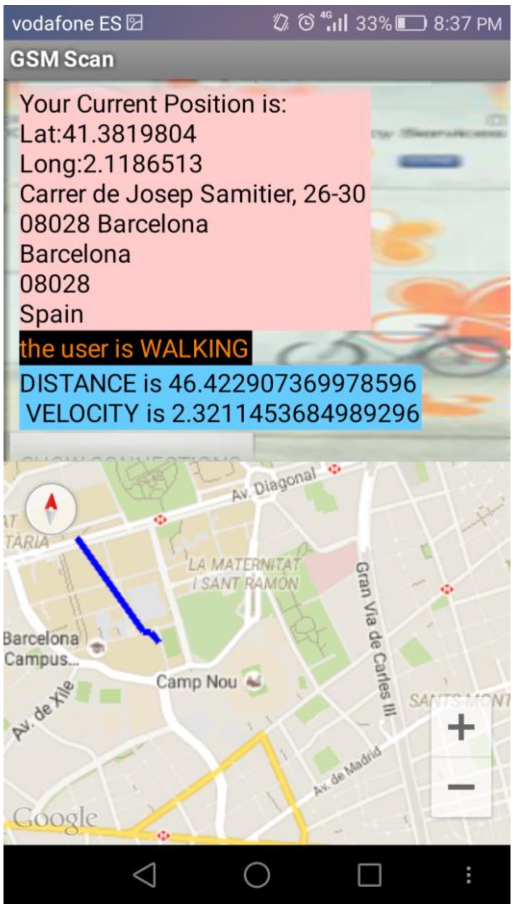
In this session, the project is run to see how it works. This test is done during a walking and running from UPC to Collblanc station in 03/09/2015.

Test of Project


vodafone ES 33% 8:36 PM

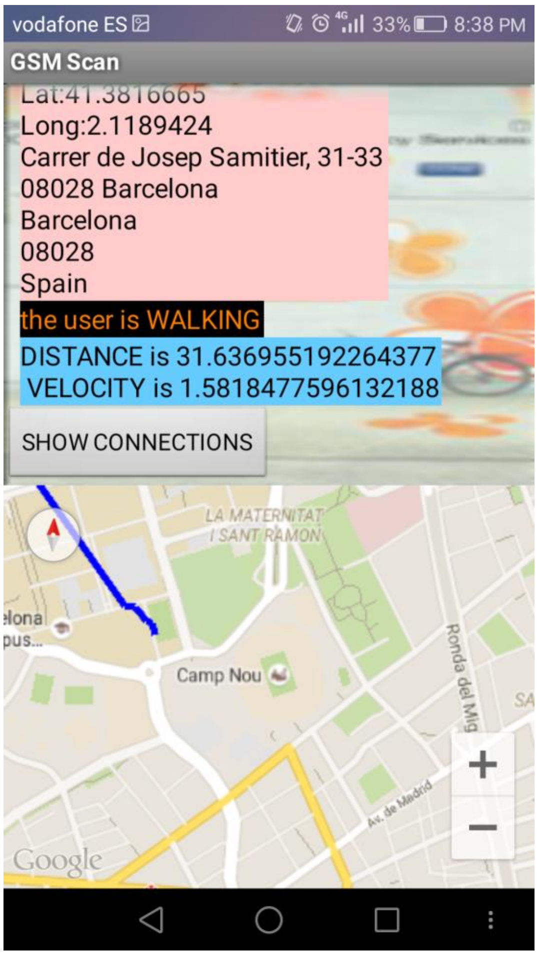
GSM Scan

Your Current Position is:
Lat:41.3822972
Long:2.118084
Carrer de Baldiri Reixac, 19
08028 Barcelona
Barcelona
08028
Spain
the user is WALKING
DISTANCE is 10.881773689910883
VELOCITY is 0.5440886844955442


