Development of Monitor for Android systems

Android OS App to control and monitorize a motor

Project memory

Created by:	Francesc Trallero Blanca
Student number:	69887
Faculty:	Electronic and Informatic
Studies coursing:	Industrial Electronic and Automatism
Supervisor:	Prof. Dr.-Ing. Heinrich Steinhart Laboratory for power electronics and electric drive technology

Development of Monitor for Android Systems

Abstract

This project has developed an application for operating systems based on Android 4.0 or higher. This app must be able to interact with the systems which it is connected. In the specific case for which it has been realized, this project allows to control and obtain information, for example, of an engine. The data are transmitted over the internet, through a Wi-Fi connection, which is done through a board that in turn makes the connection with the motor.

Key words: Android Studio, Java, Wi-Fi, hotspot, layout, IP, port, socket, server, client.

Abstracte

En aquest projecte s'ha desenvolupat una aplicació per a sistemes operatius basats en *Android* 4.0 o superiors. Aquesta *app* ha de poder interactuar amb els sistemes als qual es connecta. En el cas concret en el qual s'ha realitzat, aquest projecte permet controlar i obtenir la informació, per exemple, d'un motor. Les dades son transmeses per internet, mitjançant una connexió Wi-Fi, que es realitza a través d'una placa que a la vegada realitza la connexió amb el motor.

Paraules clau: Android Studio, Java, Wi-Fi, hotspot, layout, IP, port, socket, servidor, client.

Abstracto

En este proyecto se ha desarrollado una aplicación para sistemas operativos basados en Android 4.0 o superiores. Esta app debe poder interactuar con los sistemas a los cuales se connecta. En el caso concreto para el cual se ha realizado, este proyecto permite controlar y obtener información, por ejemplo, de un motor. Los datos son transmitidos por internet, mediante una conexión Wi-Fi, que se realiza a través de una placa que a su vez realiza la conexión con el motor.

Palabras clave: Android Studio, Java, Wi-Fi, hotspot, layout, IP, puerto, socket, servidor, cliente.

Glossary

IDE – Integrated Development Environment

OS – Operative System

WORA – Write Once Run Anywhere

JVM – Java Virtual Machine

IP – Internet Protocol

Wi-Fi – Wireless Fidelity

API - Application Programming Interface

App – Application

SSID – Service Set Identifier

Dpi – Dots per Inch

Dp – Density-independent Pixels

USB – Universal Serial Bus

APK – Android Application Package

XML – eXtensible Markup Language

LED – Light Emitting Diode

TCP – Transmission Control Protocol

US – United States

DE – Deutschland

Acknowledgment

First, I would like to thank all those people who have supported me during the completion of this final project.

I am especially grateful to Prof. Dr. Ing. Heinrich Steinhart for overseeing this project as well as his advice and kindness towards me.

I also thank all my colleagues in the laboratory, who during these months have provided me with invaluable help during the development of the application. I would also like to thank all those programmers who, through the use of differences forums on internet have helped others, in solving code errors or similar problems, among which I include myself.

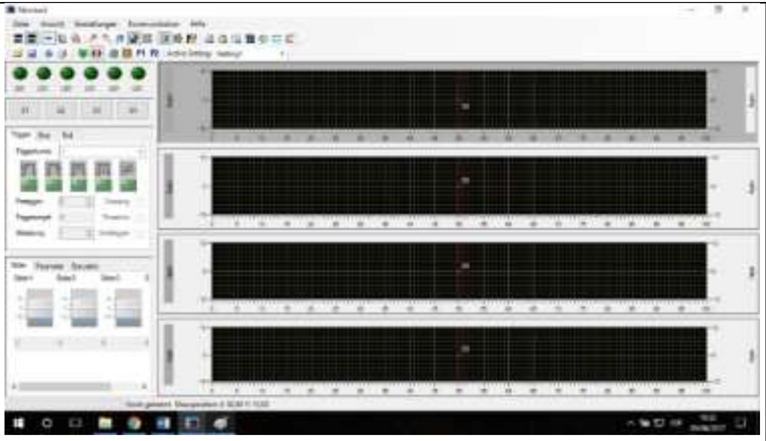
Aalen 20.06.2017

Francesc Trallero Blanca

Introducction

About Monitor 2

Monitor 2 is a computer program developed by the *Hochschule Aalen*. This program allows the connection to a system which must be monitored, by means of a board like *STM32F4DISCOVERY* as it has been used, as its name indicates, in a *STM32F407* microcontroller. Even that, the program of this microcontroller is made in *C* language, so many boards can handle it. In the specific case of *Monitor 2,* the control has performed during this project on a motor, as an example system to interact with, and the board in question reads of diverse magnitudes: currents, voltages, etc.

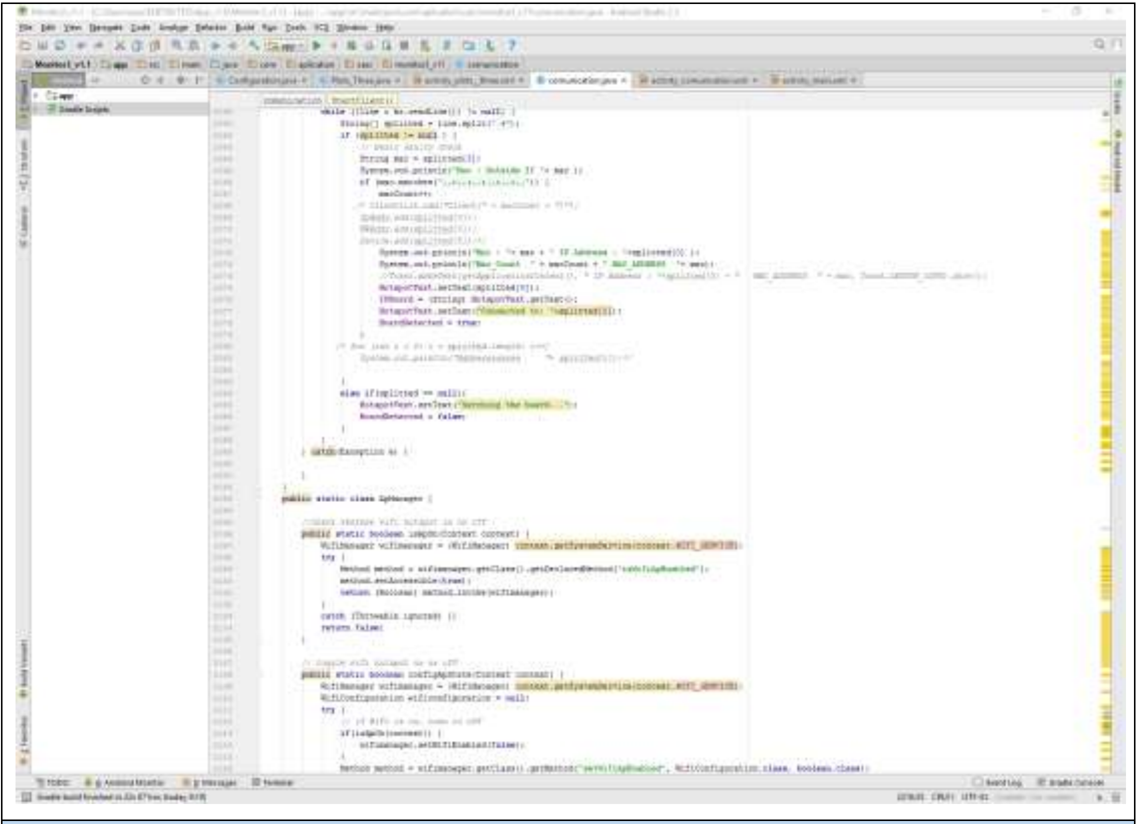
	
Picture 1: board *STM32F4DISCOVERY*	***Picture 2:*** *Monitor 2*, computer program

About Android Studio and Java Language

Android Studio

Android Studio is an official integrated development environment for the *Android* operating system based on the *Java* programming language.

Created by Google on May 16, 2013, and released in December 2014, *Android Studio* replaced *Eclipse* as the official *IDE* for developing applications on Android *OS*. That is reason why, for this project, this program has been chosen (*v.1.0*) instead others such as *Eclipse* or *NetBeans*.



[illegible]

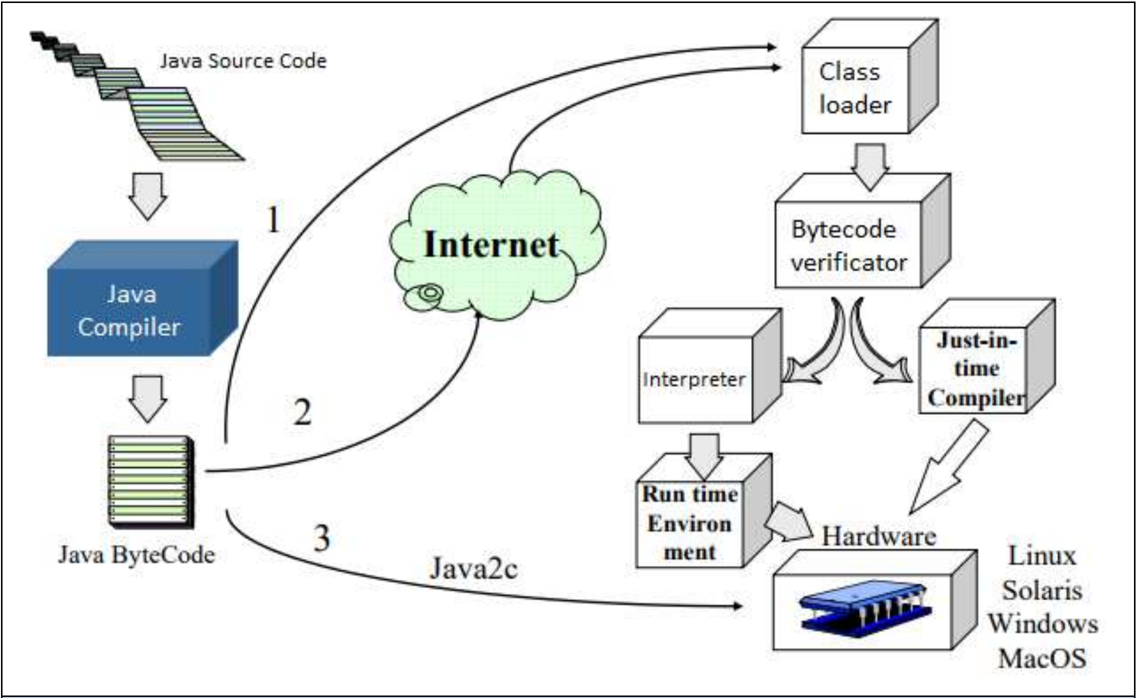
Picture 3: Android Studio IDE application development environment

Java Language

It is appropriate before going completely into the subject to do a little review on the fundamentals of Java, as it may be necessary if any of the readers of this report has never worked in this programming language.

Java is an object-oriented programming language, which means that it allows the development of basic applications, business, mobile, etc.

Java was born as a programming language that could be multiplatform and multi-device, under the paradigm of *WORA*, "*Write Once Run Anywhere*". In this way, a program written in Java, must be able to be executed in *Windows*, *MacOS* and *UNIX*. In order to achieve this, Java does not generate source code but *bytecodes*, which are interpreted by a virtual machine called JVM (Java Virtual Machine).



Picture 4: How Java works

That said, let's look at some basic concepts that we should know when programming in Java and we will focus on programming for mobile devices; especially because it is the reason of this project. So, three basic concepts to learn are: Object, Class and Inheritance.

The object is an element of software that attempts to represent a concept of the real world. It has specific properties as well as actions that can be performed on it. The properties of these objects and the interactions with them are made by methods. If we want to know information about a particular object, we call one of its methods to know it; instead of directly accessing the properties of that object.

```
IPBoard = (String) HotspotText.getText();
```

Classes are a generalization of a set of objects. It is where we really define the properties and methods that each of the instances of the objects can contain. This section leads to the inheritance, since both are strongly linked.

Inheritance is a way of structuring software. By means of it we can indicate that one class inherits from another, in which the class extends the capacities, properties and methods that it has and adds new properties and actions.

```java
public class comunication extends AppCompatActivity
```

Variables in Java work in the same way as in any other programming language, although it must be borne in mind that objects are also usually defined in a similar way to the usual variables.

```java
float IntOffset1 = 0;
float IntOffset2 = 0;

EditText Unit1;
EditText Unit2;
```

For those people who have previously worked with programming in *C#*, as it was my case before learning Java, you will see that there are many similarities between both, however Java is a programming language of higher level and complexity and has the opportunity of multiplatform, which *C#* does not, since it was developed by *Microsoft*.

Devices used

This project has been carried out mainly by communicating two Tablets to each other. The one that carries the application of *Monitor 3* and one that carries an auxiliary program to demonstrate the functionality of the *app*.

Even so, the ultimate goal of such an application is the connection to the *CC3200* board. This board communicates through *Wi-Fi* with the *Android* operating system and also with the system to be controlled (as example, the engine previously discussed).

Picture 5: Connection board *CC3200*

In addition to communication through the *Android* devices, tests have also been performed with the board and verifying their correct operation and thus obtaining positive results.

The main Tablet used was an ODYS XELIO 10, while the auxiliary was a SAMSUNG GALAXY Tab2 7.0.

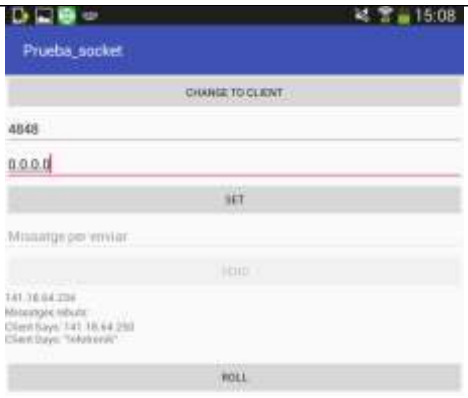
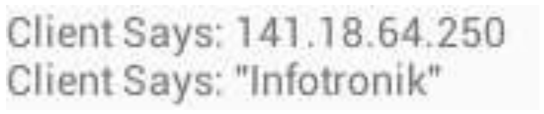
Monitor 3 for Android

How does the App works

As we already know this application intend to communicate with a board that will send you the data to be plotted. In general terms, when a socket server is enabled, it is not necessary to know the IP of the device which we are communicating with, however, by constitution or security programming, said board it is need to know the IP of the server and the name of *Wi-Fi* it should be connected. Since the board needs these information, a *HotSpot* method has been developed to allow an initial communication between both systems.

This method of *HotSpot* is intended to solve the problem of delivering the *IP* and network to the board. By creating this access point we can connect to the desired board using a specific port, that is to say, if we have three boards and each program default carries a different port, we can use an *editText* to select the port and thus choose the board (given that now there is only one board, the default port is *4848*).

So, starting the application with the *Wi-Fi* enabled, the app will read its IP and it will turn on a *HotSpot*. This mentioned connectivity, will allow the board to be connected to the tablet, being that connection detected by this one and thus saving the *IP* of the detected device. Connecting as a server to the port, the board receives the *IP* and the *Network* on which *Monitor 3* works (outside of the *HotSpot*).

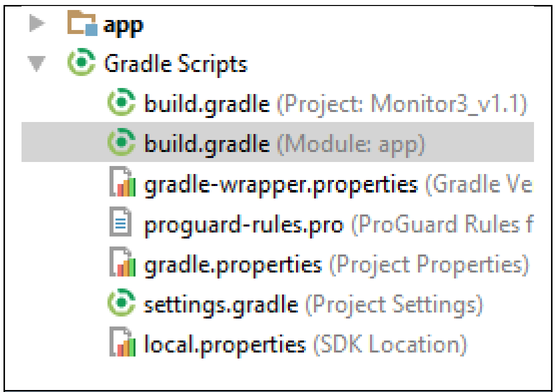
 15:08 Prueba_socket CHANGE TO CLIENT SET Missatge per enviar ROLL 4848 0.0.0.0 141.18.64.236 Missatges rebuts: Client Says: 141.18.64.250 Client Says: "Infotronik"	 Client Says: 141.18.64.250 Client Says: "Infotronik"
Picture 6: Date received in the auxiliary app	***Picture 7***: Zooming of received data

Received this data, the *HotSpot* will automatically be disconnected and the tablet, also automatically, connects to the last *Wi-Fi* in which it was working. Both sides, tablet and board, back on the same *Wi-Fi*, start with sending the IP of the board to the tablet. The latter is already in default server mode after the connection to the *Wi-Fi*, so that when receiving the *IP* will save it to be able to use it in the rest of activities. At the end of this process the app is ready to start working normally.

It is possible to say that the *IP* of the tablet is dynamic and could have changed, that is why the initial *IP* and the final *IP* are displayed. In case of not agreeing, reconnecting pressing *Connect* button again, will make these steps performed again until it coincided.

Gradle App

Gradle is an automation tool for building our code. As a general rule, this file is only modified to add libraries, in the case of this project and before entering, how the programming of the app has been made, it is necessary to take into account some important points in the *gradle* of this application.


```
▶ app
▼ Gradle Scripts
    build.gradle (Project: Monitor3_v1.1)
    build.gradle (Module: app)
    gradle-wrapper.properties (Gradle Ve
    proguard-rules.pro (ProGuard Rules f
    gradle.properties (Project Properties)
    settings.gradle (Project Settings)
    local.properties (SDK Location)
```

Picture 8: Location of *gradle* in the *IDE*

Libraries

Libraries are lines of code within our program that allow us to simplify our tasks by simply calling a function. They are the most essential part, at the time of making an application that must graph, because without libraries ourselves we should perform the whole process of creating an object to display it, etc.

So, before choosing which library to use, a thorough research has been made of the advantages and disadvantages of each of those that exist for the *Android* environment. Finally it has been decided to use a library called *MPAndroidStudio*, it works in *APIs 8* or higher and it is an open-source *Apache license v.2.0.*

The reason to choose this one and not another one like *GraphView* is that the finish of *MPAndroidStudio* is more professional, allows to make samplings of greater complexity and, although its use is more complex, it is more complete.

```
compile 'com.github.PhilJay:MPAndroidChart:v3.0.1'
```

API

API (Application Programming Interface) is the set of functions and procedures that a certain library offers to include specifications of routines, data structures, classes of objects and variables.

The *API* of an application is an integer value that is in the gradle as *targetSdkVersion*. In the case of our app was identified as 25 (also corresponding with the version of *Android 6.0*). For the use of the *Hotspot* the version of the *API* had to be reduced to 21 to allow it to work appropriately. For some reason, in versions superior to *Android 5.0* the use of the *Hotspot* was not viable and for it was necessary to lower the version.

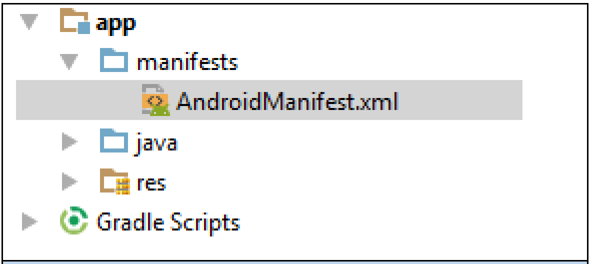
```
targetSdkVersion 21
```

Also this line of code was added in the *Manifest* to be able to solve this error.

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

Manifest – Application Permissions

Inasmuch as we are talking about the *Manifest* file; this file is in charge of the permissions that the application needs as well as configurations, styles, icons, type of app, etc. Therefore, it is important to take into account several aspects that have been modified during the creation of the application.



Picture 9: Location of *Android Manifest*

Permissions

The application developed here needs several permissions related to network modification conditions and for *IP* and *SSID* readings. That is why several of the permissions imply "*ACCES*" and "*CHANGE*".

The following are the necessary permissions for the operation as well as the previously mentioned permission of "*WRITE_SETTINGS*".

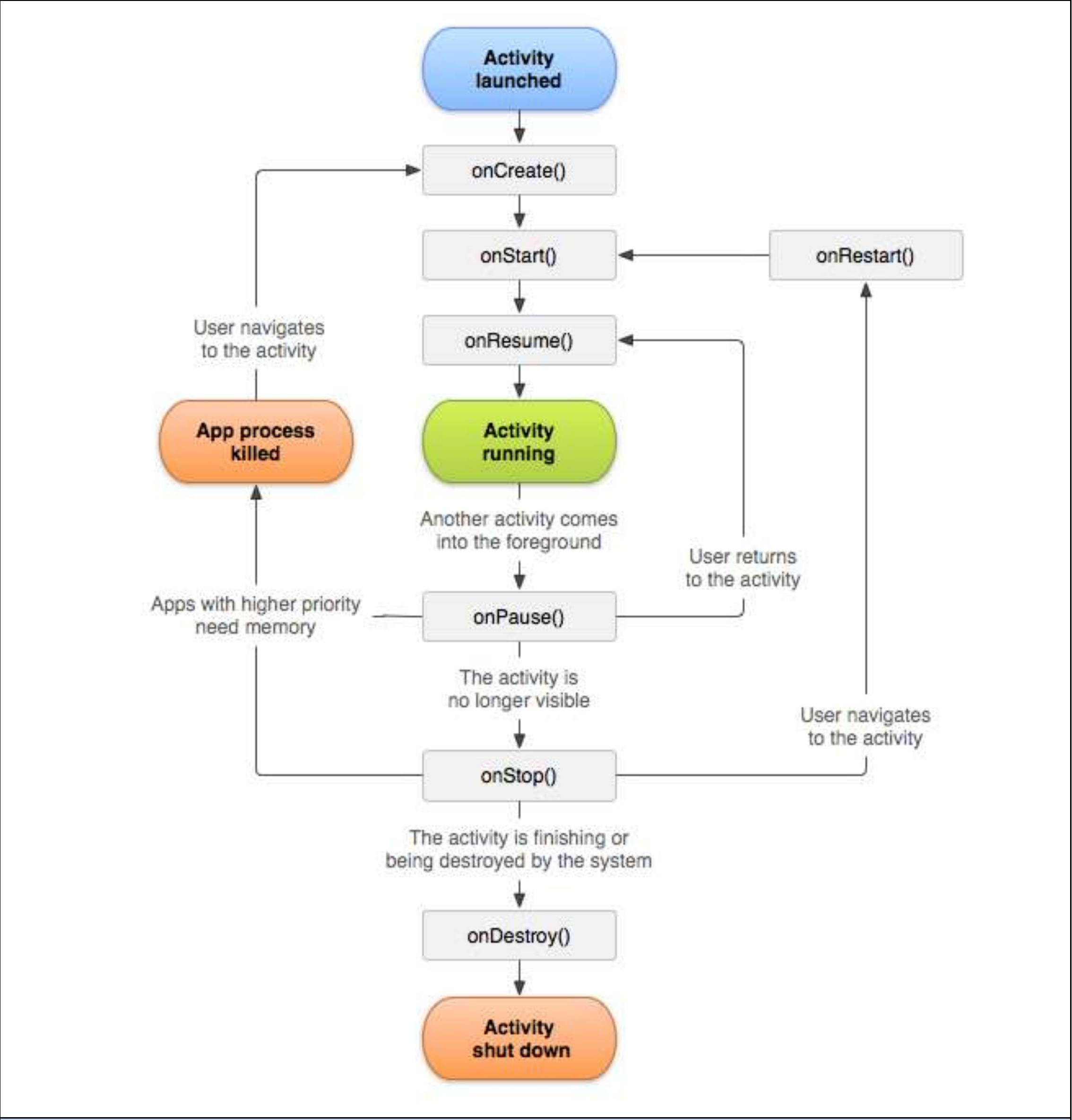
```xml
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"></uses-permission>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.WRITE_SETTINGS" />
<uses-permission android:name="android.permission.CHANGE_WIFI_MULTICAST_STATE" />
```

Activity Lifecycle

In slightly more abstract terms, we speak of the life cycle of an activity. The lifecycle consists of a set of functions, which are automatically called when a certain event occurs. So, when we start an activity, the *onCreate( )* function is called because, at its name is said, we are *creating* the activity.

This function of our code is where the basis of our programming occurs, the variables are linked to the layout components and we can perform the main program that will use that activity.

One of the requests we have in an application, is that it musts work in real time, is solving the situation of when the user locks the screen or navigate between the activities that has the app. Generally any of those situations would start the function *onStop( )* and later *onDestroy( )*, however it is not what interests us, since we want to continue working in case the screen is blocked.


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

Picture 10: Activity Lifecycle

OnStop( ) is also called when the screen is rotated from horizontal to vertical or vice versa. This is also a problem for real-time activity. If we turn the screen after standing for a moment, the cycle shows us that the activity goes to *onRestart( ),* causing all the data already graphed to be lost and the activity starts again with the data that remains in the server port, if there is any more information left.

Thus, a solution has been thought to avoid this cycle follow-up. To do this, and starting with the case of the orientation, it has been decided to apply the following line of code, which forces the screen to ignore its position and always remain horizontal. This request must be in all the activities we made in order to force them to be in the horizontal position even if the user change it.

```
setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE); //set orientation
```

As with the orientation, the lock causes the reset of the data and therefore the loss of these. This inconvenience has been solved with a modification of the change configuration in the *Manifest* file. In the following line of code you can see how it was used:

```
android:configChanges="keyboardHidden|orientation|screenSize":
```

This causes a "jump" in the lifecycle, avoiding the reset that occurs after the stop, however, the activity stops and does not continue to plot, although it resumes upon entering the screen again. That basically means that the app is not working while the screen is off, but it also does not lose information because it will take it again once the user gets in the activity again.

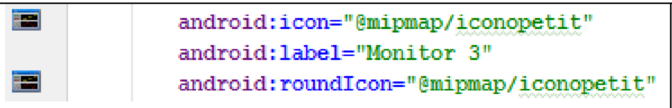
Launcher y Default Activities

As in many applications, *Monitor 3* has an initial boot screen that is commonly called *Splash Screen.* It consists of an activity that acts as an intermediary between the start of the app and the main screen. In order to make a screen like that, is necessary to define the activity in question as *Launcher* and the main one as *Default*. This can be found, in this *Manifest.xml* that we are talking about, as *intent categories*.

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

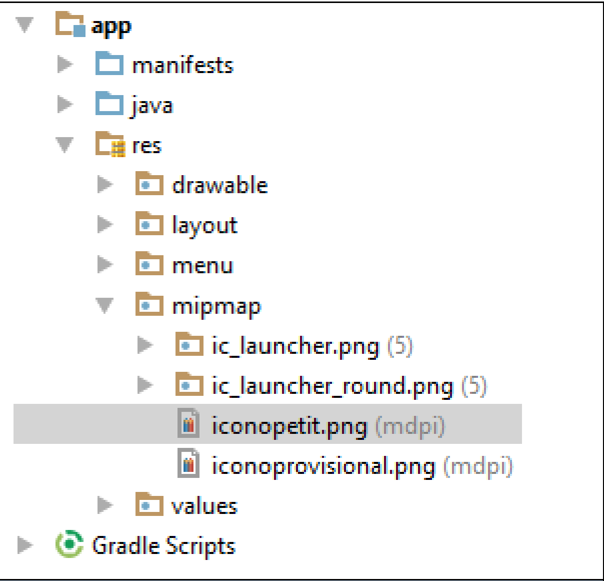
Icon and Name

Also in the Manifest we can define what name we want to use for the application we are developing and the icon we want to display (if we do not introduce any name, automatically the *Android Studio IDE* creates it with the name given to the program and the icon would be *Andy*, the robot image of android).

	
Picture 11: Icon of the application	***Picture 12:*** Default Icon

```
android:icon="@mipmap/iconopetit"
android:label="Monitor 3"
android:roundIcon="@mipmap/iconopetit"
```

Normally the images used in the application are stored in the *res/drawable* directory, however for the app icon, it must be saved in *res/mipmap*.



```
app
  manifests
  java
  res
    drawable
    layout
    menu
    mipmap
      ic_launcher.png (5)
      ic_launcher_round.png (5)
      iconopetit.png (mdpi)
      iconoprovisional.png (mdpi)
    values
Gradle Scripts
```

Picture 13: Location of the icon inside of *res*

Layouts

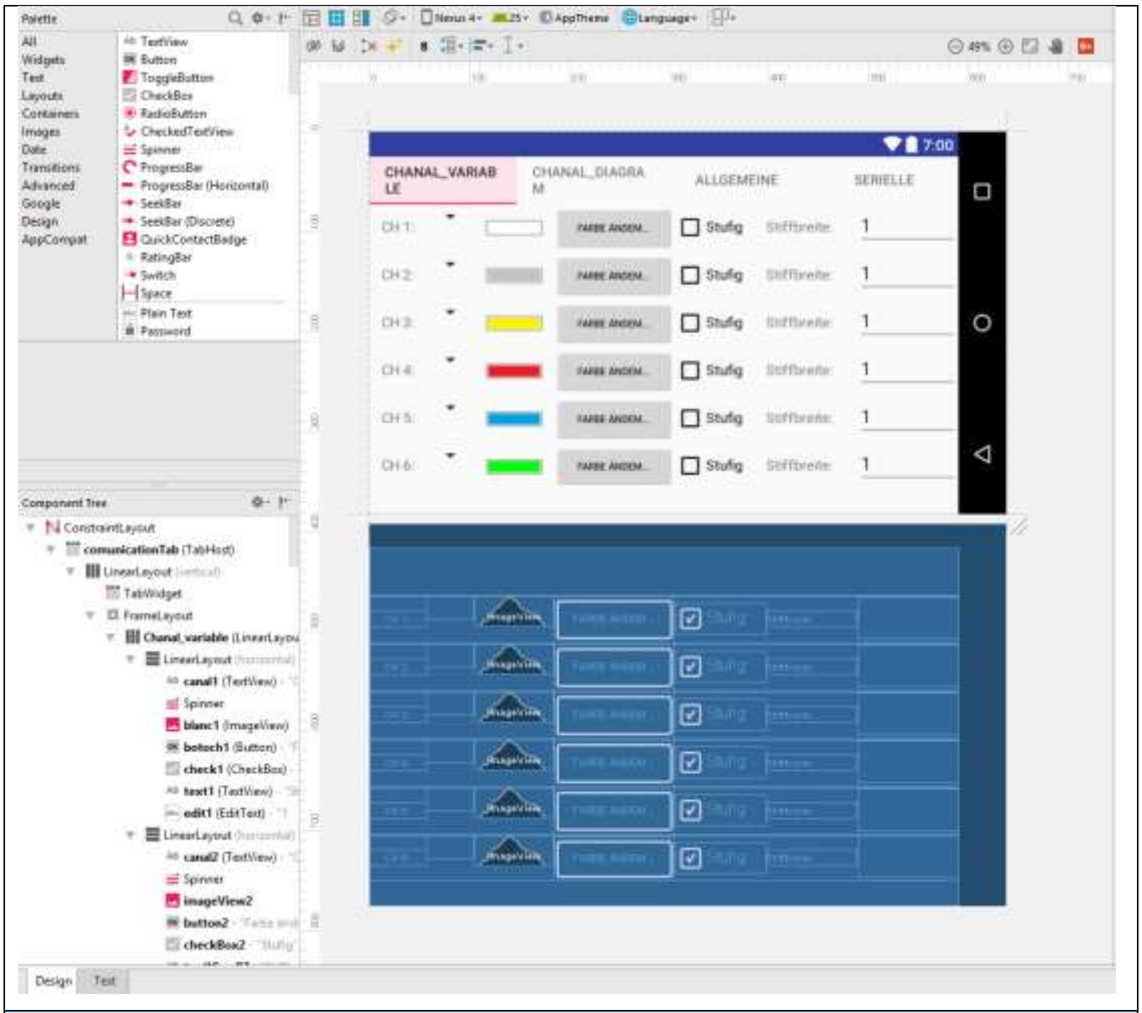
Layouts are the direct connection to the user. That is why there are different studies that focus on the interfaces, making the user-device relationship as better as possible, is what is usually called user-friendly interfaces. The user-friendly is the term that is used to refer those aplications that do not need explanation to the user to understand them, they must be intuitive. To achieve this, It has been tried to make an efficient application, with an easy navigation using the back button and the icons on the left of the screen.

These files require a little imagination and design, as well as the use of the appropriate icons, colors and proper placement of the components that make up the application.

There are two ways to do application layouts.

The first one, and usually the least used due to its limitations, is the design mode. On this mode we can see on the computer screen how the interface we are creating is, despite that, you have to download the model of phone or tablet that you want your application to run on. Otherwise it is possible to use the *Nexus* that come by default with different inches of screen to emulate the device we want, but this method is not as safe as downloading the model we want or using the real device as I made during this project.

As it is said this is the less used mode because it does not fix correctly the components. Also, manipulating their properties from the design side is quite complicated and tedious.

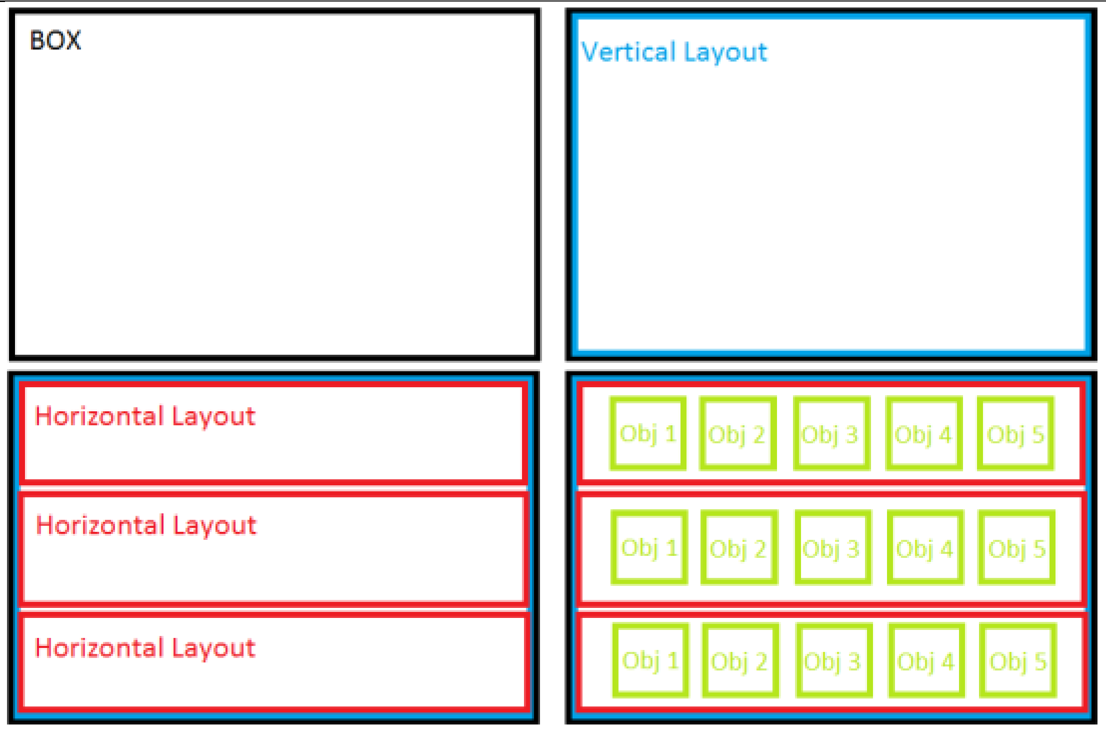


Picture 14: Creation of layouts in design mode

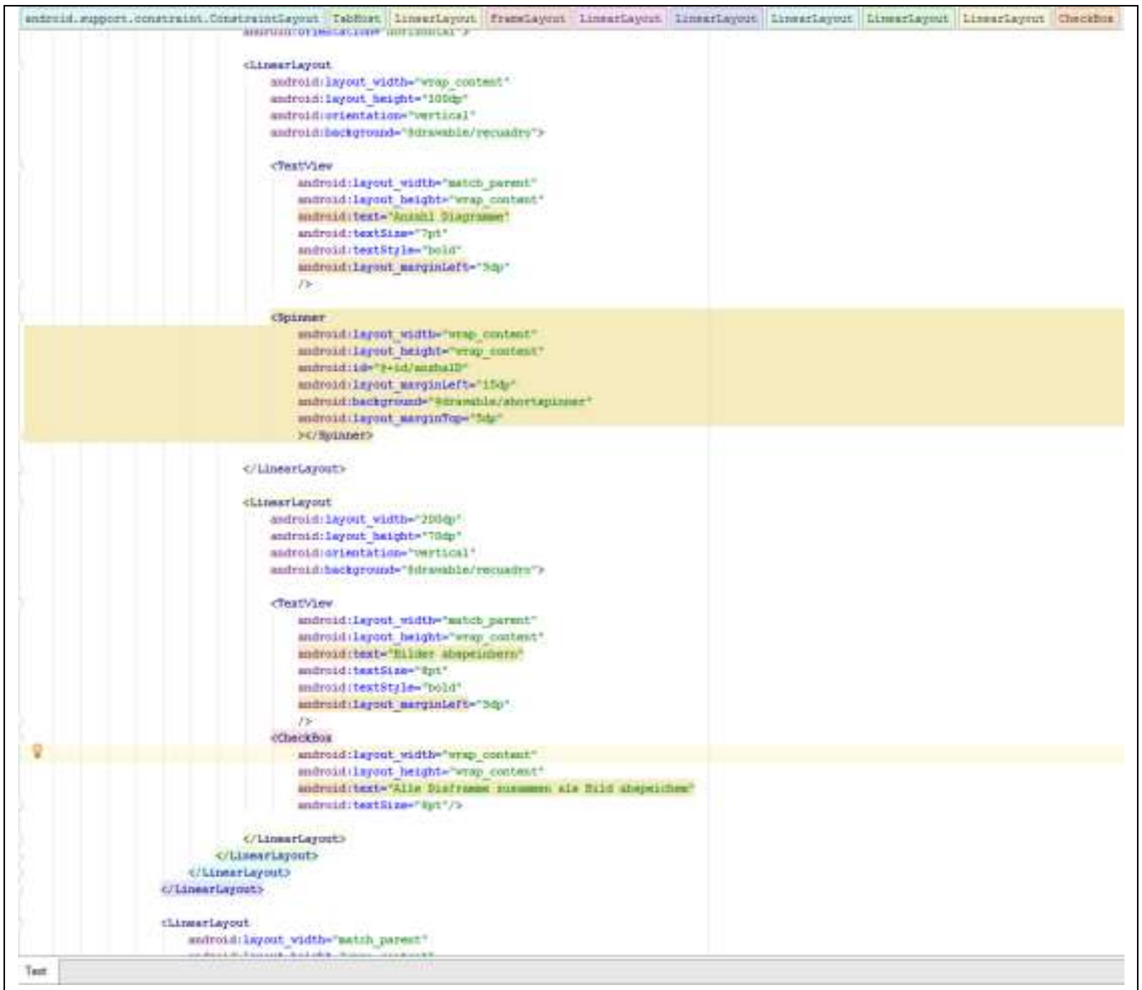
The design mode also offers the option of dragging from the palette window the components that we want to use for the activity. However, this mode is not useful on its own since the parameters of positions and sizes must be set by programming.

Also, to work with *TabHost* as shown in the previous image, it only allows the first window to be viewed on the computer screen, so it is impossible to see the rest of the windows and to be able to add any component in them.

On the other hand, text mode allows you greater freedom to place components, modify properties, positions, etc. To ensure that this is done well, it is important to create several linear layouts within each other. That is, if we have a box with 3 rows of 5 components, the best way to proceed is to do a vertical linear layout. Next we add 3 horizontal layouts, so that they will be one on top of the other, and then the 5 components or objects for each layout that as they are in a horizontal layout, they will be one next to each other. This makes a more organized and aesthetically elegant application.



Picture 15: Organized way to create a layout



```
android.support.constraint.ConstraintLayout  TabHost  LinearLayout  FrameLayout  LinearLayout  LinearLayout  LinearLayout  LinearLayout  LinearLayout  CheckBox
                        android:orientation="horizontal">

                        <LinearLayout
                            android:layout_width="wrap_content"
                            android:layout_height="100dp"
                            android:orientation="vertical"
                            android:background="@drawable/recuadro">

                            <TextView
                                android:layout_width="match_parent"
                                android:layout_height="wrap_content"
                                android:text="Anzahl Diagramme"
                                android:textSize="7pt"
                                android:textStyle="bold"
                                android:layout_marginLeft="5dp"
                                />

                            <Spinner
                                android:layout_width="wrap_content"
                                android:layout_height="wrap_content"
                                android:id="@+id/anzahlD"
                                android:layout_marginLeft="15dp"
                                android:background="@drawable/shortspinner"
                                android:layout_marginTop="5dp"
                                ></Spinner>

                        </LinearLayout>

                        <LinearLayout
                            android:layout_width="200dp"
                            android:layout_height="70dp"
                            android:orientation="vertical"
                            android:background="@drawable/recuadro">

                            <TextView
                                android:layout_width="match_parent"
                                android:layout_height="wrap_content"
                                android:text="Bilder abspeichern"
                                android:textSize="8pt"
                                android:textStyle="bold"
                                android:layout_marginLeft="5dp"
                                />
                            <CheckBox
                                android:layout_width="wrap_content"
                                android:layout_height="wrap_content"
                                android:text="Alle Diagramme zusammen als Bild abspeichern"
                                android:textSize="8pt"/>

                        </LinearLayout>
                    </LinearLayout>
                </LinearLayout>
            </LinearLayout>

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
Text
```

Picture 16: Text mode of layout creation

Making an application for different devices and screen sizes involves making different layouts that cover the needs of each of the systems to which we want to get with our application. This is because the app cannot automatically adapt to different resolutions and screen sizes.

There is a set of 6 generalized densities, grouped in *dpi* (dots per inch):

- ldpi (low density) ~ 120 dpi
- mdpi (medium density) ~ 160 dpi
- hdpi (high density) ~ 240 dpi
- xhdpi (extra high density) ~ 320 dpi
- xxhdpi (extra extra high density) ~ 480 dpi
- xxxhdpi (extra extra extra high density) ~ 640 dpi

However, more important than pixel density, for our application, it is to look especially at the screen size.

When designing the interface of an application for different screen sizes, we will verify that each type of size has a minimum amount of space, that it is, each group in which the screen sizes are divided are defined by the minimum spaces (*dp*). This allows the system to avoid taking into account changes in screen densities.

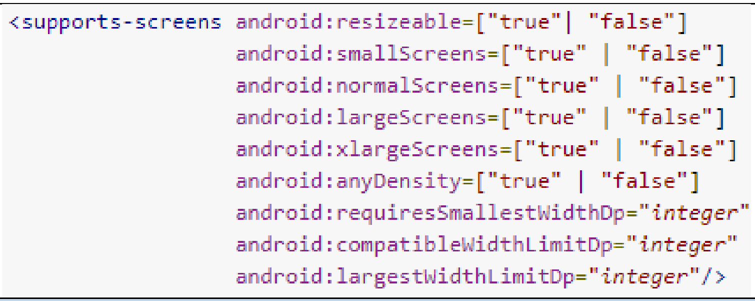
There are 4 different groups of screen sizes:

- Small screens that measure at least 426 x 320 dp.
- Medium screens that measure at least 470 x 320 dp.
- Large screens that measure at least 640 x 480 dp.
- Extra-large screens that measure at least 960 x 720 dp.

How to support multiple screens

Even that in this application has not been implemented, it is important to highlight the necessary way to combine the app with different screen sizes.

First, it is necessary to make a modification on the *Manifest* file so that it knows which screens and devices support the application we have made. This prevents the download of the app (in case it is upload on *Google Play*) on these Android systems that are not able to work with it. To do this, the element <*supports-screens*> is added in the *Manifest*.



```
<supports-screens android:resizeable=["true"| "false"]
                  android:smallScreens=["true" | "false"]
                  android:normalScreens=["true" | "false"]
                  android:largeScreens=["true" | "false"]
                  android:xlargeScreens=["true" | "false"]
                  android:anyDensity=["true" | "false"]
                  android:requiresSmallestWidthDp="integer"
                  android:compatibleWidthLimitDp="integer"
                  android:largestWidthLimitDp="integer"/>
```

Picture 17: *Manifest* code to add for supporting multiple screens

So, it is necessary to create a total of 4 layouts, one for each screen size. To the names of the layouts are added some qualifiers that allow the program to differentiate between the four categories of screens. These qualifiers are small, normal, large and xlarge. For example, designs for an extra-large screen, like the one our tablet has, should go into *layout-xlarge /*.