vodafone ES 33% 8:37 PM

GSM Scan

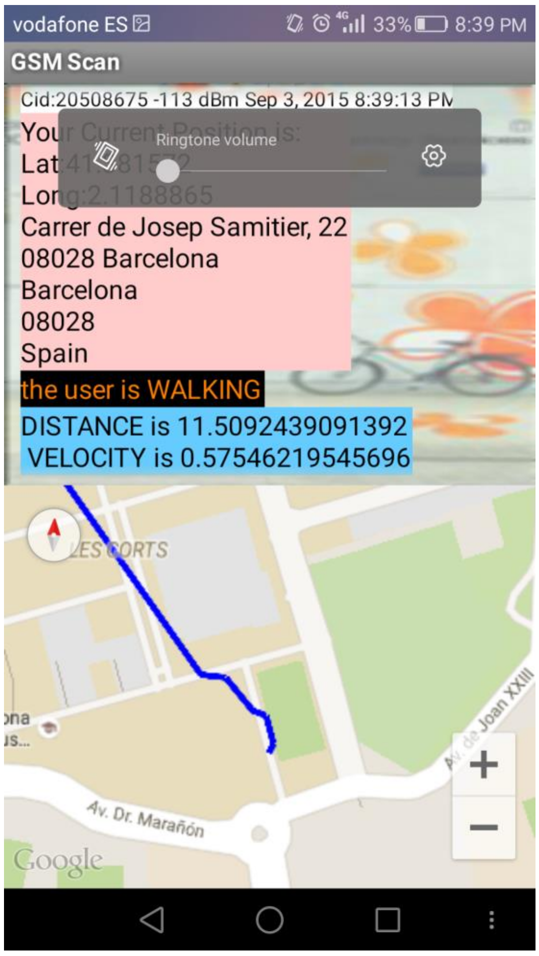
Your Current Position is:
Lat:41.3819804
Long:2.1186513
Carrer de Josep Samitier, 26-30
08028 Barcelona
Barcelona
08028
Spain
the user is WALKING
DISTANCE is 46.422907369978596
VELOCITY is 2.3211453684989296


vodafone ES 33% 8:38 PM

GSM Scan

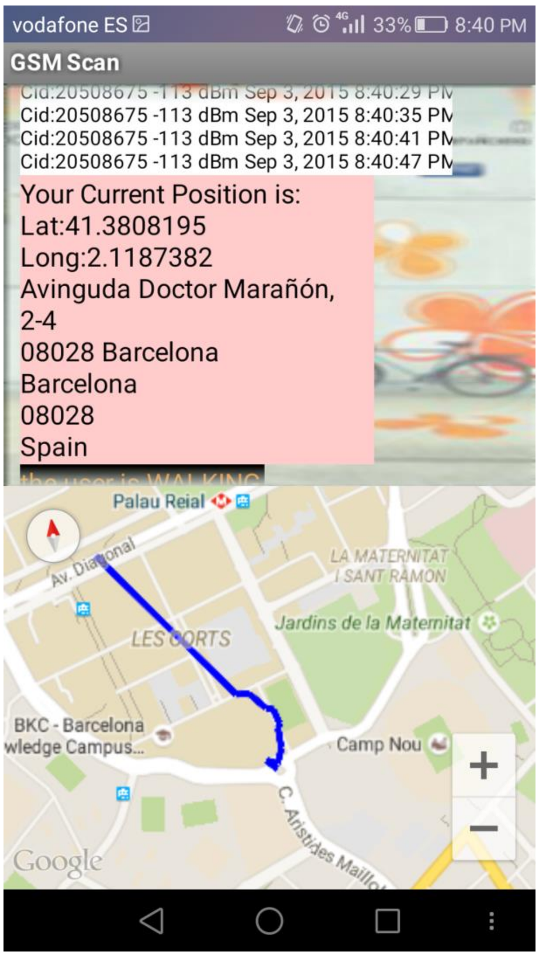
Lat:41.3816665
Long:2.1189424
Carrer de Josep Samitier, 31-33
08028 Barcelona
Barcelona
08028
Spain
the user is WALKING
DISTANCE is 31.636955192264377
VELOCITY is 1.5818477596132188

SHOW CONNECTIONS


vodafone ES 33% 8:39 PM

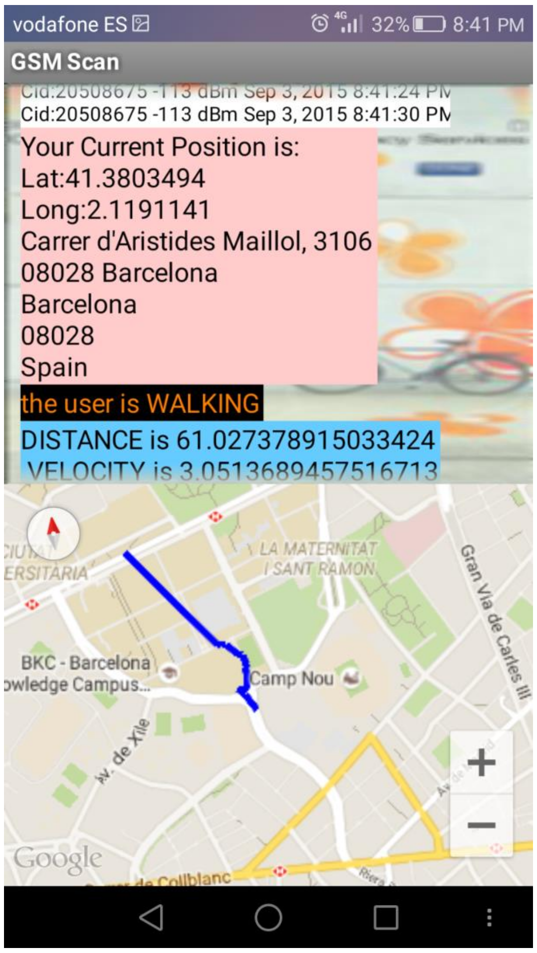
GSM Scan

Cid:20508675 -113 dBm Sep 3, 2015 8:39:13 PM
Your Current Position is:
Lat:41.3815772
Long:2.1188865
Carrer de Josep Samitier, 22
08028 Barcelona
Barcelona
08028
Spain
the user is WALKING
DISTANCE is 11.5092439091392
VELOCITY is 0.57546219545696


vodafone ES 33% 8:40 PM

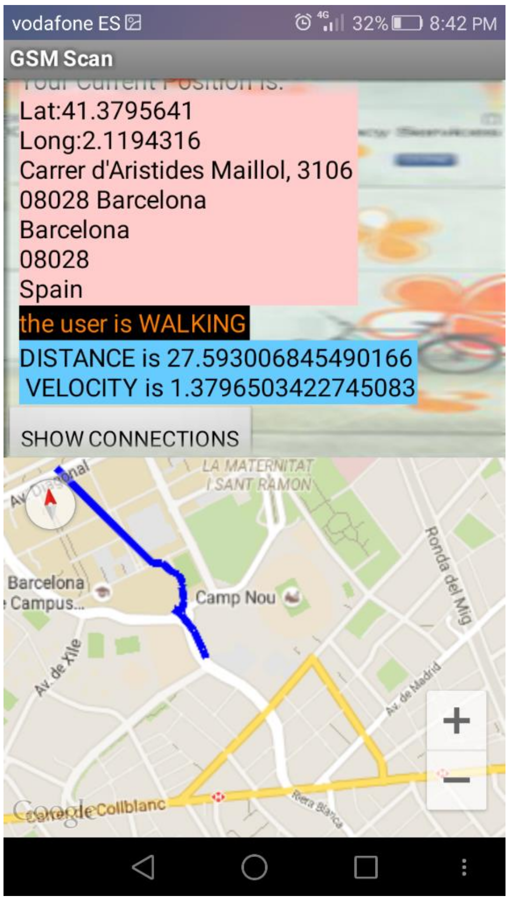
GSM Scan

Cid:20508675 -113 dBm Sep 3, 2015 8:40:29 PM
Cid:20508675 -113 dBm Sep 3, 2015 8:40:35 PM
Cid:20508675 -113 dBm Sep 3, 2015 8:40:41 PM
Cid:20508675 -113 dBm Sep 3, 2015 8:40:47 PM
Your Current Position is:
Lat:41.3808195
Long:2.1187382
Avinguda Doctor Marañón,
2-4
08028 Barcelona
Barcelona
08028
Spain
the user is WALKING


vodafone ES 32% 8:41 PM

GSM Scan

Cid:20508675 -113 dBm Sep 3, 2015 8:41:24 PM
Cid:20508675 -113 dBm Sep 3, 2015 8:41:30 PM
Your Current Position is:
Lat:41.3803494
Long:2.1191141
Carrer d'Aristides Maillol, 3106
08028 Barcelona
Barcelona
08028
Spain
the user is WALKING
DISTANCE is 61.027378915033424
VELOCITY is 3.0513689457516713