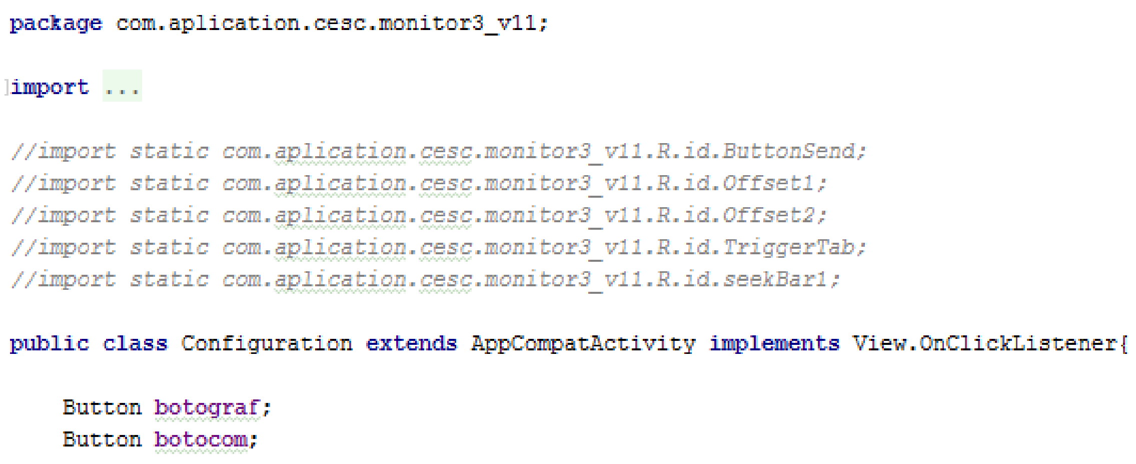
Taking advantage of these new interfaces by size, it is also a good practice to use visualization resources as icons or images that are inserted in different resolution qualities and bigger or smaller sizes. This allows a greater performance on smaller displays and higher quality on larger ones.

Java files

Java files are those that allow us to give functionality to the interface or layout that we have created.

In itself, these files are the engine of the app, the ones in charge of making it work through its code. In them the lifecycle, of which has been discussed previously in this report, is executed following the steps of that shown schematic. Before we get into the programming of the application, it is necessary to see in general terms how these files work.

Initially an import of the libraries and components as well as the definition of the variables, constants or other elements that will be used within the class of activity that we choose and the implementations its needs.



```
package com.aplication.cesc.monitor3_v11;

import ...

//import static com.aplication.cesc.monitor3_v11.R.id.ButtonSend;
//import static com.aplication.cesc.monitor3_v11.R.id.Offset1;
//import static com.aplication.cesc.monitor3_v11.R.id.Offset2;
//import static com.aplication.cesc.monitor3_v11.R.id.TriggerTab;
//import static com.aplication.cesc.monitor3_v11.R.id.seekBar1;

public class Configuration extends AppCompatActivity implements View.OnClickListener{

    Button botograf;
    Button botocom;
```

As we have been able to see in the section of *Activity Lifecycle*, when calling our activity the *onCreate( )* is originated.

It is in this place where the relationships between the components defined in the activity class and those created in the layout are made.

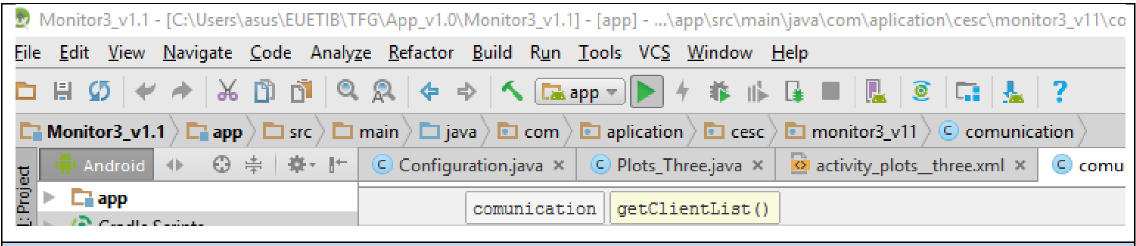
```
startShot = (Button) findViewById(R.id.startShot);
startShot.setOnClickListener(this);
```

In order to navigate between activities without losing information, we use elements called intents. These are action objects that invoke other activities leaving the current one that the user sees. In turn, we can add information, *putExtra( )*, to the intents we create so that we can then use it in the activity to which it is called.

```
Intent intent = new Intent().setClass(MainActivity.this,Configuration.class);
intent.putExtra("ok"," ");
intent.putExtra("punts",0);
intent.putExtra("punts2",0);
intent.putExtra("verif",false);
//intent.putExtra("numpoints",false);
intent.putExtra("RollShot",false);
intent.putExtra("triggS",0);
intent.putExtra("timeS",0);
intent.putExtra("triggR",0);
intent.putExtra("timeR",0);
startActivity(intent);
```

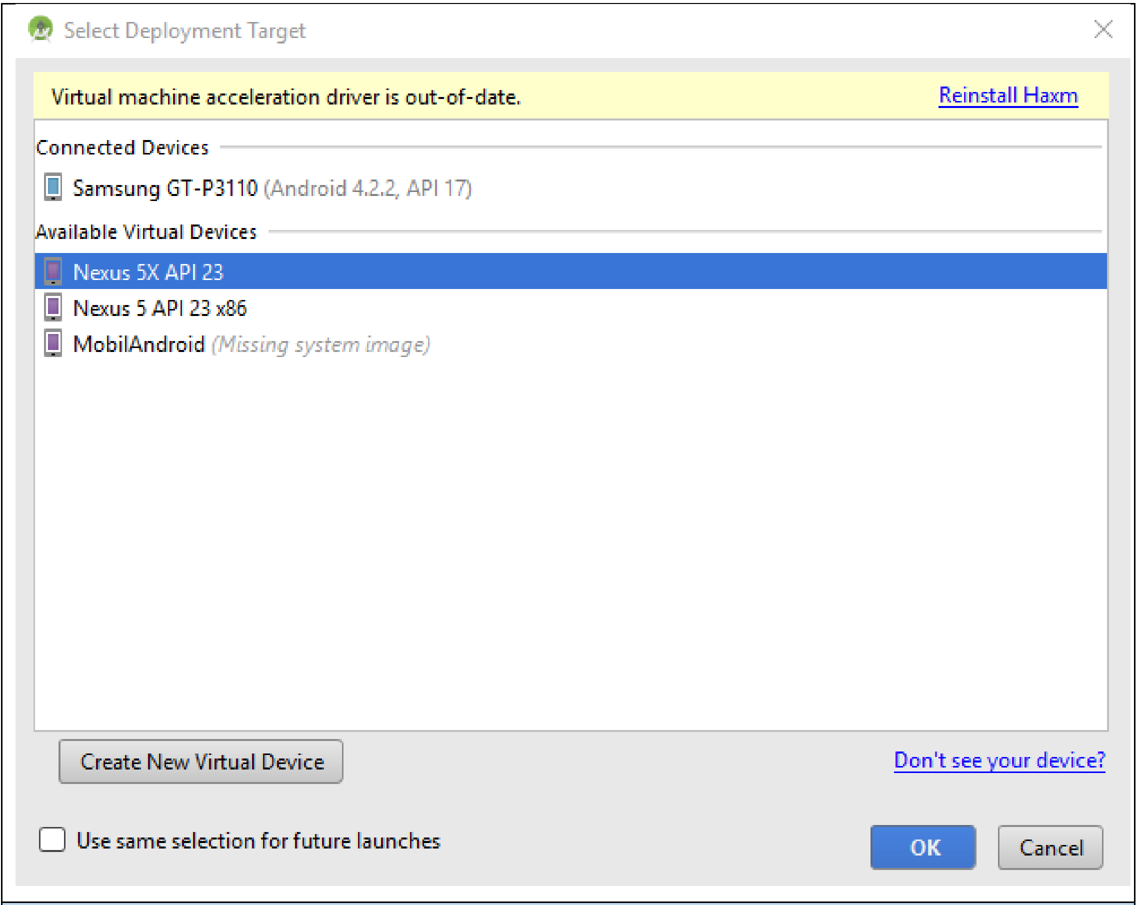
How to run an app

There are several ways to test the performance of the application on our *Android* devices. To do this with the tablet or mobile connected, press the run button in the top toolbar or type *Caps Lock + F10*.


Monitor3_v1.1 - [C:\Users\asus\EUETIB\TFG\App_v1.0\Monitor3_v1.1] - [app] - ...\app\src\main\java\com\aplication\cesc\monitor3_v11\co
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
app
Monitor3_v1.1 app src main java com aplication cesc monitor3_v11 comunication
Android
Configuration.java
Plots_Three.java
activity_plots_three.xml
comu
1: Project
app
comunication
getClientList()

Picture 18: Location of *run* button

Once pressed, a dialog box like the one shown below opens. In it, you can select between the available virtual devices (or emulators) and real ones that we have connected (as the Samsung that appears).


Select Deployment Target

Virtual machine acceleration driver is out-of-date. Reinstall Haxm

Connected Devices

Samsung GT-P3110 (Android 4.2.2, API 17)

Available Virtual Devices

Nexus 5X API 23

Nexus 5 API 23 x86

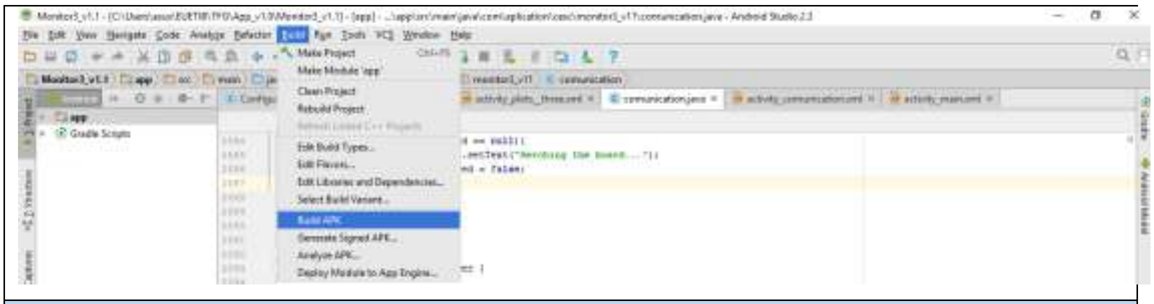
MobilAndroid *(Missing system image)*

Create New Virtual Device
Don't see your device?
Use same selection for future launches
OK
Cancel

Picture 19: Deployment Target window

If we select the real device, the application will be installed on it and it will run automatically, however, the emulator mode is another possibility to take in case of not having a real device. Unfortunately, it takes a little longer on execute and for an app as complex as the one developed in this project, it is not recommended to use it.

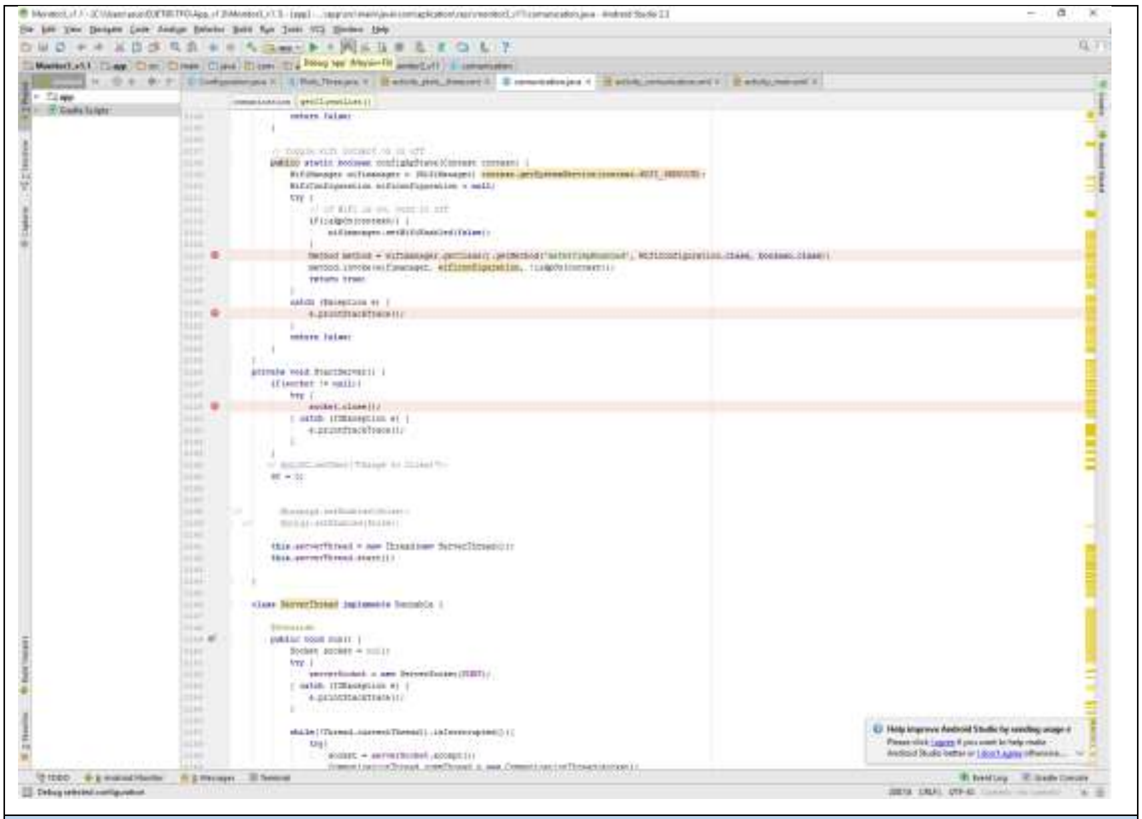
In case that for some reason we cannot connect the device through the *USB* to our computer, either because we do not have the cable or the *USB* port does not detect our system, there is always the possibility of performing a build generating an *.apk* file that we can send to our tablet or mobile through *Wi-Fi* and install it from them.



Picture 20: Location of *Build APK*

One of the most important tools when programming an application, is the use of the debug. Making an app to the first try and without any mistakes is practically impossible, for this, the best way to find the problems of our activities quickly, is to go step by step. Thus, red dots are placed on the left side of the code, next to the line number. These points will be the places where the app will stop when it gets on that line, if everything is correct, we can continue running the activity by pressing F9 until it stops at the next point.

The following image shows how the points are and what icon to use for the debug (we can also type *Caps Lock + F9*).



Picture 21: How points and debugging works

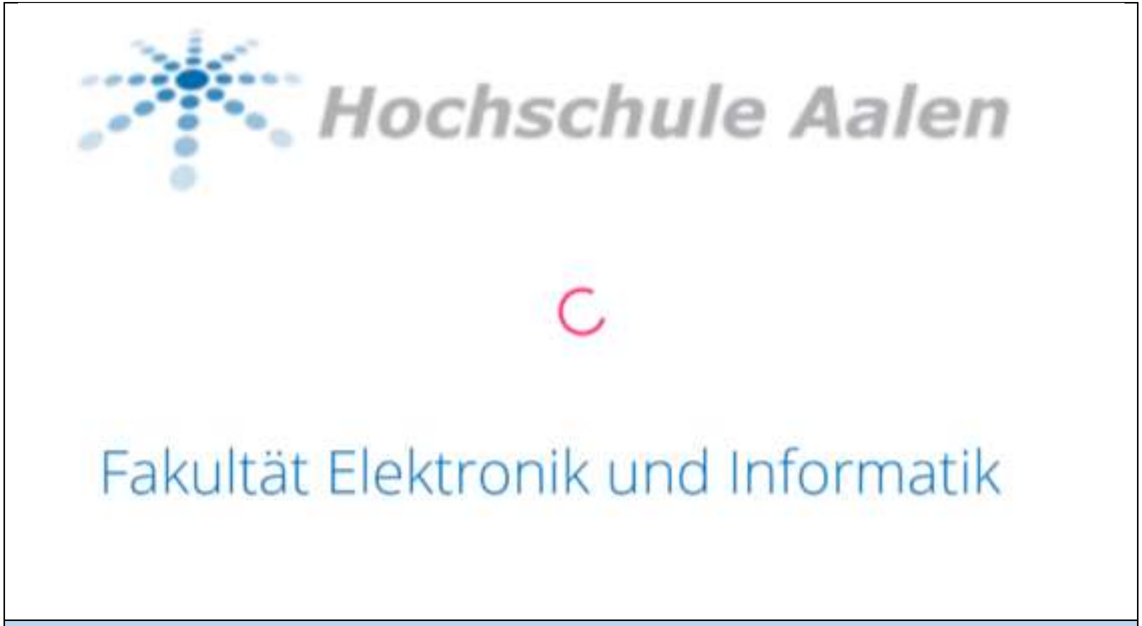
Activities of Monitor 3

Generally seen the layouts, the java and knowing what we must modify in the *Manifest* and in the *Gradle*, we can go to the explanation of *Monitor 3*. For this we will focus specifically on the java files, making some small mention to the *.xml*, where the components are.

This app is based on three main screens that allow the communication with the board, the configuration of the *Roll* or *Shot* and the activity which is responsible of plotting the information that receives creating the real time graphics.

MainActivity

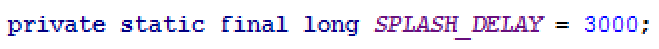
With this name generated by default this activity is in charge of performing the *Splash Screen* that we already talked in this report. It is made just to show the logo and name of the university as well as the faculty in which the application has been developed. It also incorporates a progress bar that, despite not being a bar is called equal because, it is responsible for implying that the *Application* is loading (or in progress).



Picture 22: *Splash Screen;* called *MainActivity* in the programm

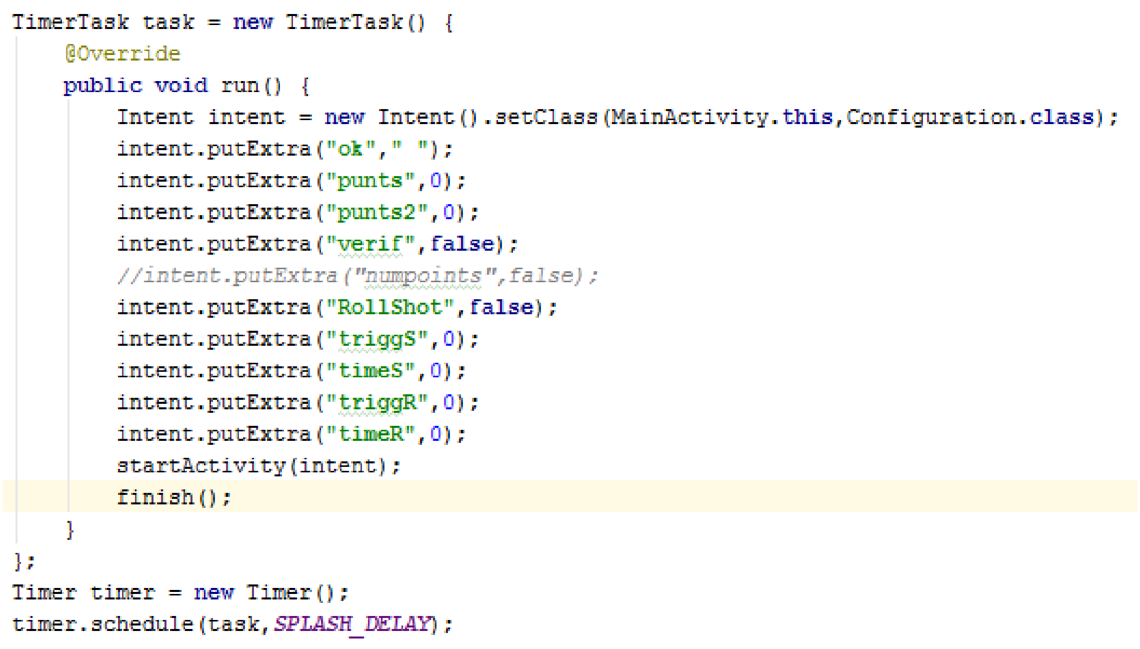
To obtain this screen has been used two *ImageView* containing the writings of *Hoschschule Aalen* and *Fakultät Electronik und Informatik*. The execution of the timeout that is used in this *Splash Screen* is achieved by using the so-called *TimerTask*.

This timer waits for a certain time that has been saved in the *onCreate( )* as *SPLASH_DELAY*. In this case it is 3 seconds, although it is perfectly variable according to our choice and can also be linked to a real process to know that the application is still working on it.



```
private static final long SPLASH_DELAY = 3000;
```

After this time, the Timer executes a type of function called run( ) and it just calls the main activity through an intent.

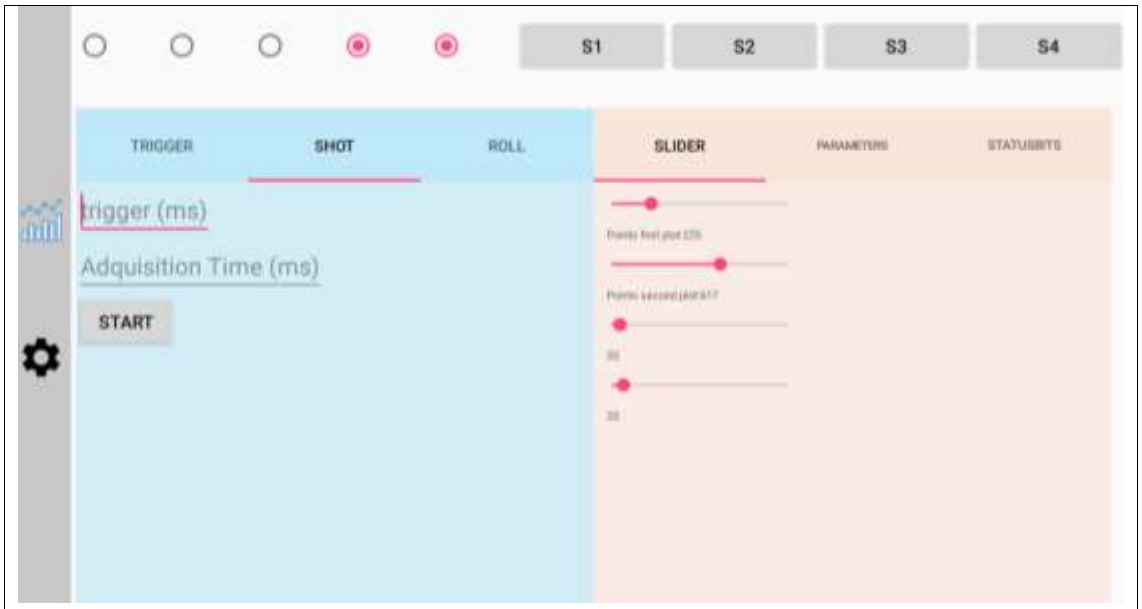


```
TimerTask task = new TimerTask() {
    @Override
    public void run() {
        Intent intent = new Intent().setClass(MainActivity.this,Configuration.class);
        intent.putExtra("ok"," ");
        intent.putExtra("punts",0);
        intent.putExtra("punts2",0);
        intent.putExtra("verif",false);
        //intent.putExtra("numpoints",false);
        intent.putExtra("RollShot",false);
        intent.putExtra("triggS",0);
        intent.putExtra("timeS",0);
        intent.putExtra("triggR",0);
        intent.putExtra("timeR",0);
        startActivity(intent);
        finish();
    }
};
Timer timer = new Timer();
timer.schedule(task,SPLASH_DELAY);
```

So the activity that the intent calls is *Configuration*, passing it the variables that can be seen in the previous image as putExtra. The value that appears next to these variables is a default one that is applied in case they have a null value. It is basically, *0* for integers or floats, *false* for the Booleans and "" for the strings.

Configuration

The activity called configuration is properly the main activity of our application. It consists of a bar with two buttons on the left, several "*LEDs*" (that actually are *RadioButtons*) some buttons up to the right and two *TabHost* in the middle that include three windows each. In this activity is which the configuration of the trigger and display parameters for the *Roll* and *Shot* are performed. It is also where the databits are activated and the *Sliders* are updated, although these two functions are not currently active.



Picture 23: Layout of the activity *Configuration*

The layout of this activity count with a total of 10 *RadioButtons*, 13 *Buttons*, 2 *TabHost*, 7 *EditText*, 4 *Sliders* and 7 *TextViews*.

Once defined all these components in the Java file, they are linked inside the *onCreate( )* function, as we have seen in the previous sections. It is also at this time when we collect the information, if any, from the previous activity. This is done through a *getIntent* that, as we can see, can get different variables.

```
Offset1 = getIntent().getExtras().getFloat("Offset1",0);
Offset2 = getIntent().getExtras().getFloat("Offset2",0);

grafcaract1 = getIntent().getExtras().getInt("GrafCaract1");
grafcaract2 = getIntent().getExtras().getInt("GrafCaract2");

IPBoard = getIntent().getExtras().getString("ipPlaca","");

BackButton = getIntent().getExtras().getBoolean("BakcButton",false);
```

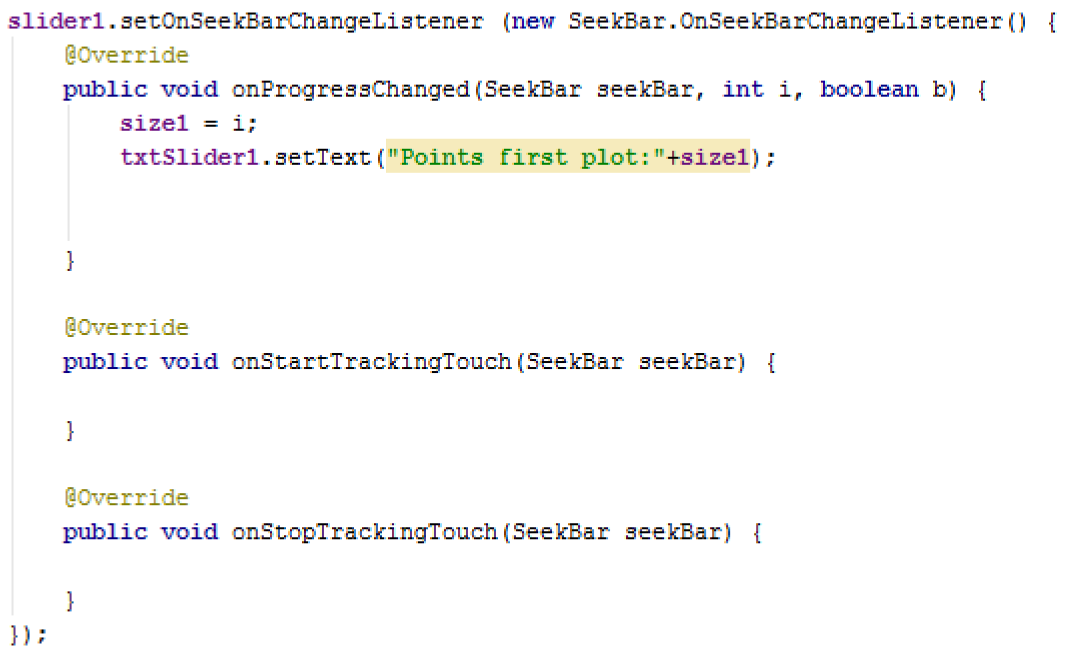
Unlike the Splash Screen, the configuration screen has to be listening to the different buttons and sliders which the user can interact with in the interface.

For the sliders, there is a particular method of *BarChangeListener* that generates three *Override* with different functions in each:

OnProgressChanged( ), is the general use one, causes a change in a variable associated to the slider that can go from a boolean change to an increase of floats passing through functions or other parts of the program if desired.

OnStartTrakingTouch( ) and *OnStopTrakingTouch( )* have a very similar function, however, in contrast to the previous one, these functions perform actions when the slider movement starts or when it is finished. Usually this type of functions is usually used for Booleans, to pause a thread or process while the user is interacting, for changes in the interface such as colors or shapes, etc.

For this app neither of the *OnStartTrakingTouch( )* or O*nStopTrakingTouch( )* were used, however *OnProgressChanged( )*, has been used to make the increment or decrement of an integer variable.



```
slider1.setOnSeekBarChangeListener (new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
        size1 = i;
        txtSlider1.setText("Points first plot:"+size1);


    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {

    }
});
```

Continuing with the listeners, we get to the buttons. These are programmed into a function, called *onClick( ),* which detects the button pressed by a *switch/case* method that contains the id of each of the linked buttons in the layout. At the end of the *switch* a default is added to prevent errors.

```
@Override
public void onClick(View view) {

    switch (view.getId()){

        case R.id.ButtonServerClient:
```

...

```
            }
            break;

        case R.id.ButtonConnect:
```

...

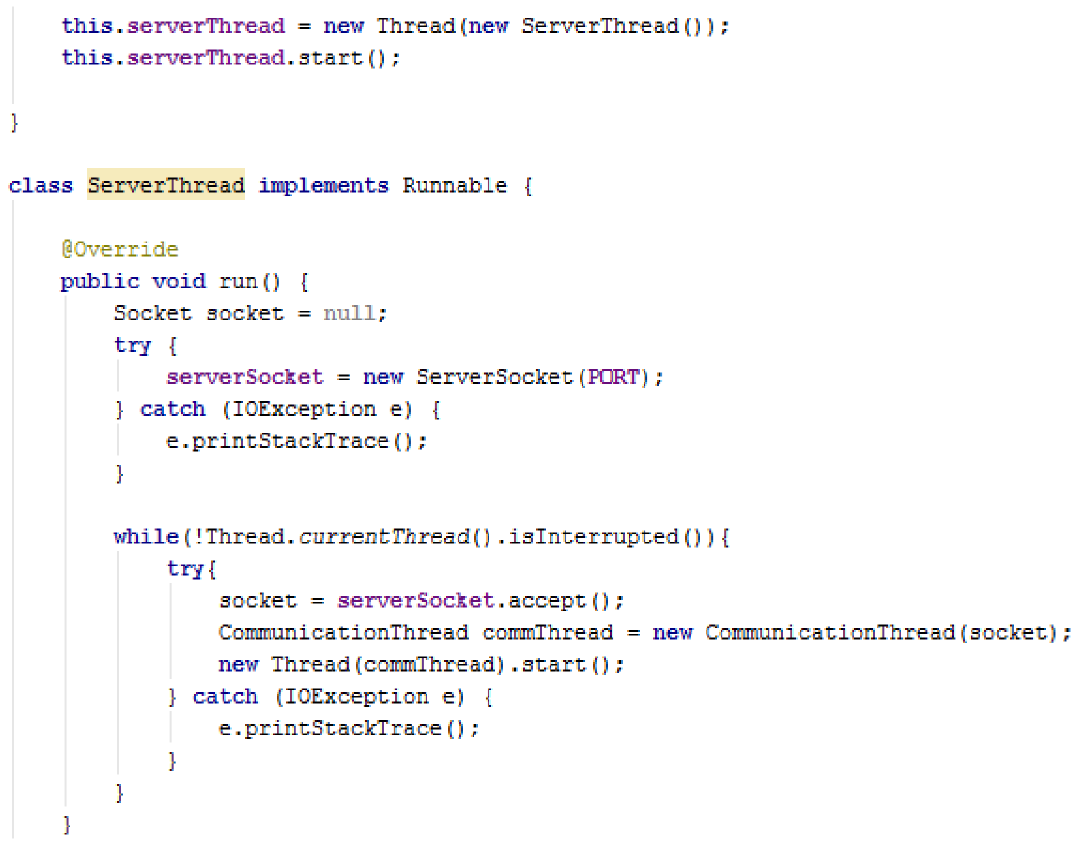
```
            break;

        default:
            break;
    }
```

After a process of information research it was decided, which was the most suitable method to communicate through *Wi-Fi*. This method is call *Sockets*. As it name says it basically consist on the use of sockets to send and receive information. In itself, sockets are a mechanism for delivering data packets from one device that are carried by using *TCP/IP* protocols to another.

To operate this method requires both parties to define the IP (except in server mode) and the port of the other. In this activity a client-server operating model has been developed, so that it can be able to send and receive without needing to change to another activity.

In client-server mode, it is the server that receives the data. Configured as a server, it is not able to send anything to the client with which it is linked to, for this, both must switch to the opposite mode in which they are. The process of change usually creates complications, a way to solve this, if you have several nodes with which you have to communicate, is to create a single server with the other devices as clients. Unlike servers, clients can send and receive information.

So, starting with the server mode, you create a thread that starts a runnable by creating the server and leaving it waiting for the acceptance of a client, *serverSocket.accept( )*. As the code below shows, these processes must be surrounded of *try/catch* to prevent a fatal error that can crash the app.



```
    this.serverThread = new Thread(new ServerThread());
    this.serverThread.start();

}

class ServerThread implements Runnable {

    @Override
    public void run() {
        Socket socket = null;
        try {
            serverSocket = new ServerSocket(PORT);
        } catch (IOException e) {
            e.printStackTrace();
        }

        while(!Thread.currentThread().isInterrupted()){
            try{
                socket = serverSocket.accept();
                CommunicationThread commThread = new CommunicationThread(socket);
                new Thread(commThread).start();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
```

In the same way the *CommunicationThread* and *updateUIThread* classes are thread, as the name implies, they generate a runnable expecting to receive an information socket on the port previously defined (*PORT*) and display the message in a *textView* (*MisRebut*). Before displaying the message it is made a check that the reading buffer is not empty, null, and a string is added that says *Client Says:*

```
public void run() {
    while(!Thread.currentThread().isInterrupted()){
        try{
            String read = input.readLine();
```

...

```
public void run() {
    if ((msg == null) || (msg == "null")){
    }
    else if ((msg != null) && (msg != "null")){
        if (config == false) {
            MisRebut.setText(MisRebut.getText().toString()+"Client Says: "+msg + "\n");
```

The server is ready to receive as soon as the program calls it. So, we can proceed with the client who should send the sockets with information that among it, will contain the *Roll* and *Shot* configurations.

The client has a similar operation to the server; a thread is created that generates a runnable which tries to connect to the server using the *try/catch* method. It must be taken into account that in order to use the client we must know the IP of the server to be able to connect to it (besides using the same port).

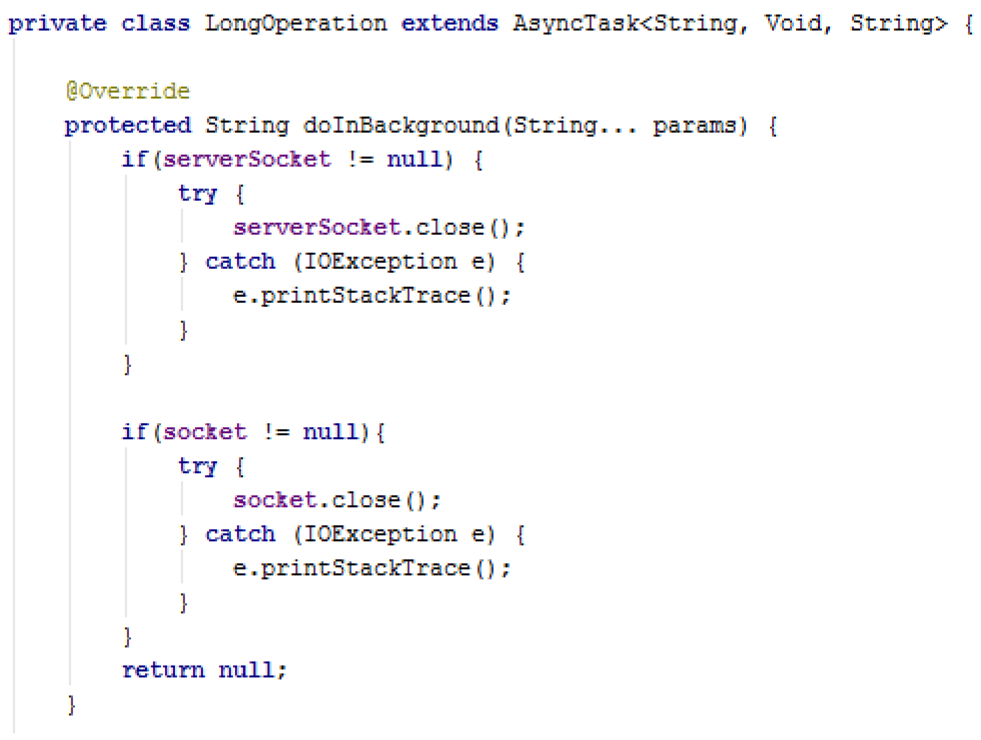
```
public void run() {
    try {
        InetAddress serverAddr = InetAddress.getByName(IPBoard);
        socket = new Socket(serverAddr, PORT);
```

The process of sending data is done by generating a *bufferwriter* to which the data is added. These data, which are integers from the *editText*, must be previously converted to string using *Integer.toString (Variable).*

Subsequently create the already commented buffer in which that variable is written and the output is generated in the socket that we have defined.

```
String tmR;
tmR = Integer.toString(IntTimeRoll);
PrintWriter out2 = new PrintWriter(new BufferedWriter(
        new OutputStreamWriter(socket.getOutputStream())),true);
out2.println(tmR);
```

It is necessary to know, that to change between server and clients a closing of the previous socket is necessary. To do this, the *socket.close( )* or *serverSocket*.close( ) functions are used. In addition, it is possible to ensure the correct operation of such operations by performing them asynchronously in the background of the application. In this case, an *AsyncTask* has been created that allows to realize it as we just described. These background operations allow the application to continue with the visible processes it has, at the same time it is able to make the sockets close.

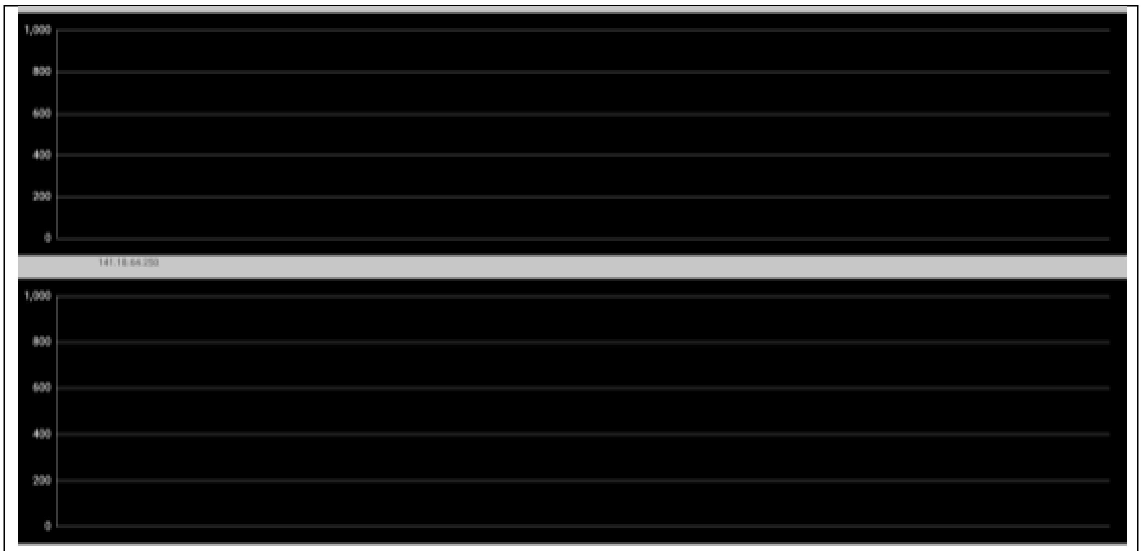


```java
private class LongOperation extends AsyncTask<String, Void, String> {

    @Override
    protected String doInBackground(String... params) {
        if(serverSocket != null) {
            try {
                serverSocket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        if(socket != null){
            try {
                socket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return null;
    }
```

Plots_Three

Plots_Three is the activity that shows what is happening in the motor which is being controlled. In this activity, unlike in the previous two, only the server is needed in order to receive the data that the board is sending.

That is why there is no client in the program, despite this, it may be necessary if we want to send a confirmation message to the board every time the information arrive. In that case, by adding the client as used in the *Configuration* or *Communication* screens, bidirectional communication can be performed.



Pictura 24: Interface of *Plots_Three* activity

The layout of this activity consists of two *MPAndroidChart* charts. This library contains a large number of different graphs, linear, scatter, circular, bar, etc. That is why it is necessary to specify in the *.xml* file that those used for this program must be linear. For this, they have been defined as follows:

```
<com.github.mikephil.charting.charts.LineChart
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/chart1_3"
    app:layout_constraintTop_toBottomOf="parent"
    android:layout_marginTop="40dp"
    />
```

Returning to Java programming, after defining the variables and others as explained above, a set of the main parameters of the graphs is made:

```
mChart = (LineChart) findViewById(R.id.chart1_3);
mChart2 = (LineChart) findViewById(R.id.chart2_3);
mChart.setOnChartGestureListener(this);
mChart2.setOnChartGestureListener(this);
mChart.setOnChartValueSelectedListener(this);
mChart2.setOnChartValueSelectedListener(this);
```

Obviously the set continues, it is necessary to define whether or not description is desired in the graph, to make it possible to be *touchable*, if it is allowed to move, to scale, to make double click to increase the view, etc. All these references are in the annex with the code and properly commented in the program.

In this activity once the *getIntents* are done, a selection of different functions is made to know where the application is, that means, if it must work like *Roll*, *Shot*, *Re-Roll*, *Re-Shot* or none of these. The first two are *Roll* and *Shot* in their traditional mode, receive and plot. Those that carry the prefix *re-*, are used at the time of leaving *Plots* and re-enter from the main activity by means of the button of *Graphs*.

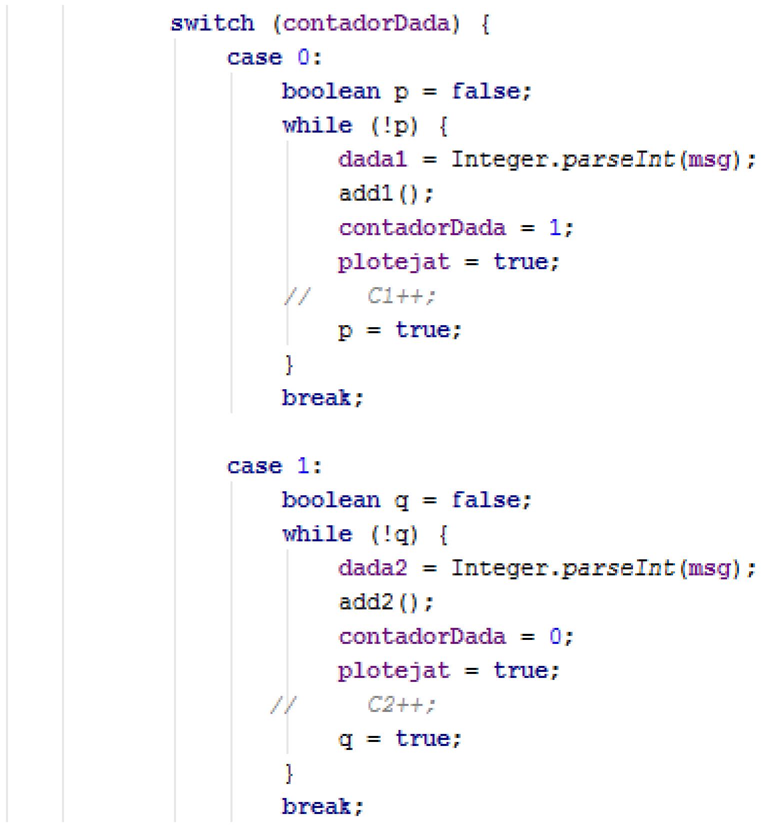
Then what happens is that it is re-graphed what was previously made and so the user can see it again (in that case no server is started and if you want to receive new data it is necessary to re-establish the values in the *Configuration* activity).

Since the previous activity has already explained how the *serverSocket* works, we will lighten the explanation by going directly to the moment when a data is received in the port of the tablet.

Within the function *run( )*, the reading of the message has been programmed into a *switch* with a variable that works alternating between 0 and 1. This allows one data to go to first graph and the next to second one, being this perfectly modifiable in case of wanting to receive in groups of values instead of one by one.