vodafone ES 32% 8:42 PM

GSM Scan

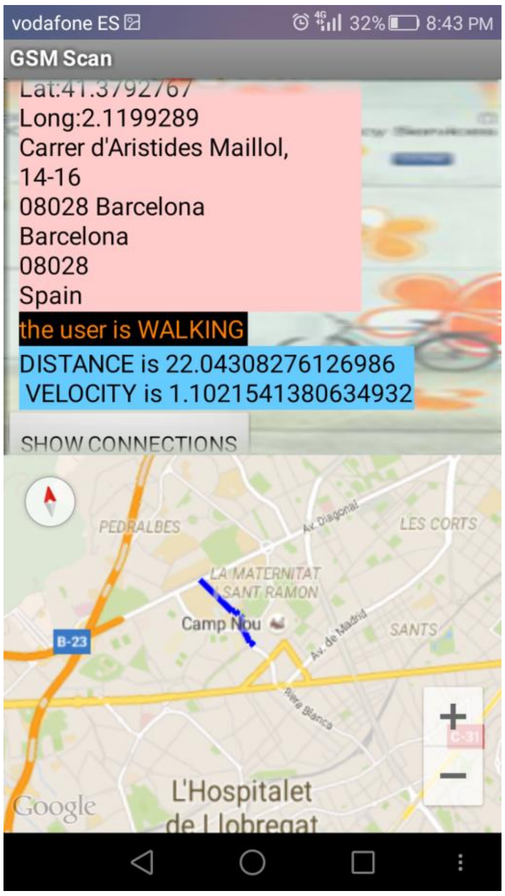
Your Current Position is:

Lat:41.3795641
Long:2.1194316
Carrer d'Aristides Maillol, 3106
08028 Barcelona
Barcelona
08028
Spain

the user is WALKING

DISTANCE is 27.593006845490166
VELOCITY is 1.3796503422745083

SHOW CONNECTIONS


vodafone ES 32% 8:43 PM

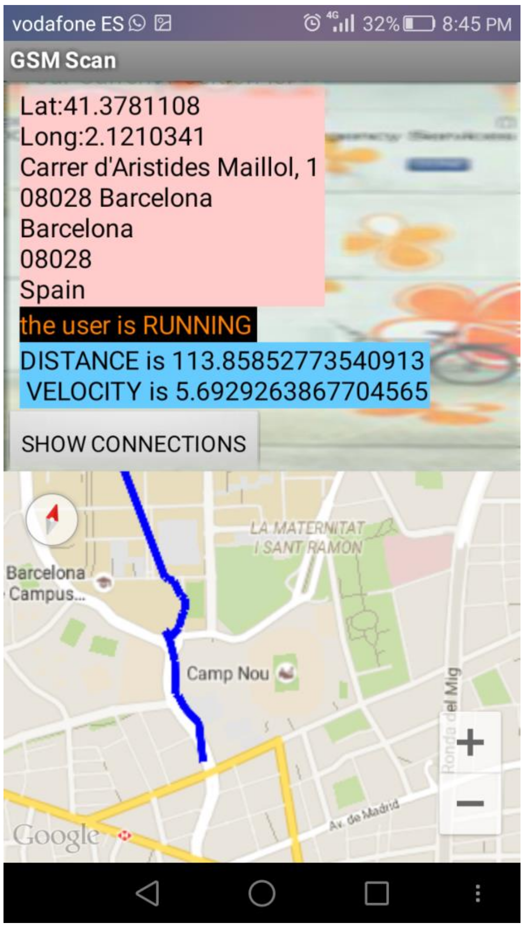
GSM Scan

Lat:41.3792767
Long:2.1199289
Carrer d'Aristides Maillol,
14-16
08028 Barcelona
Barcelona
08028
Spain