Once inside, the string is converted to integer (although it can also be done with floats) and the *add( )* function is accessed. Initially a graphing of the values was done as they were received by the port of the app. However, the process of obtaining, selecting and sampling was slower than the arrival of values, resulting in a sampling that grouped several values at once rather than plot one at a time.

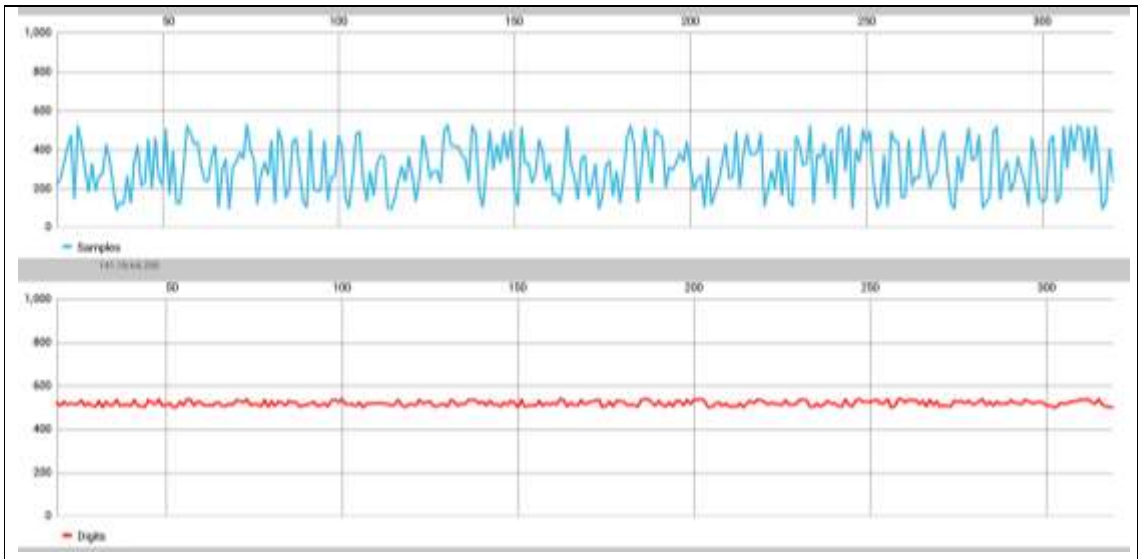
It is for this reason that a *while* function were added to the inside the *add( )* one, to force the program to first show the point on the graph before collecting the next one. Unfortunately this method is not entirely effective since it causes an accumulation of information in the port generating a delay in the sampling that increase during the time it last.



```
switch (contadorDada) {
    case 0:
        boolean p = false;
        while (!p) {
            dada1 = Integer.parseInt(msg);
            add1();
            contadorDada = 1;
            plotejat = true;
        //    C1++;
            p = true;
        }
        break;

    case 1:
        boolean q = false;
        while (!q) {
            dada2 = Integer.parseInt(msg);
            add2();
            contadorDada = 0;
            plotejat = true;
       //     C2++;
            q = true;
        }
        break;
```

In this function, it is checked if you are working in roll mode or shot mode. In the first case, the data is already taken as an integer and plotted. In the second, another check is made, looking at whether or not it has collected as much data as it should in a shot.

For example, if a 100ms trigger is performed, and we want to see on the screen the data that has been obtained every 500ms, the number of points to accumulate before displaying is 5. If it has not yet reached that number of points, the program will go to the next value of the port, if it fulfills this condition, it enters a loop that graphs all the accumulated values.

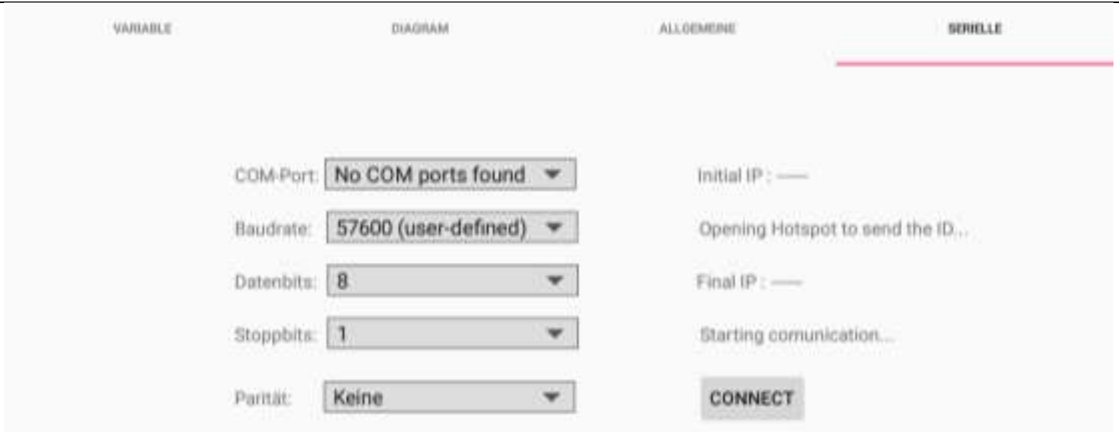


Picture 25: How *Plots_Three* activity works

Communication

The communication activity consists of a single *TabHost* with four windows as it can be seen in the lower images.

In total the layout of this activity consists of 38 *TextViews*, 16 *Spinners*, 7 *ImageViews*, 19 *Buttons*, 9 *CheckButtons* and 16 *EditText*. The activity is responsible for the initial communication between the two devices that must be connected, from the *HotSpot*, to the arrival of the IP of the board with which information is transferred from the microcontroller.



COM-Port: No COM ports found
Baudrate: 57600 (user-defined)
Datenbits: 8
Stoppbits: 1
Parität: Keine

Initial IP : ---
Opening Hotspot to send the ID...
Final IP : ---
Starting comunication...

Picture 26: *Communicaton – Seriell* window

To make such communication we must select the last of these windows, *Serielle*; is responsible of making the communication. To do this, when the user presses connect, the activity begins by reading the IP of the *Wi-Fi* network in which it is initially connected.

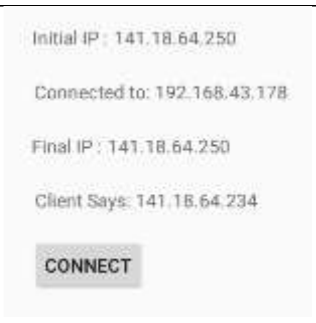
Once saved that address, it generates the *HotSpot* disconnecting itself from the network and waiting for someone to connect and enter into the same port.

```
private void Hotspot() {

    while(!BoardDetected) {
        //------------------------------HotSpot---------------------------------
        WifiAccessManager.setWifiApState(this, true); //creates the HotSpot
        //   Configuracio_placa();
        getClientList(); //Print the IP of the client
        //-------------------------------------------------------------------------

    }
    StartServer();
    StartClient();
```

The program automatically detects the two moments: the connection of a device to the *HotSpot* and the entry to the same communication port. As soon as this last process occurs, the program changes the working mode of the tablet from server to client, being able then to send the values of its IP and of the network in which it was connected.

When this is done, the *HotSpot* is disabled and the activity is automatically connected to the same *Wi-Fi* it was connected to. Then re-read the IP to verify that it is the same as the initial one.

If these requirements are met, the activity is waiting to receive the IP of the board so that it can communicate anywhere in the app.



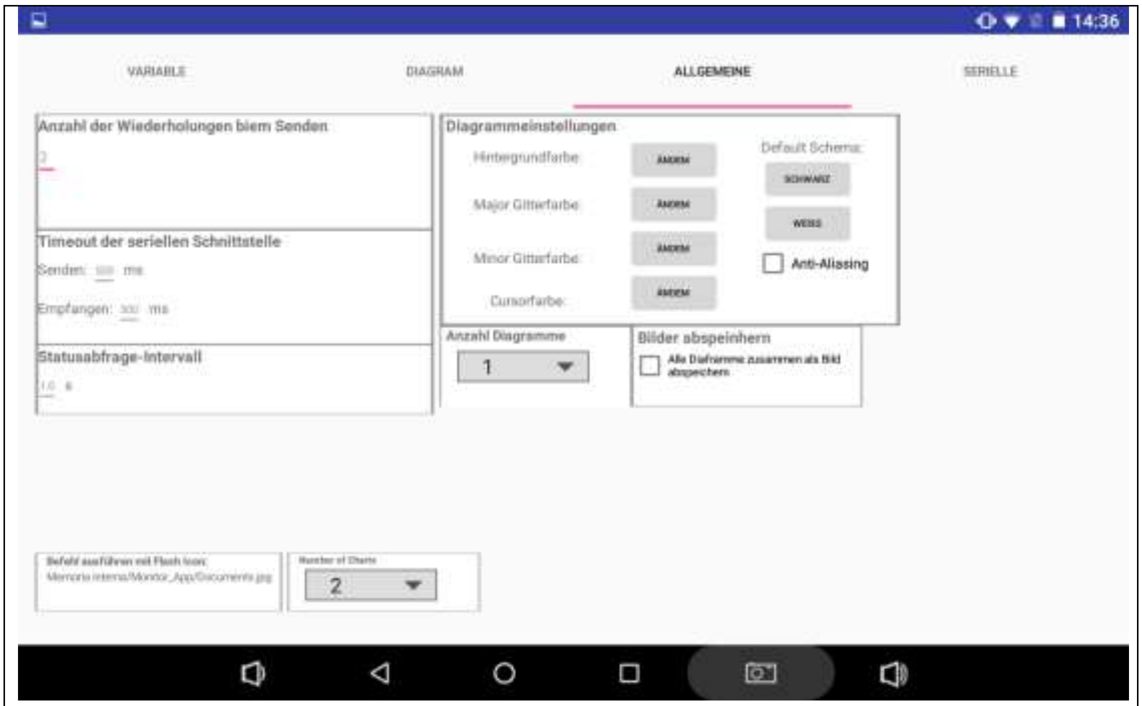
Initial IP : 141.18.64.250

Connected to: 192.168.43.178

Final IP : 141.18.64.250

Client Says: 141.18.64.234

Picture 27: Compared IP and received one



Anzahl der Wiederholungen biem Senden
2

Timeout der seriellen Schnittstelle
Senden: 100 ms
Empfangen: 300 ms

Statusabfrage-Intervall
1.0 s

Diagrammeinstellungen
Hintergrundfarbe:
Major Gitterfarbe:
Minor Gitterfarbe:
Cursorfarbe:
Default Schema:
☐ Anti-Aliasing

Anzahl Diagramme
1

Bilder abspeinhern
☐ Alle Diaframme zusammen als Bild abspeichern

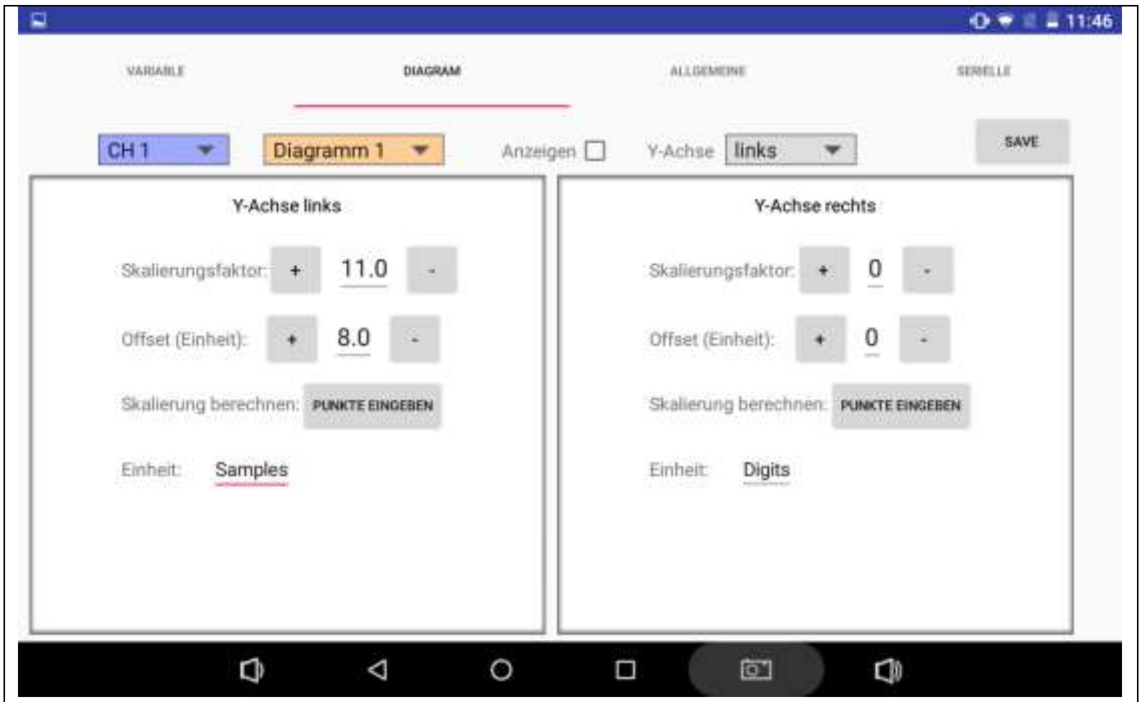
Befehl ausführen mit Flash Icon:
Memoria interna/Monitor_App/Documents.jpg

Number of Charts
2

Picture 28: *Communicaton – Allgemeine* window

The window *Diagram* is the one which allows the user to add offsets and/or increment the scale of one or both graphics that will appear in *Plots_Three*. It is also possible to change the units.

To make it work properly, it is necessary to select the second *Spinner Diagramm 1* or *Diagramm 2*, depending on whether we want to modify the first, or second graphic. Selecting one of them and giving them the scale values (*Skalierungsfaktor*), offset and the new units, we will press *SAVE* before changing to the following diagram. When we have finished the second one, we will save again to have registered the changes we want to make in the graphs.

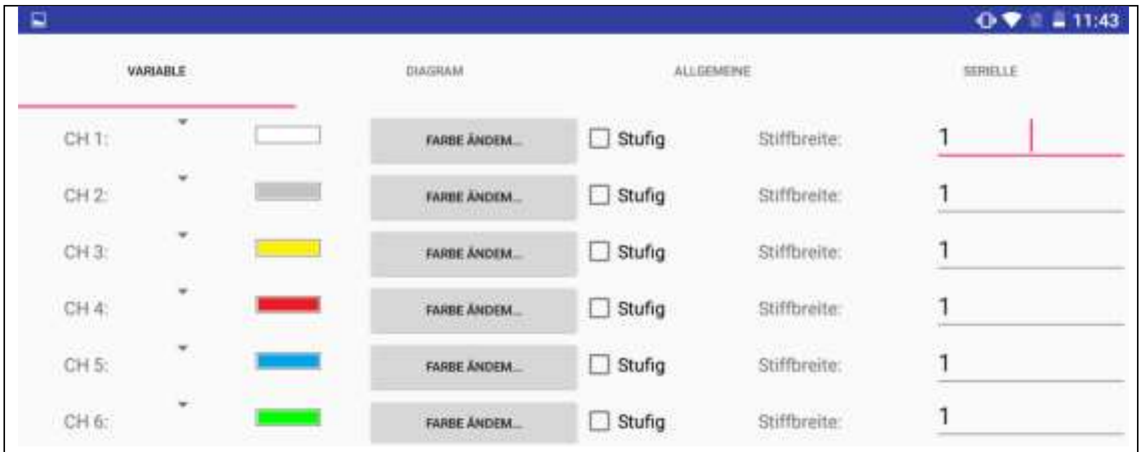


CH 1 | Diagramm 1 | Anzeigen ☐ | Y-Achse links

Y-Achse links				Y-Achse rechts			
Skalierungsfaktor:	+	11.0	-	Skalierungsfaktor:	+	0	-
Offset (Einheit):	+	8.0	-	Offset (Einheit):	+	0	-
Skalierung berechnen:	PUNKTE EINGEBEN			Skalierung berechnen:	PUNKTE EINGEBEN		
Einheit:	Samples			Einheit:	Digits		

Picture 29: *Communicaton – Diagram* window

Finally the *Variable* window does not yet have a specific job function.



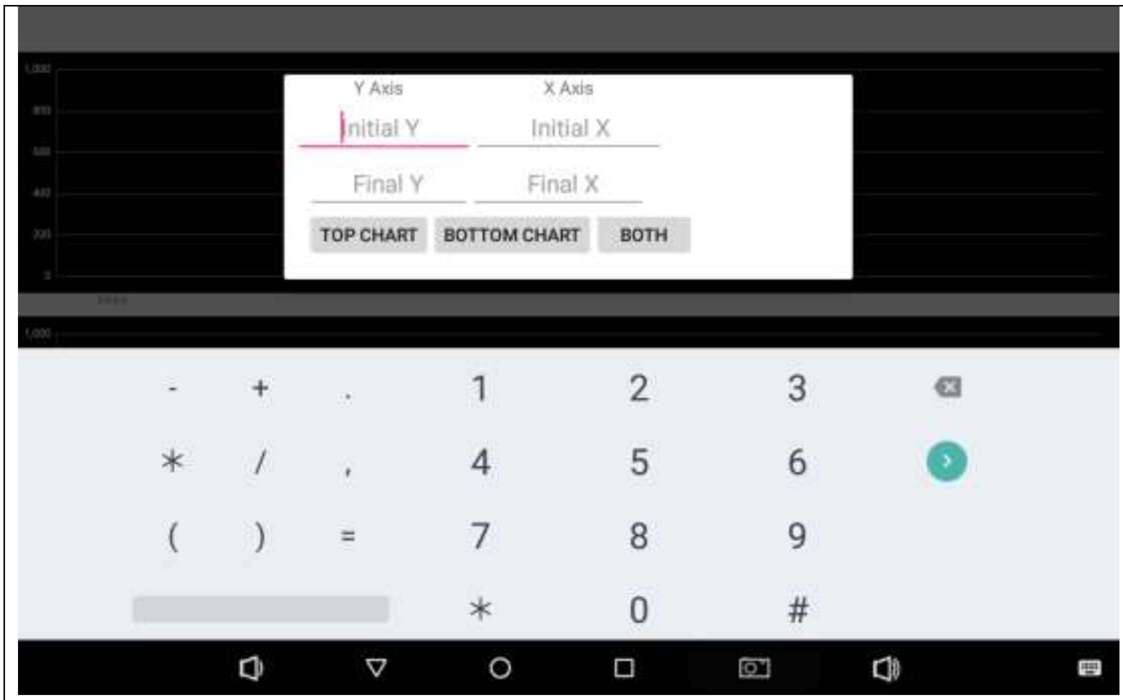
CH 1:	⬜	FARBE ÄNDEM...	☐ Stufig	Stiffbreite:	1
CH 2:	⬜	FARBE ÄNDEM...	☐ Stufig	Stiffbreite:	1
CH 3:	🟨	FARBE ÄNDEM...	☐ Stufig	Stiffbreite:	1
CH 4:	🟥	FARBE ÄNDEM...	☐ Stufig	Stiffbreite:	1
CH 5:	🟦	FARBE ÄNDEM...	☐ Stufig	Stiffbreite:	1
CH 6:	🟩	FARBE ÄNDEM...	☐ Stufig	Stiffbreite:	1

Picture 30: *Communicaton – Variable* window

Other files

In the *Android Studio* program there is a layout and an activity, (and also its corresponding layout) that have not been mentioned in the previous sections.

The surplus layout is called *dialog_axis* and it is the one that allows to make precise zooms in the activity of *Plots_Three* when double clicking on one of the graphs. That is, this *xml* file is a dialog box that does not require any * *.java* file because it is already used by the *Plots_Three*.



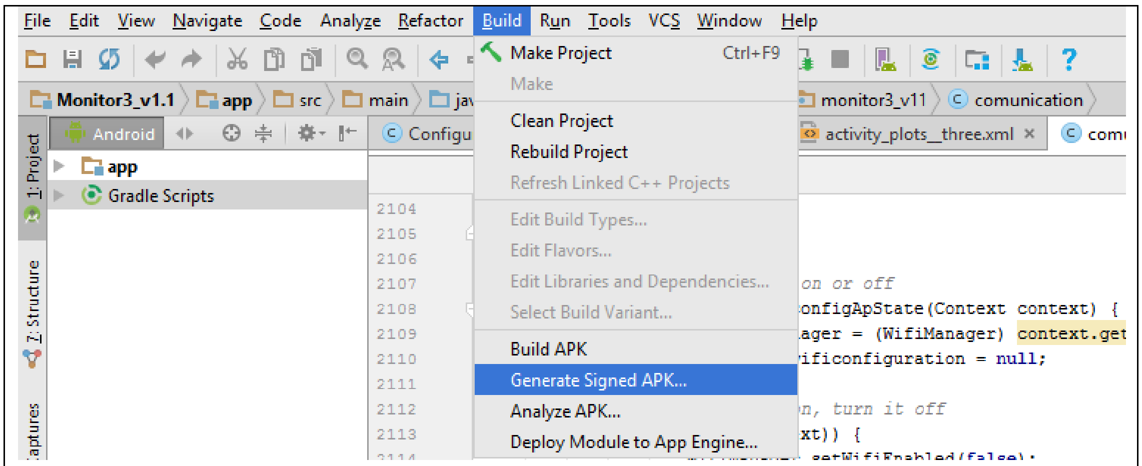
Picture 31: Dialog window when double tapping on one of the charts

The surplus activity is called *Plots*. This activity is the original that created the graphics but at the time of making *Roll* because of the thread the graphing was not correct. It has been decided to leave it to serve as a "*dirty paper*" for future changes if it is needed.

Upload to Google Play

Although the application has not been uploaded to the Internet, it is interesting to explain how the process works, because in the first place, there is something important to do in our *Android Studio* program; generate a signed *APK*.

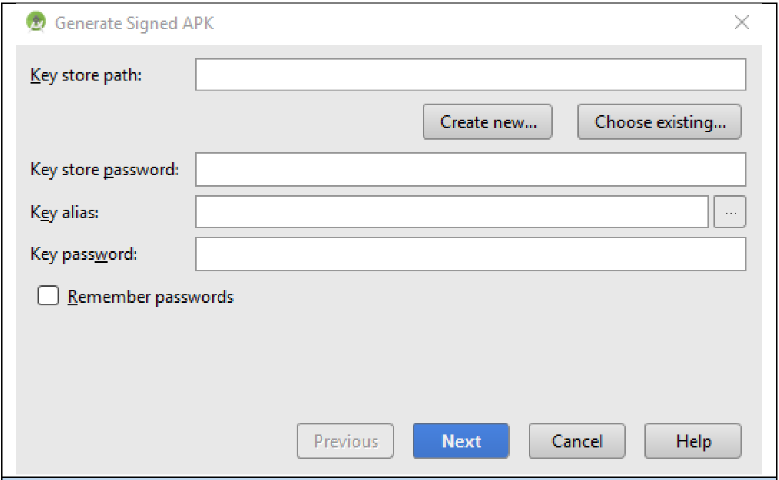
To do this, in the *Build* section, select the *Generate Signed APK* option.



```
on or off
onfigApState(Context context) {
ager = (WifiManager) context.get
ificonfiguration = null;

n, turn it off
xt)) {
```

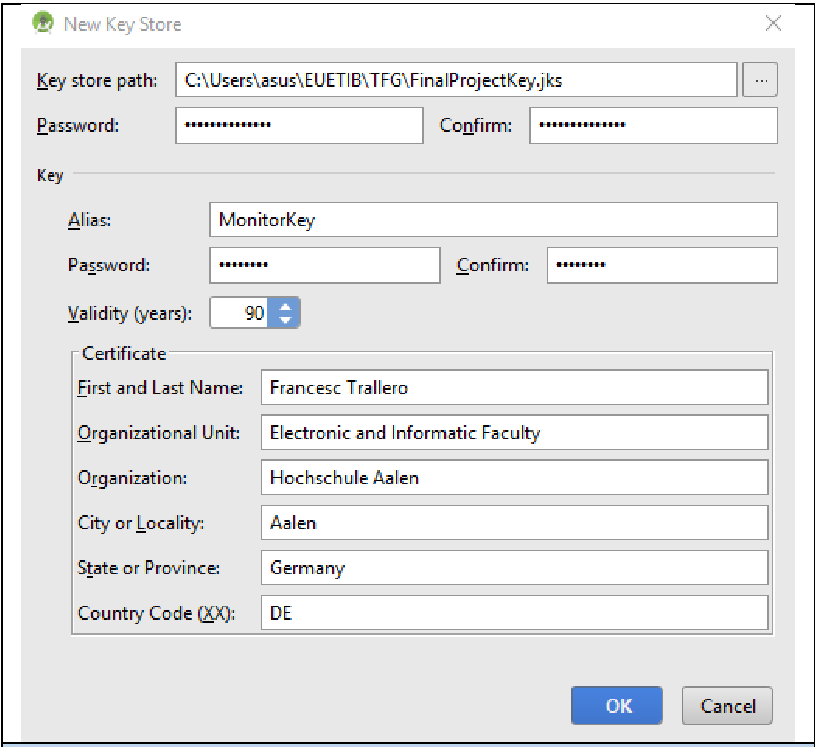
Picture 32: Location of *Build Signed APK*

Selecting this option will open a window where you will be asked for four keys to be able to generate the signed APK. If you have already created keys select *Choose existing…*, if not, we created them in *Create new…*.



Picture 33: Signed *APK* window

In case of creating a new one we will find a window identical to the one shown below. In it we will must fill the necessary fields to generate our new key.


New Key Store

Key store path: C:\Users\asus\EUETIB\TFG\FinalProjectKey.jks
Password: •••••••••••••• Confirm: ••••••••••••••

Key

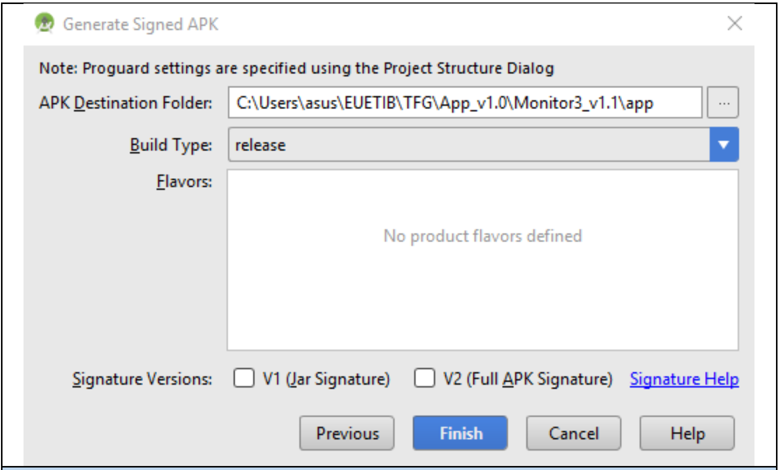
Alias: MonitorKey
Password: •••••••• Confirm: ••••••••
Validity (years): 90

Certificate

First and Last Name:	Francesc Trallero
Organizational Unit:	Electronic and Informatic Faculty
Organization:	Hochschule Aalen
City or Locality:	Aalen
State or Province:	Germany
Country Code (XX):	DE

Picture 34: Creating news keys

Finally specify the type of *Build* that we want to do and we can also choose the version of signature. Finished with the generation of the *APK*, we can start publishing it on *Google Play*.


Generate Signed APK

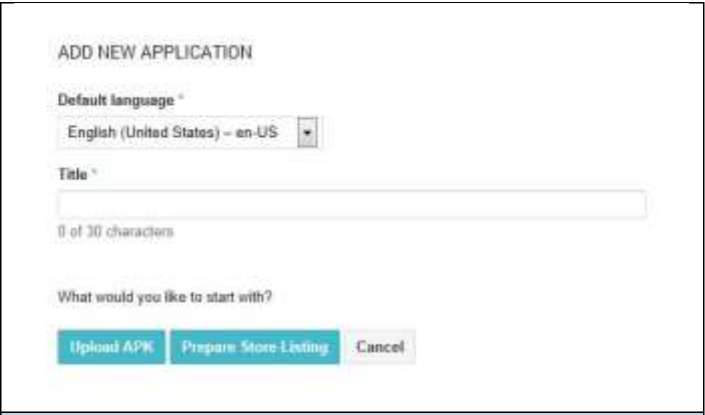
Note: Proguard settings are specified using the Project Structure Dialog
APK Destination Folder: C:\Users\asus\EUETIB\TFG\App_v1.0\Monitor3_v1.1\app
Build Type: release
Flavors: No product flavors defined
Signature Versions: V1 (Jar Signature) V2 (Full APK Signature) Signature Help

Picture 35: Build Signed *APK* last step

Once our *APK* file is generated, we will access the Google Play Developer Console, where a payment of 25(US $) must be made if it is our first time in uploading an app. When the payment is made, it is possible to access to the management and information center as developers. In this web we could see:

- Listing of our applications
- Services for Google Play Games
- Reports of our benefits
- Configuration
- Advertisements
- Alerts

To add a new application to *Google Play*, simply click on the *Add new application* button and it will make appear a dialog where for choosing the default language of the application and the title.


ADD NEW APPLICATION

Default language *

English (United States) – en-US

Title *

0 of 30 characters

What would you like to start with?

Picture 36: Add new app on *Google Play*

Clicking on *Upload APK* it will be able to upload the signed *APK* that has been made previously and we will be ready to fill in the *Store Listing*, where we will write the description, promotion text, upload screenshots, icon, category, etc.

In *Pricing & Distribution*, it is where the countries where the application will be available are chosen as well as whether it will be free or paid. Once this step is complete, it is be able to publish the app, removing it from the draft state.

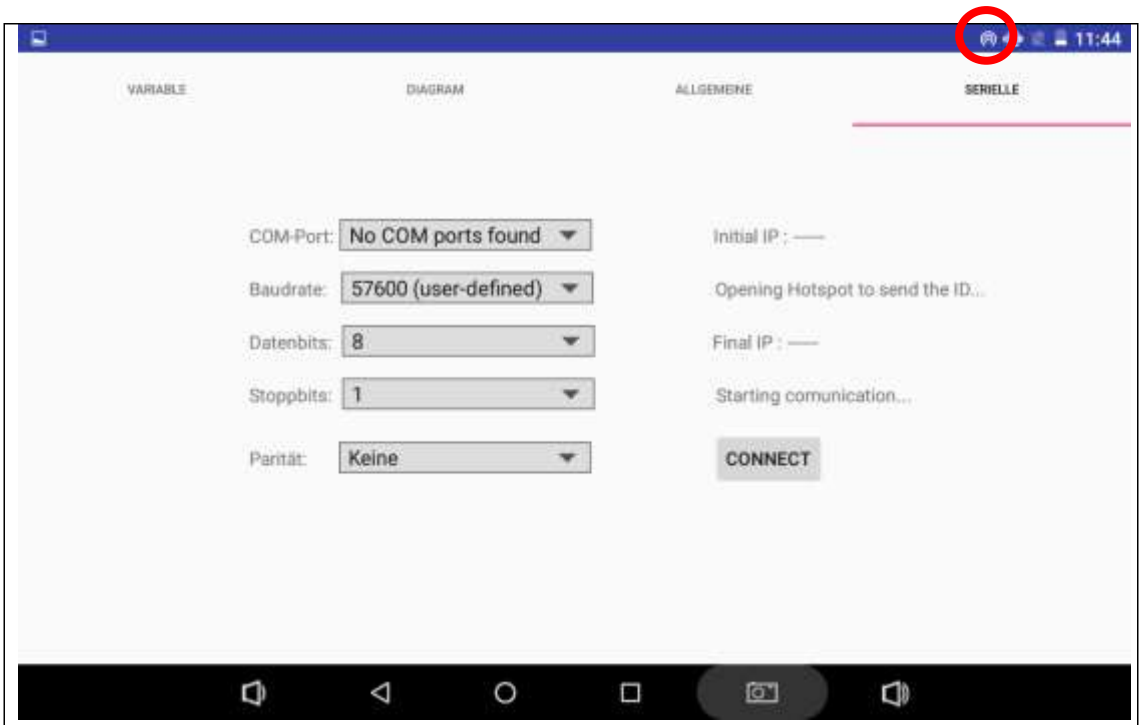
User guide

This section is intended to make a small step-by-step guide to run the application correctly. It is necessary to know that this application has been designed for later versions to Android 4.0 IceCream.

With *Wi-Fi* enabled, we launch the app. None of the roll or shot systems will run until the *HotSpot* has been made and the initial communication with the board has been made.

So we started making this initial communication. To do this, access the *Communication* screen using the icon on the left of the main activity. Once there, in the *Serielle* window press the *Connect* button.

If the process was successful, the *Wi-Fi* icon on the tablet should change to the *HotSpot* icon as shown in the following image, thus generating the access point so that the board can be connected. By default the IP that is given to the device when generating the HotSpot is 0.0.0.0.

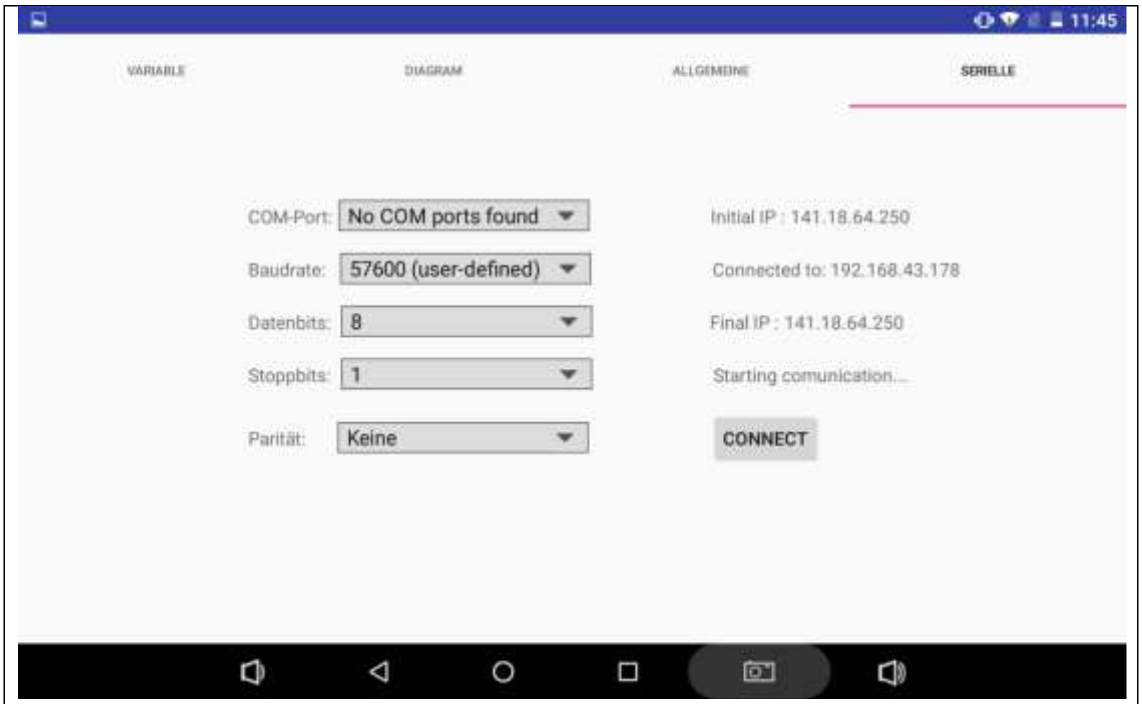


COM-Port: No COM ports found
Baudrate: 57600 (user-defined)
Datenbits: 8
Stoppbits: 1
Parität: Keine

Initial IP : —
Opening Hotspot to send the ID...
Final IP : —
Starting comunication...

Picture 37: Hotspot icon

Once the connection is made and the board is connected to the port (or the other tablet if a test it is being performed), the texts will be updated and the *Wi-Fi* will reconnect automatically.



COM-Port: No COM ports found
Baudrate: 57600 (user-defined)
Datenbits: 8
Stoppbits: 1
Parität: Keine

Initial IP : 141.18.64.250
Connected to: 192.168.43.178
Final IP : 141.18.64.250
Starting comunication...

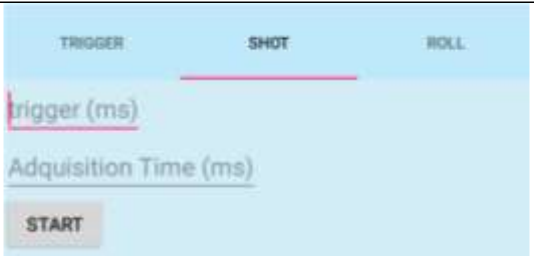
Picture 38: Compared information after port connection

On the screen there will be a *Starting comunication* ... until the activity has received the IP of the device with which it is communicating. Then you can make changes to the configurations parameters such as graphics background, offsets or scaling factors.

Pressing the back button ◁ of the *Android* operating system, located at the bottom of our app, it will return to the configuration screen.

On that screen functions of shot and roll are accecible. The value for each time a sample is requested (trigger) is added in the first *editText*. The second text depends on the graphing mode in which the user is located.

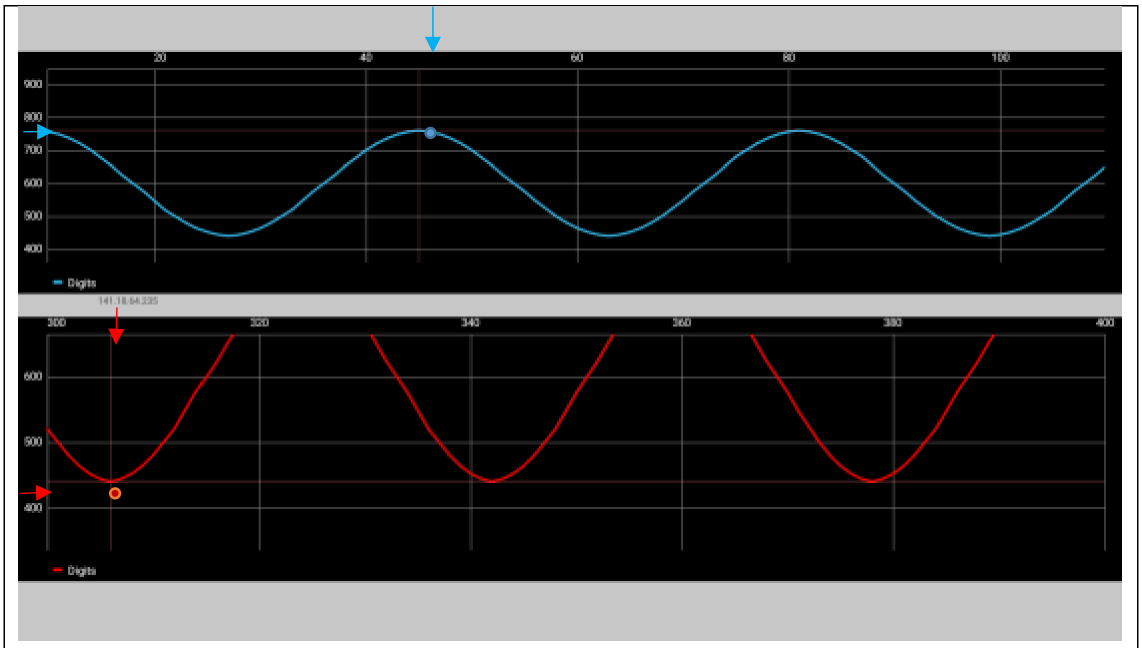
In Shot mode, the time in *ms* is added, this times is the one the user wants the graph to be refreshed and visualize the new values. On the other hand the *Roll* is added the number of points to be seen on the screen at the same time. If the number of points exceeds this value, the graph will move from right to left to allow the visualization of new ones without losing those that are no longer visible.



Picture 39: Shot window

In the Plots screen you can make interactions with the graphs. Double click on any of them opens a called dialog that allows making a specific zoom in one or both graphs. However, it is also possible to do this by moving the fingers on the graph. Placing both fingers together at the same point and separating them slowly expand the graphs, otherwise contract them to let more points appear.

Clicking on a point generates a horizontal line and a vertical line that allows to visualize with more clarity the value of that point.



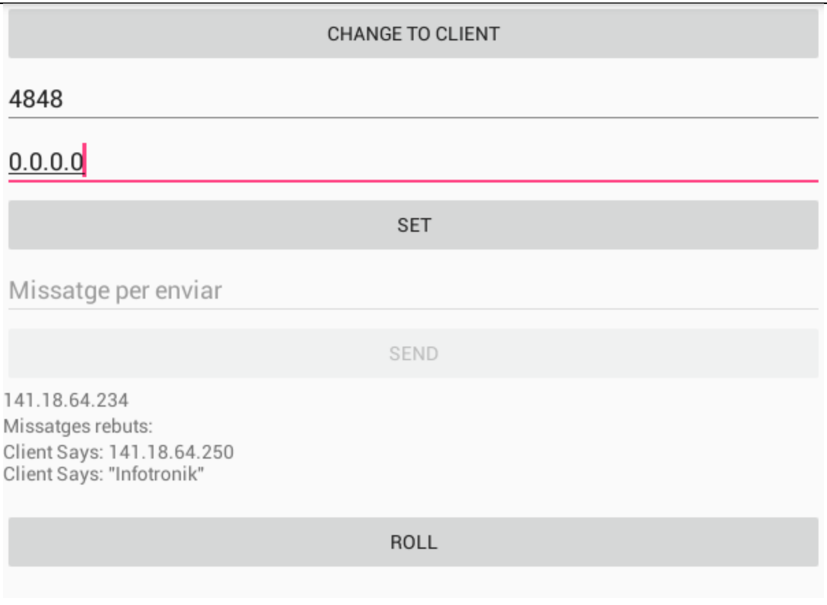
Picture 40: Lines of the clicked point (arrows and points are not visible in the app)

As in the *Communication* screen, by clicking the back button, you return to the main configuration screen, from where you can make the graph again or navigate to the screens to configure the *Plots* activity.

Auxiliary Application

Like configuration and communication activities, this app integrates the client and socket server systems. Thus, with the correct timing, the behavior of the board can be emulated.

So when the tablet that carries the Monitor creates the *HotSpot*, we will connect with the other and initiate the communication by writing the port (*4848*) and the IP (*0.0.0.0*). Then press *Set* and at the bottom we should receive the IP of the other system and the network to which it will be connected.



141.18.64.234
Missatges rebuts:
Client Says: 141.18.64.250
Client Says: "Infotronik"

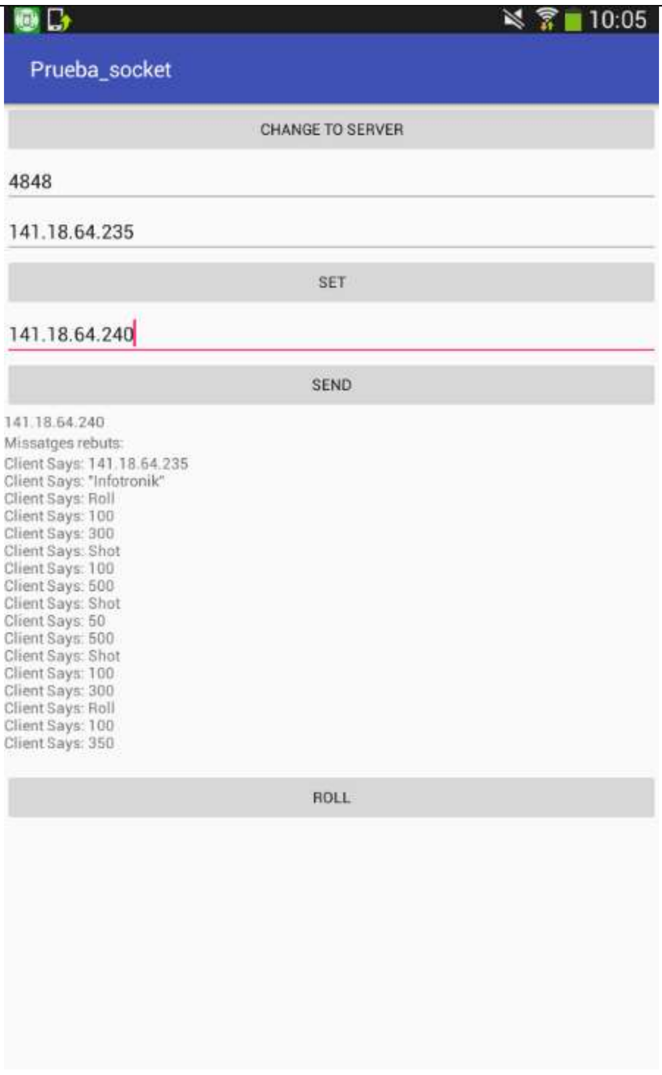
Picture 41: Information received from *Monitor 3*

Changing the IP that we had written (*0.0.0.0*) for the one we just received and pressing *SET* will start the communication with the tablet.