the user is WALKING

DISTANCE is 22.04308276126986
VELOCITY is 1.1021541380634932

SHOW CONNECTIONS


vodafone ES 32% 8:45 PM

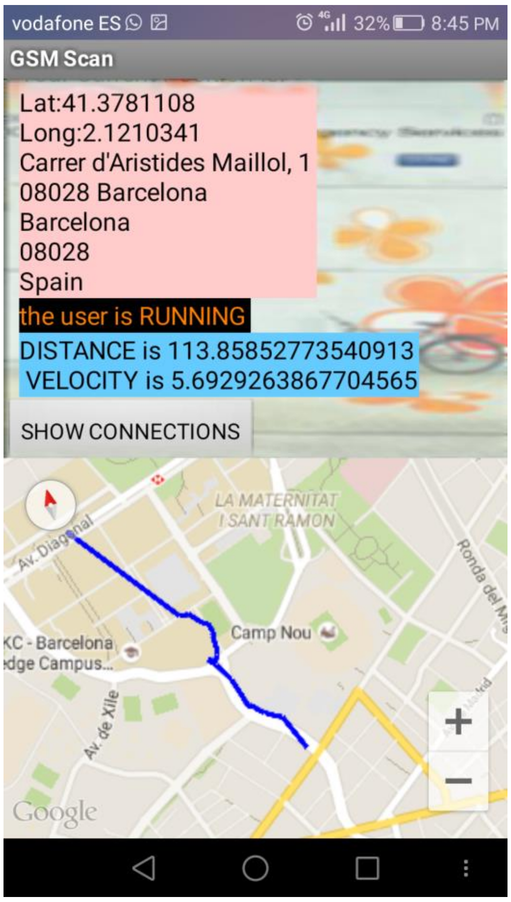
GSM Scan

Lat:41.3781108
Long:2.1210341
Carrer d'Aristides Maillol, 1
08028 Barcelona
Barcelona
08028
Spain

the user is RUNNING

DISTANCE is 113.85852773540913
VELOCITY is 5.6929263867704565

SHOW CONNECTIONS


vodafone ES 32% 8:45 PM

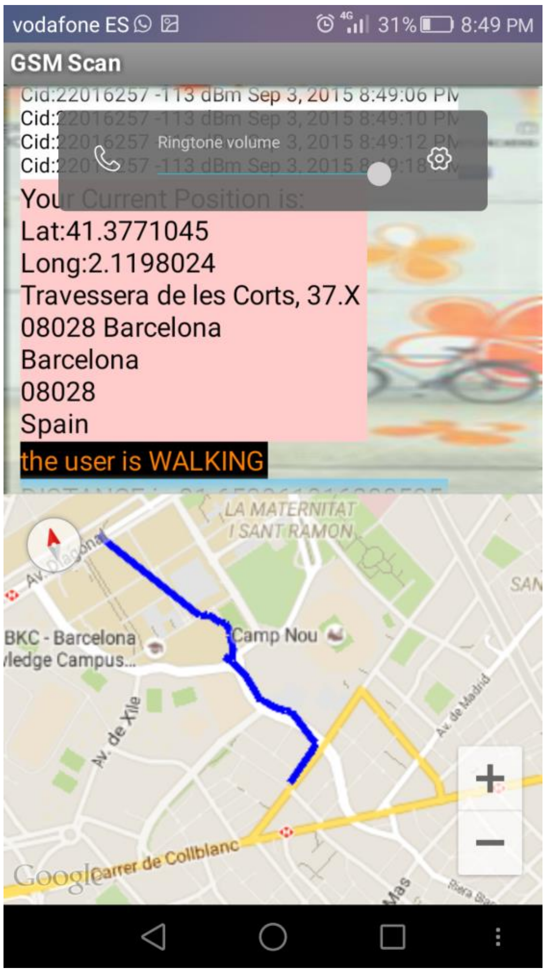
GSM Scan

Lat:41.3781108
Long:2.1210341
Carrer d'Aristides Maillol, 1
08028 Barcelona
Barcelona
08028
Spain