So the first message to send is our IP. To do this, pressing the button which says *CHANGE TO CLIENT* will change from server to client, leaving the *EditText* and *SEND* button enabled so that we can write and send the IP.

As soon as Monitor 3 has received the IP of the auxiliary tablet, we change to server by pressing *Change_to_server* in order to receive the data of *Roll* or *Shot* that the user wants to send.

When you have done it in *Monitor 3* the *Plots* screen will appear without any data, waiting for it to arrive. So there are two possibilities; it could be made a data-to-data sending if we change to client and in the *EditText* we write integers, or we can make a shipment followed by integers by clicking on *ROLL*. Thus we will start sending the data, with a velocity of 100ms each sample (fixed by the auxiliary app) and with a total of 500 samples for each graph.



4848

141.18.64.235

141.18.64.240

```
141.18.64.240
Missatges rebuts:
Client Says: 141.18.64.235
Client Says: "Infotronik"
Client Says: Roll
Client Says: 100
Client Says: 300
Client Says: Shot
Client Says: 100
Client Says: 500
Client Says: Shot
Client Says: 50
Client Says: 500
Client Says: Shot
Client Says: 100
Client Says: 300
Client Says: Roll
Client Says: 100
Client Says: 350
```

Picture 42: How the auxiliary app works

Applicable improvements

Port saturation

During the data acquisition process a delay occurs as the communication port (*4848*) fills up with too much information.

The way in which the app plots the data is by taking a point, checking if it is *shot* or *roll* and ploting it in case it must. During this process, several data continue to arrive at the port slowing down the process in which the points in the *Plots_Three* activity are plotted.

The buffer created that is responsible for collecting the information cannot be cleaned or reset. One of the options with which it could start working would be; try to close the buffer every time you collect a data and not reopen it until the graphing process is complete.

It is not known if the port would receive the same or not, initially to a blocked port the device that sends the information goes to "*standby mode*" until it is reopened, so data should not be lost.

Shot Loops

Probably because of code problems the number of points and also the value that *Shot* shows depends on the value we put in the acquisition time. This is most likely due to the loops in the process, however, it is necessary to debug to check when and why it happens.

Complete the uses of Configuration and Communication

During this project the main challenge in which the time has been dedicated to has been the communication and the graphing of the data, in other words, the main function of the monitor. Controlling and monitoring the motor is perfectly viable in the application that is now presented in this report, but there are many functions of Monitor 2 that have been implemented at the layout level but not developed in programming due to lack of time.

App optimization

The optimization of the application at the level of elimination of surplus code, comments, etc. It allows a better performance of this, it is advisable to do it once the app is complete, and this will help to improve the processing times and to avoid possible errors that did not appear in the first tests.

Disable buttons and editTexts

Making a more solid and mature application is the number one priority when it is perfectly functional and optimized. To do this, button locks, edits and other interface components must be made to make it more user-friendly.

This requires several tests with beta-testers; people who without any instruction are given the app for use. This allows you to see any failures or improvements that the programmer does not see because he is too familiar with the application. This process is most important creating an application for clients.

References

MPAndroidChart

Phil Jay: *MPAndroidChart Home Page* GitHub, Inc. [US], 2016, Checked March 2017 [*url*]: https://github.com/PhilJay/MPAndroidChart

Phil Jay: *MPAndroidChart Realtime Line Chart* GitHub, Inc. [US], 2016, Checked March 2017 [*url*]: https://github.com/PhilJay/MPAndroidChart/blob/master/MPChartExample/src/com/xxmassdeveloper/mpchartexample/RealtimeLineChartActivity.java

Numetric Technologies: *Android Line Chart using MPAndroidChart Tutorial*, January 2016, Checked March 2017 [*url*]: https://www.numetriclabz.com/android-line-chart-using-mpandroidchart-tutorial/

Developers (Webpage)

Developers: *Wi-Fi Peer-to-Peer*, Checked May 2017 [*url*]: https://developer.android.com/guide/topics/connectivity/wifip2p.html

Developers: *Maneja el cambio de configuración tú mismo,* [Spanish version] Checked May 2017 [*url*]: https://developer.android.com/guide/topics/resources/runtime-changes.html#HandlingTheChange

Developers: *Saving Persistent State*, Checked May 2017 [*url*]: https://developer.android.com/reference/android/app/Activity.html#SavingPersistentState

Developers: *Cómo volver a crear una actividad*, [Spanish version] Checked May 2017 [*url*]: https://developer.android.com/training/basics/activity-lifecycle/recreating.html

Developers: *Cuadro de diálogo* [Spanish version] Checked April 2017 [*url*]: https://developer.android.com/guide/topics/ui/dialogs.html?hl=es

Developers: *android.net.wifi* Checked May 2017 [*url*]: https://developer.android.com/reference/android/net/wifi/package-summary.html

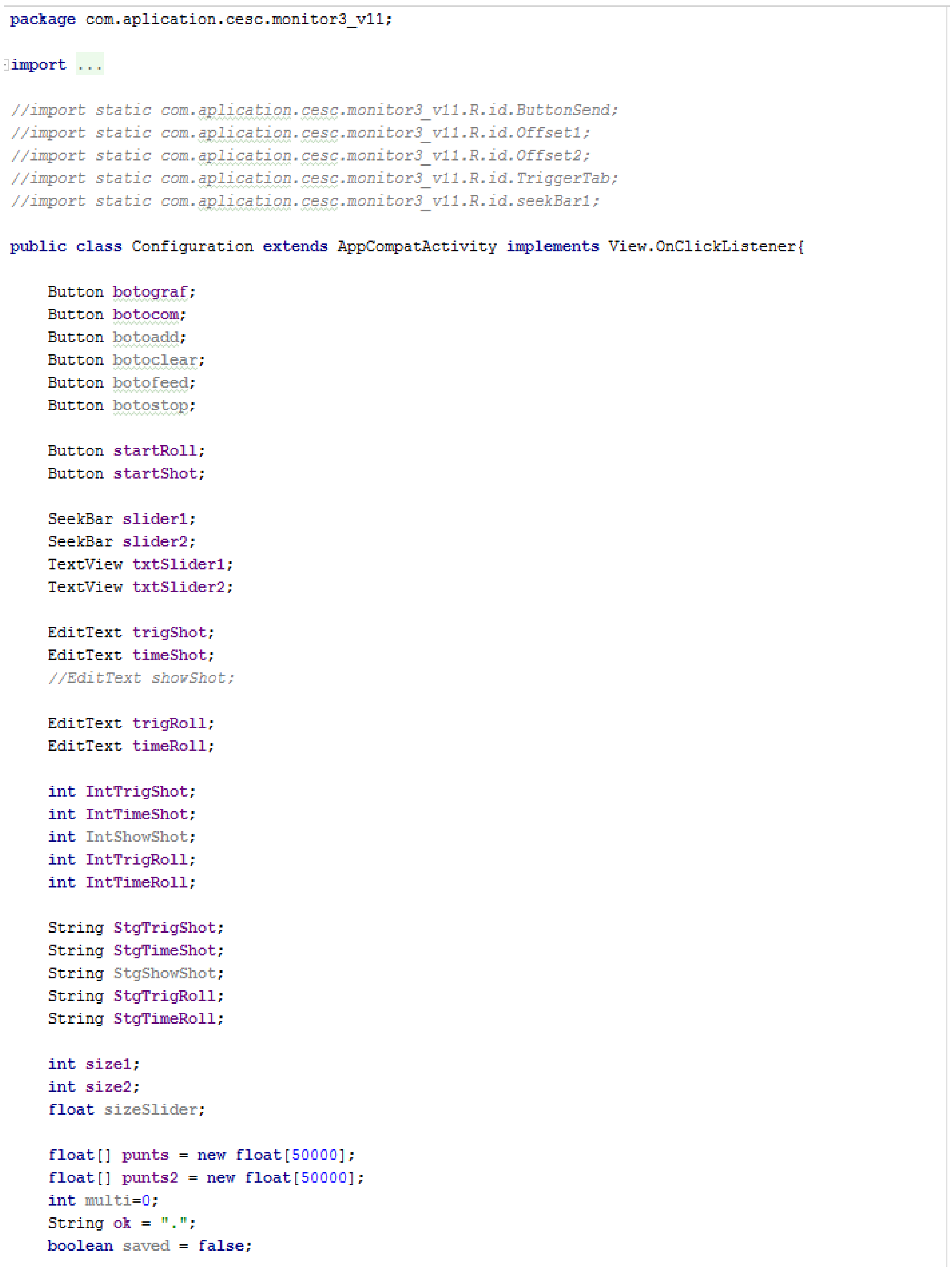
Developers: *Socket* Checked June 2017 [*url*]: https://developer.android.com/reference/java/net/Socket.html

Developers: *AsyncTask* Checked June 2017 [*url*]: https://developer.android.com/reference/android/os/AsyncTask.html

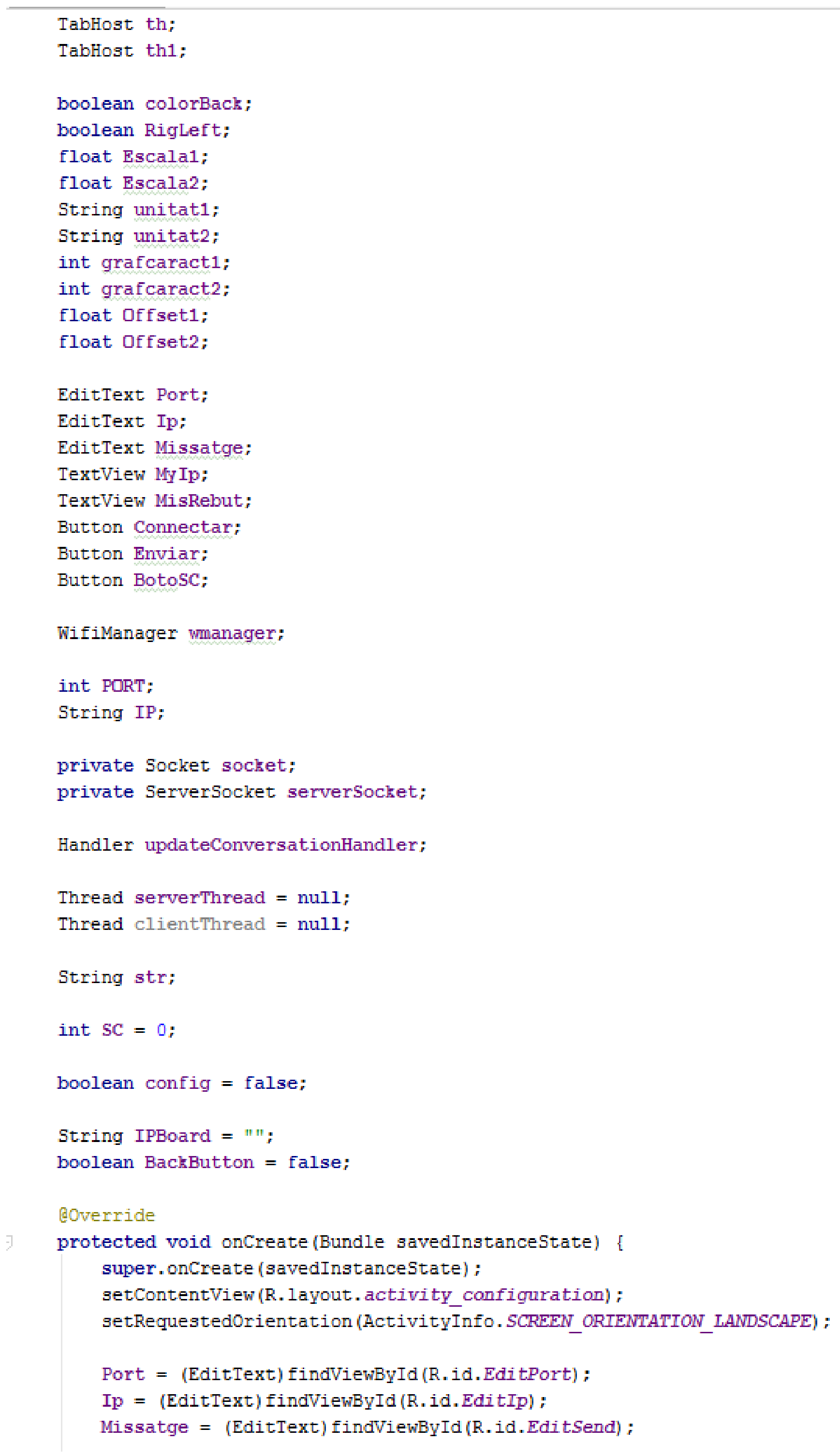
Annex - Code

This section details the code of the three main Java files to record and facilitate access to it should you need it.

Configuration

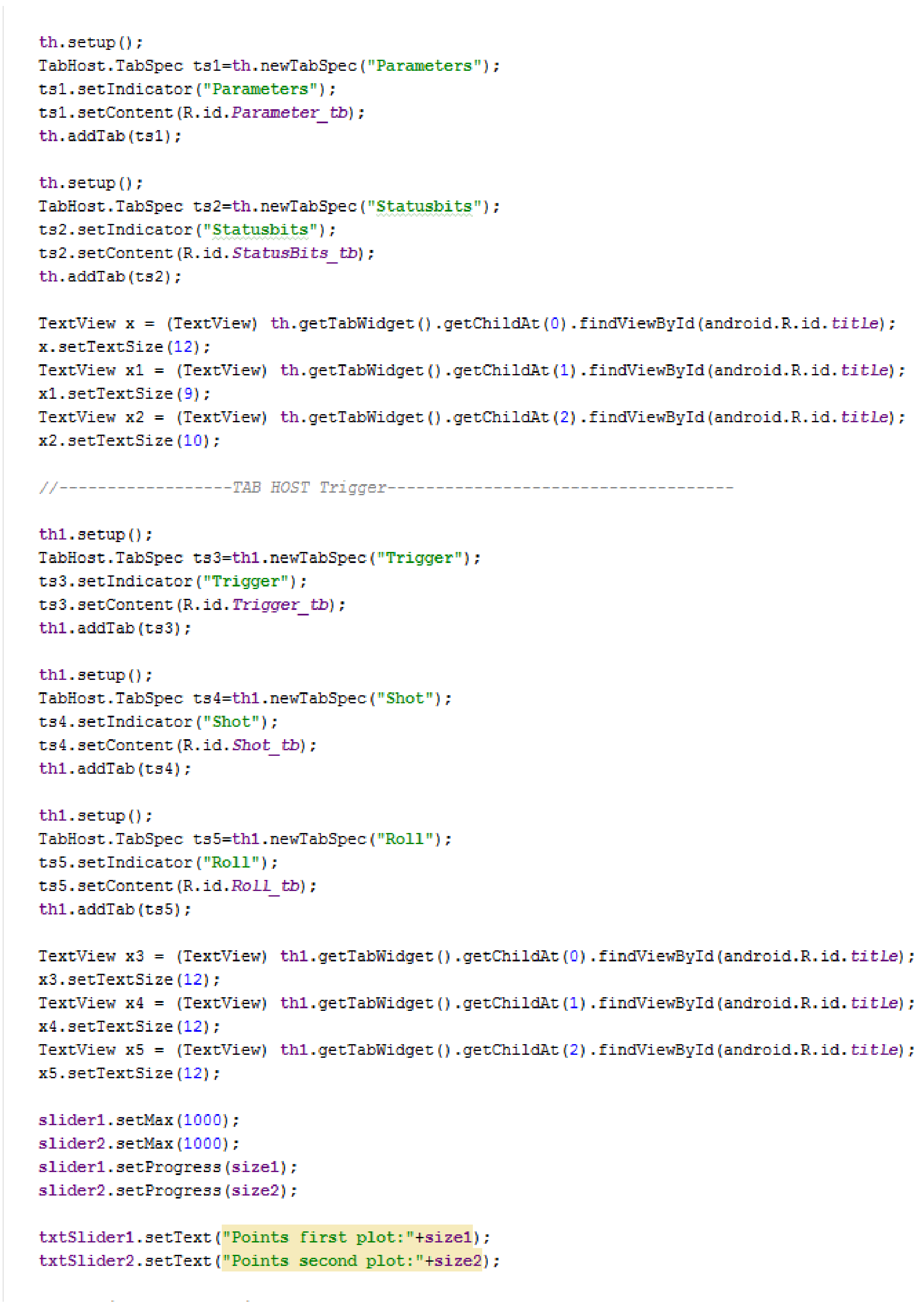


```
package com.aplication.cesc.monitor3_v11;

import ...

//import static com.aplication.cesc.monitor3_v11.R.id.ButtonSend;
//import static com.aplication.cesc.monitor3_v11.R.id.Offset1;
//import static com.aplication.cesc.monitor3_v11.R.id.Offset2;
//import static com.aplication.cesc.monitor3_v11.R.id.TriggerTab;
//import static com.aplication.cesc.monitor3_v11.R.id.seekBar1;

public class Configuration extends AppCompatActivity implements View.OnClickListener{

    Button botograf;
    Button botocom;
    Button botoadd;
    Button botoclear;
    Button botofeed;
    Button botostop;

    Button startRoll;
    Button startShot;

    SeekBar slider1;
    SeekBar slider2;
    TextView txtSlider1;
    TextView txtSlider2;

    EditText trigShot;
    EditText timeShot;
    //EditText showShot;

    EditText trigRoll;
    EditText timeRoll;

    int IntTrigShot;
    int IntTimeShot;
    int IntShowShot;
    int IntTrigRoll;
    int IntTimeRoll;

    String StgTrigShot;
    String StgTimeShot;
    String StgShowShot;
    String StgTrigRoll;
    String StgTimeRoll;

    int size1;
    int size2;
    float sizeSlider;

    float[] punts = new float[50000];
    float[] punts2 = new float[50000];
    int multi=0;
    String ok = ".";
    boolean saved = false;
```

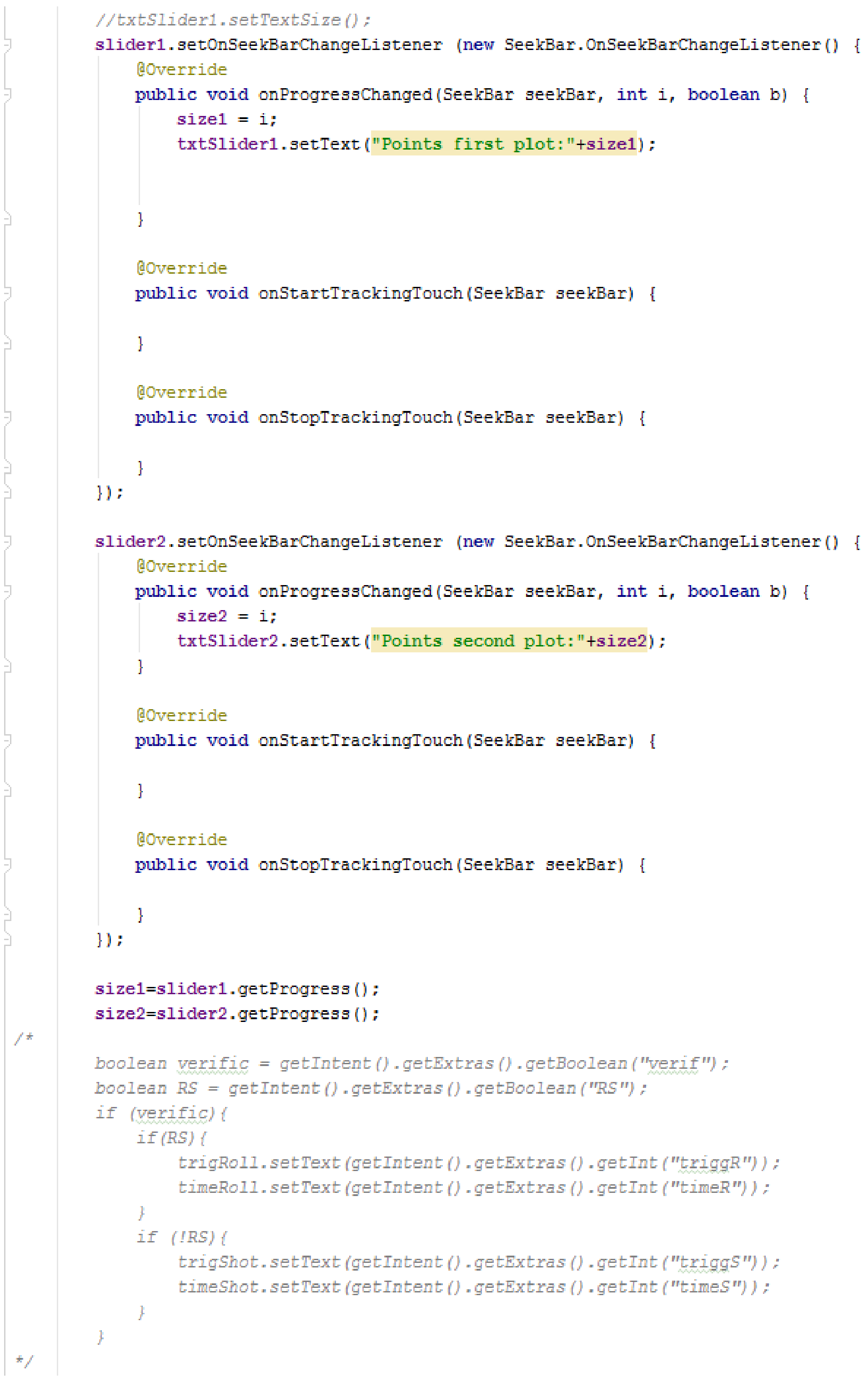


```java
    TabHost th;
    TabHost th1;

    boolean colorBack;
    boolean RigLeft;
    float Escala1;
    float Escala2;
    String unitat1;
    String unitat2;
    int grafcaract1;
    int grafcaract2;
    float Offset1;
    float Offset2;

    EditText Port;
    EditText Ip;
    EditText Missatge;
    TextView MyIp;
    TextView MisRebut;
    Button Connectar;
    Button Enviar;
    Button BotoSC;

    WifiManager wmanager;

    int PORT;
    String IP;

    private Socket socket;
    private ServerSocket serverSocket;

    Handler updateConversationHandler;

    Thread serverThread = null;
    Thread clientThread = null;

    String str;

    int SC = 0;

    boolean config = false;

    String IPBoard = "";
    boolean BackButton = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_configuration);
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);

        Port = (EditText)findViewById(R.id.EditPort);
        Ip = (EditText)findViewById(R.id.EditIp);
        Missatge = (EditText)findViewById(R.id.EditSend);
```