the user is RUNNING

DISTANCE is 113.85852773540913
VELOCITY is 5.6929263867704565

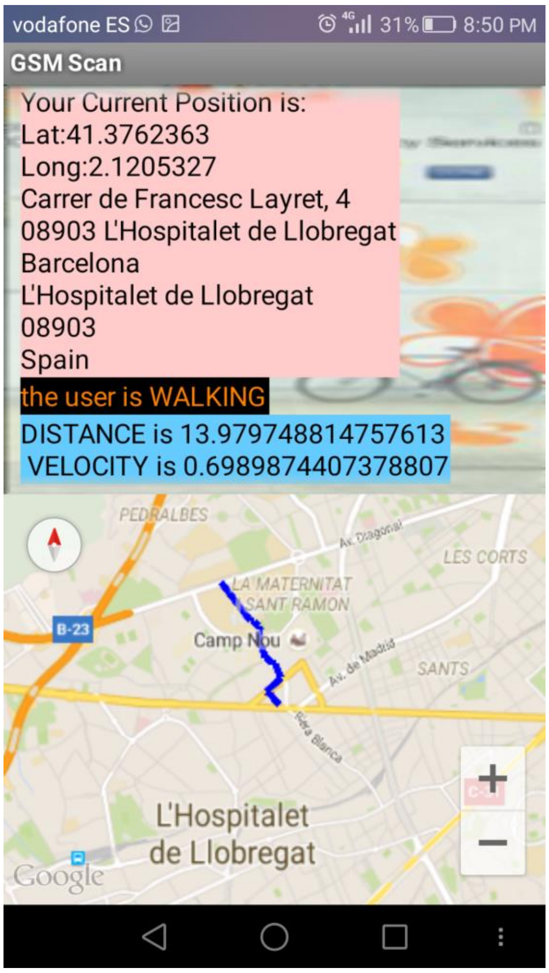
SHOW CONNECTIONS


vodafone ES 31% 8:49 PM

GSM Scan

Cid:22016257 -113 dBm Sep 3, 2015 8:49:06 PM
Cid:22016257 -113 dBm Sep 3, 2015 8:49:10 PM
Cid:22016257 -113 dBm Sep 3, 2015 8:49:12 PM
Cid:22016257 -113 dBm Sep 3, 2015 8:49:18 PM

Your Current Position is:
Lat:41.3771045
Long:2.1198024
Travessera de les Corts, 37.X
08028 Barcelona
Barcelona
08028
Spain
the user is WALKING


vodafone ES 31% 8:50 PM

GSM Scan

Your Current Position is:
Lat:41.3762363
Long:2.1205327
Carrer de Francesc Layret, 4
08903 L'Hospitalet de Llobregat
Barcelona
L'Hospitalet de Llobregat
08903
Spain
the user is WALKING
DISTANCE is 13.979748814757613
VELOCITY is 0.6989874407378807

Session 5

Conclusion

This study attempts to develop an android tool to recognize the connection type of the device, location of the user and specifically to identify the user's activity. Therefore, in this project, I have proposed an activity recognition tool of android devices which could be used as a part for other android applications which are based on location services and activity recognition. The hypothesis is to recognize five user's activities such as Standing, Walking, Running, Bicycling and Driving.

I based my project on the idea of using user's location instead of using device's embedded sensors such as accelerator. In order to achieve this goal the movement speed of user is in consideration. So the system diagnoses the device/user's latitude and longitude location every 20 seconds, then by utilizing haversine formula calculates the user's distance difference in this period.

Analyzing the output data of application demonstrated that 3 major activities including standing, walking and driving could be detect and distinguish accurately because of the difference between their rates while recognition of bicycling and running has less accuracy since they have similar rates.

Future work may consider implementing a service for users, to assure that automatic downloads or updates start to work when the device is connected to Wi-Fi. When it is connected to the GSM, any downloads and updates will be interrupted temporarily and the new frequency alters the program or postpones it based on the available bandwidth. The high prices of mobile data in-comparison to Wi-Fi connection could be a proper reason for the above, therefore for downloading and upgrading substantial size of data, Wi-Fi connection ~~is~~ should be required.

As a future work, we will also implement a user's track saver to know the user's regular track of a day.

Bibliography

- 1 - Professional Android 4 Application Development – Reto Meier - www.it-ebooks.info
- 2 - https://developer.android.com/training/index.html
- 3 – Learning Android – Marko Gargenta – O'reilly
- 4 – Android Programming Tutorials - Mark L. Murphy
- 5 - TUTORIALS POINT Simply Easy Learning
- 6 - Fall Detection based on movement and smart Phone Technology, Vo Quang Viet, Gueesang Lee.
- 7 - ACTIVITY RECOGNITION USING K-NEAREST NEIGHBOR ALGORITHM ON SMARTPHONE WITH TRI-AXIAL ACCELEROMETER Sahak Kaghyan, Hakob Sarukhanyan.
- 8 - Implementation of Location based Services in Android using GPS and Web Services - Manav Singhal1, Anupam Shukla - ABV-Indian Institute of Information Technology and Management - Gwalior, India.
- 9 - "Location Based Services on Mobile in India - For IAMAI - Version: 14 April 2008 *http://www.iamai.in/Upload/policy/LBS_Draft_Indicus.pdf*"
- 10 - [THE MOBILE USE OF THE INTERNET BY INDIVIDUALS AND ENTERPRISES]
- 11- Implementation of Location based Services in Android using GPS and Web Services - Manav Singhal1, Anupam Shukla2 - 1ABV-Indian Institute of Information Technology and Management, Gwalior, India - 2ABV-Indian Institute of Information Technology and Management Gwalior, India

12 – "Online Human Activity Recognition on Smart Phones" - Mustafa Kose, Ozlem Durmaz Incel, Cem Ersoy - Bogazici University, Istanbul, Turkey

13 – "Location Based Services using Android Mobile Operating System" - Amit Kushwaha1, Vineet Kushwaha - Department of Electronics & Communication Engineering - IIMT Engineering College, Meerut-250001, India