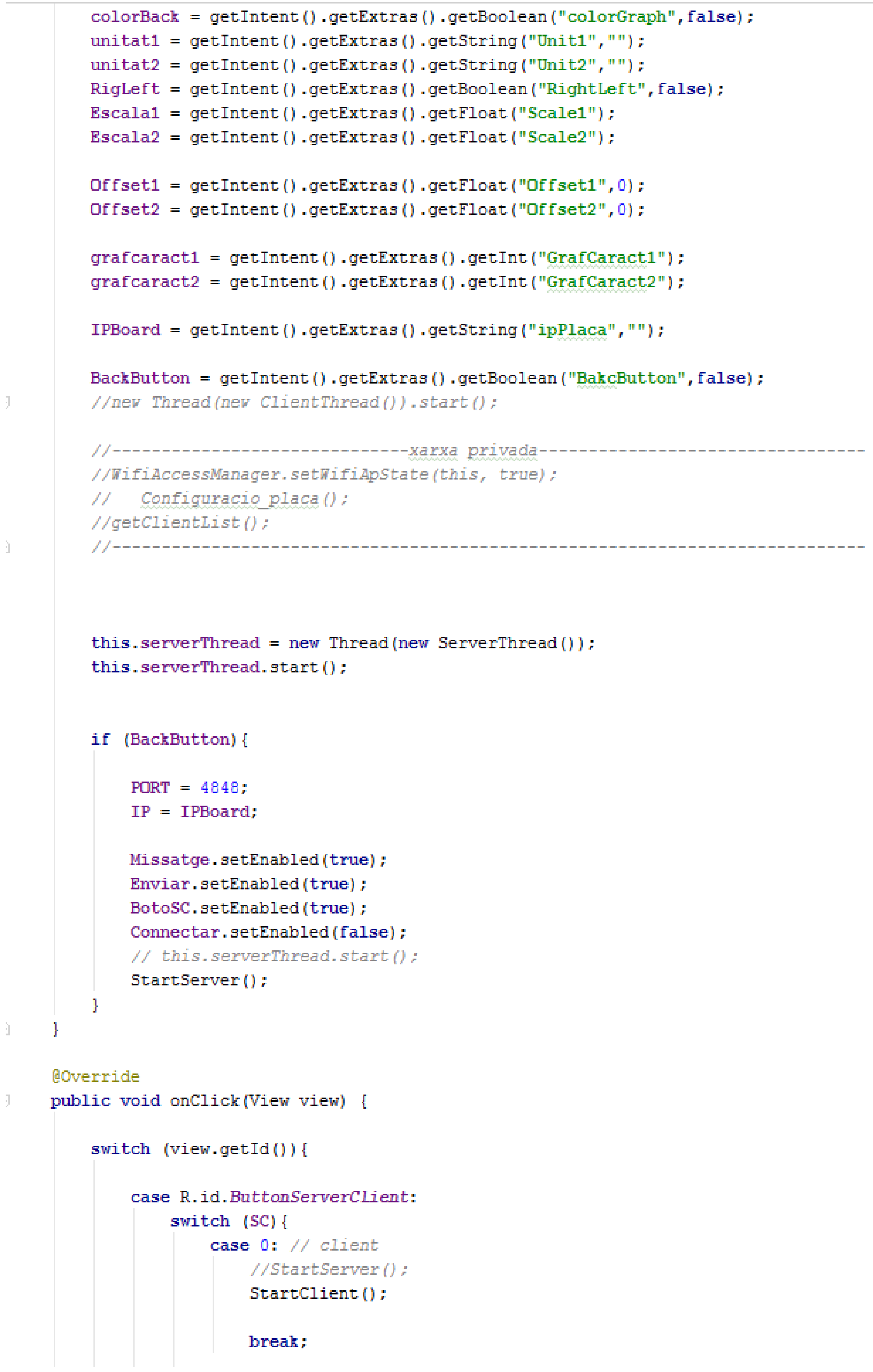
```
Connectar = (Button)findViewById(R.id.ButtonConnect);
Enviar = (Button)findViewById(R.id.ButtonSend);
Connectar.setOnClickListener(this);
Enviar.setOnClickListener(this);

BotoSC = (Button)findViewById(R.id.ButtonServerClient);
BotoSC.setOnClickListener(this);

Missatge.setEnabled(false);
Enviar.setEnabled(false);
BotoSC.setEnabled(false);

wmanager = (WifiManager) getApplicationContext().getSystemService(Context.WIFI_SERVICE);
String ip = Formatter.~~formatIpAddress~~(wmanager.getConnectionInfo().getIpAddress());
MyIp.setText(ip);

updateConversationHandler = new Handler();

new Thread(new ClientThread()).start();

botograf = (Button) findViewById(R.id.botograph);
botograf.setOnClickListener(this);

botocom = (Button) findViewById(R.id.botocomunic);
botocom.setOnClickListener(this);

slider1 = (SeekBar) findViewById(R.id.seekBar1);
slider2 = (SeekBar) findViewById(R.id.seekBar2);

txtSlider1 = (TextView) findViewById(R.id.textSeek1);
txtSlider2 = (TextView) findViewById(R.id.textSeek2);

startRoll = (Button) findViewById(R.id.startRoll);
startRoll.setOnClickListener(this);
trigRoll = (EditText) findViewById(R.id.roll_trigg);
timeRoll = (EditText) findViewById(R.id.roll_time);

startShot = (Button) findViewById(R.id.startShot);
startShot.setOnClickListener(this);
trigShot = (EditText) findViewById(R.id.shot_trigg);
timeShot = (EditText) findViewById(R.id.shot_time);
//showShot = (EditText) findViewById(R.id.show_time);

th = (TabHost) findViewById(R.id.SliderTab);
th1 = (TabHost) findViewById(R.id.TriggerTab);

th.setup();
TabHost.TabSpec ts=th.newTabSpec("Slider");
ts.setIndicator("Slider");
ts.setContent(R.id.Slider_tb);
th.addTab(ts);
```



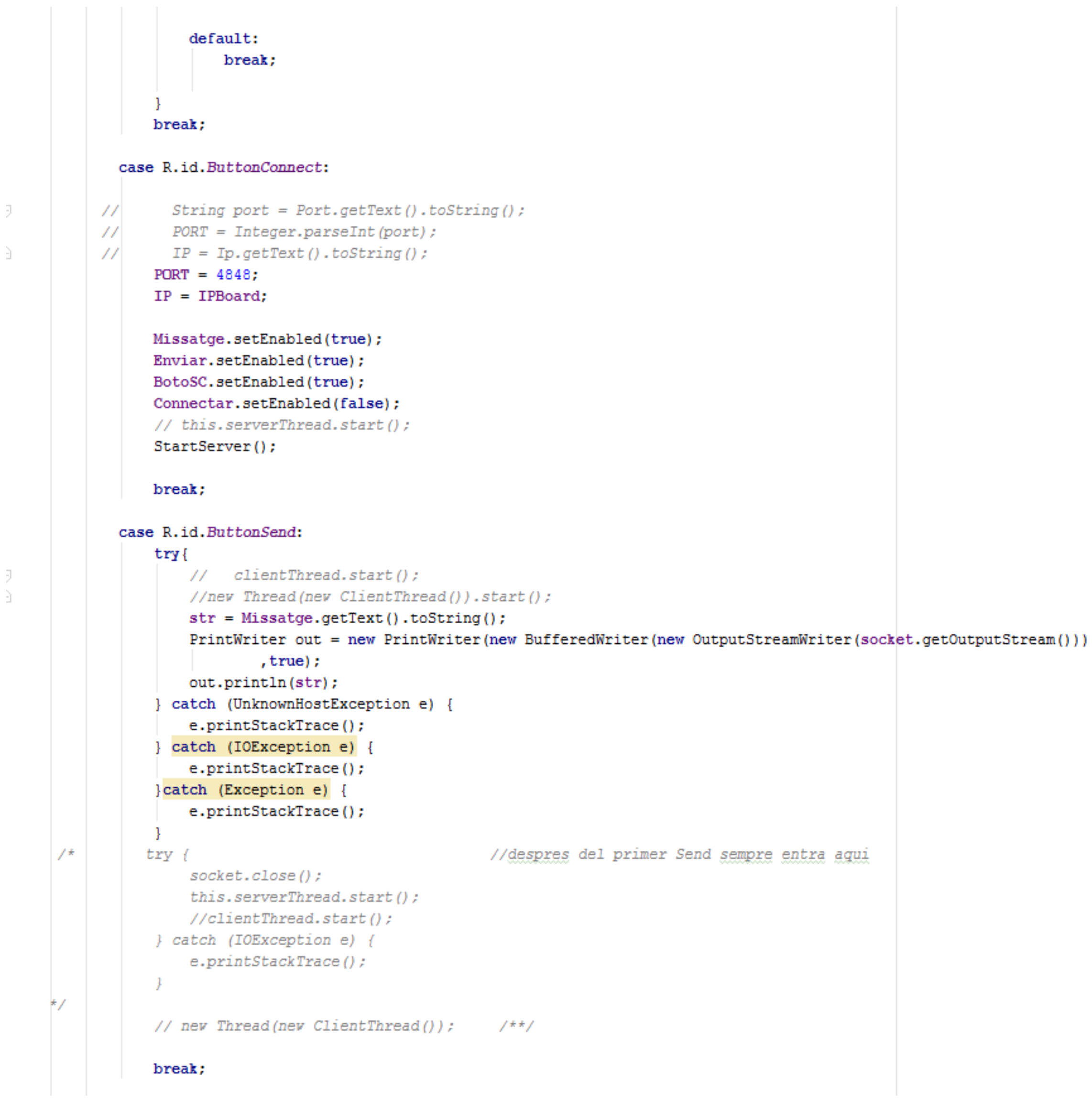
```
th.setup();
TabHost.TabSpec ts1=th.newTabSpec("Parameters");
ts1.setIndicator("Parameters");
ts1.setContent(R.id.Parameter_tb);
th.addTab(ts1);

th.setup();
TabHost.TabSpec ts2=th.newTabSpec("Statusbits");
ts2.setIndicator("Statusbits");
ts2.setContent(R.id.StatusBits_tb);
th.addTab(ts2);

TextView x = (TextView) th.getTabWidget().getChildAt(0).findViewById(android.R.id.title);
x.setTextSize(12);
TextView x1 = (TextView) th.getTabWidget().getChildAt(1).findViewById(android.R.id.title);
x1.setTextSize(9);
TextView x2 = (TextView) th.getTabWidget().getChildAt(2).findViewById(android.R.id.title);
x2.setTextSize(10);

//-----------------TAB HOST Trigger------------------------------------

th1.setup();
TabHost.TabSpec ts3=th1.newTabSpec("Trigger");
ts3.setIndicator("Trigger");
ts3.setContent(R.id.Trigger_tb);
th1.addTab(ts3);

th1.setup();
TabHost.TabSpec ts4=th1.newTabSpec("Shot");
ts4.setIndicator("Shot");
ts4.setContent(R.id.Shot_tb);
th1.addTab(ts4);

th1.setup();
TabHost.TabSpec ts5=th1.newTabSpec("Roll");
ts5.setIndicator("Roll");
ts5.setContent(R.id.Roll_tb);
th1.addTab(ts5);

TextView x3 = (TextView) th1.getTabWidget().getChildAt(0).findViewById(android.R.id.title);
x3.setTextSize(12);
TextView x4 = (TextView) th1.getTabWidget().getChildAt(1).findViewById(android.R.id.title);
x4.setTextSize(12);
TextView x5 = (TextView) th1.getTabWidget().getChildAt(2).findViewById(android.R.id.title);
x5.setTextSize(12);

slider1.setMax(1000);
slider2.setMax(1000);
slider1.setProgress(size1);
slider2.setProgress(size2);

txtSlider1.setText("Points first plot:"+size1);
txtSlider2.setText("Points second plot:"+size2);
```



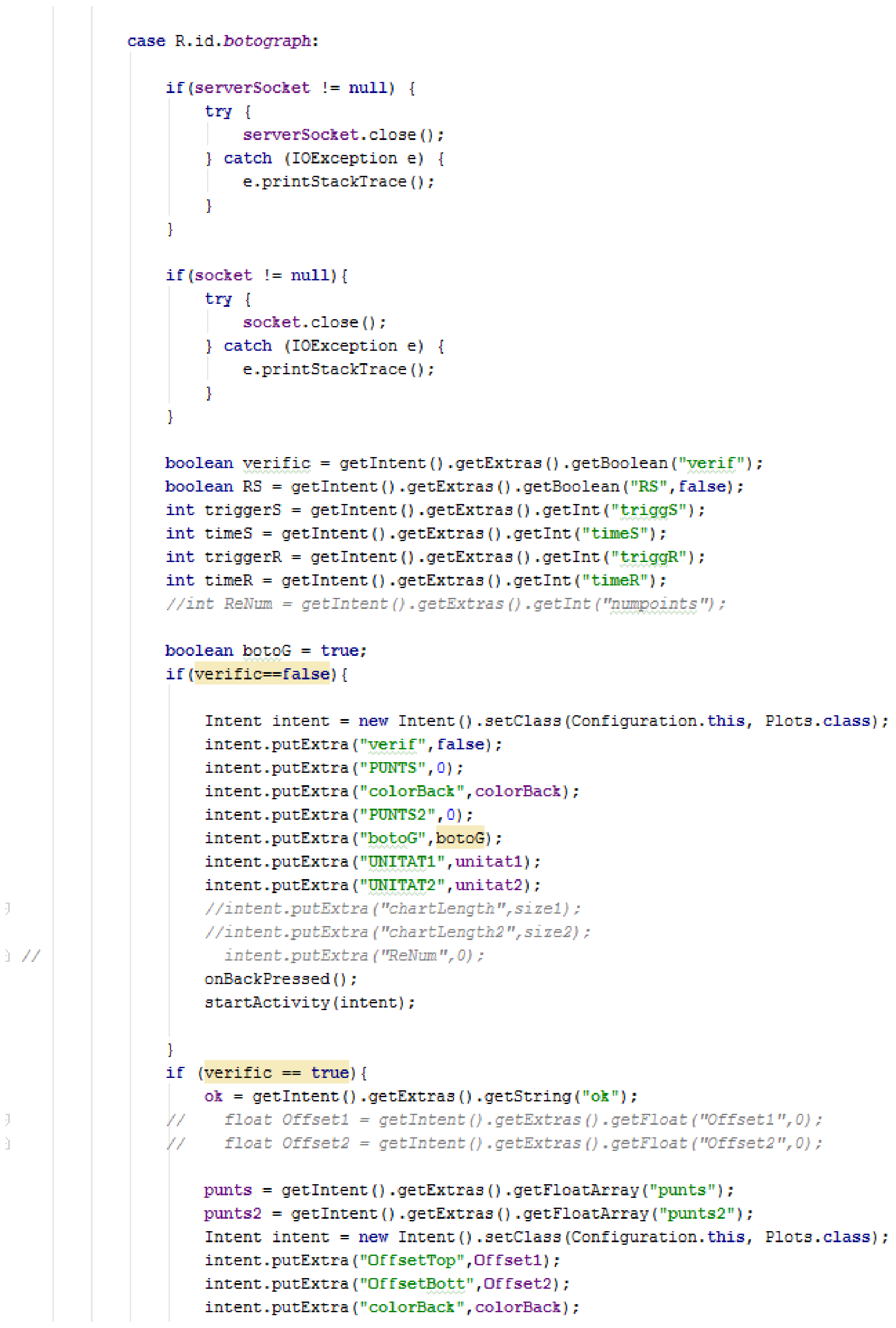
```
        //txtSlider1.setTextSize();
        slider1.setOnSeekBarChangeListener (new SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
                size1 = i;
                txtSlider1.setText("Points first plot:"+size1);


            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });

        slider2.setOnSeekBarChangeListener (new SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
                size2 = i;
                txtSlider2.setText("Points second plot:"+size2);
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });

        size1=slider1.getProgress();
        size2=slider2.getProgress();
/*
        boolean verific = getIntent().getExtras().getBoolean("verif");
        boolean RS = getIntent().getExtras().getBoolean("RS");
        if (verific){
            if(RS){
                trigRoll.setText(getIntent().getExtras().getInt("triggR"));
                timeRoll.setText(getIntent().getExtras().getInt("timeR"));
            }
            if (!RS){
                trigShot.setText(getIntent().getExtras().getInt("triggS"));
                timeShot.setText(getIntent().getExtras().getInt("timeS"));
            }
        }
*/
```



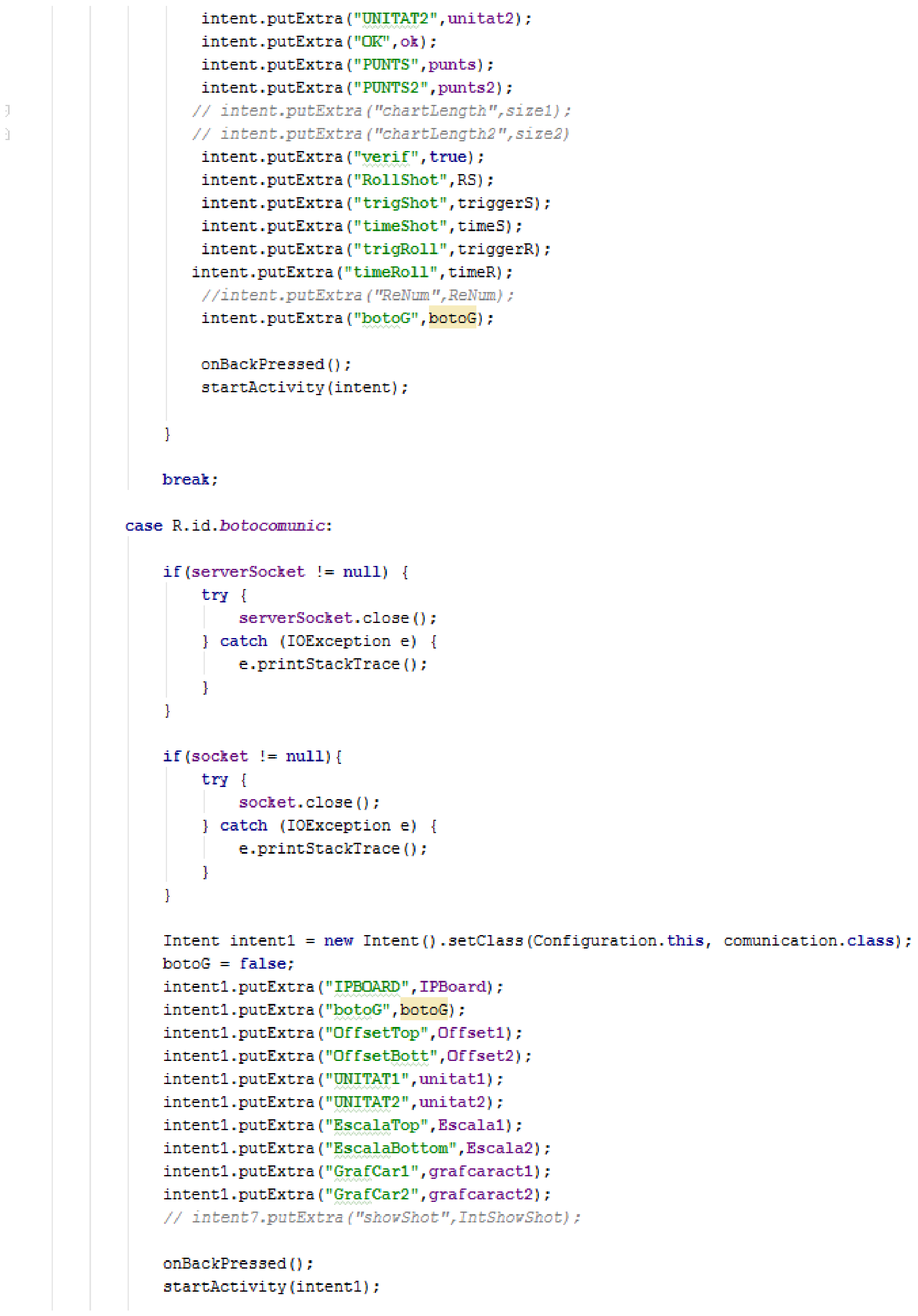
```
    colorBack = getIntent().getExtras().getBoolean("colorGraph",false);
    unitat1 = getIntent().getExtras().getString("Unit1","");
    unitat2 = getIntent().getExtras().getString("Unit2","");
    RigLeft = getIntent().getExtras().getBoolean("RightLeft",false);
    Escala1 = getIntent().getExtras().getFloat("Scale1");
    Escala2 = getIntent().getExtras().getFloat("Scale2");

    Offset1 = getIntent().getExtras().getFloat("Offset1",0);
    Offset2 = getIntent().getExtras().getFloat("Offset2",0);

    grafcaract1 = getIntent().getExtras().getInt("GrafCaract1");
    grafcaract2 = getIntent().getExtras().getInt("GrafCaract2");

    IPBoard = getIntent().getExtras().getString("ipPlaca","");

    BackButton = getIntent().getExtras().getBoolean("BakcButton",false);
    //new Thread(new ClientThread()).start();

    //-----------------------------xarxa privada---------------------------------
    //WifiAccessManager.setWifiApState(this, true);
    //   Configuracio_placa();
    //getClientList();
    //---------------------------------------------------------------------------



    this.serverThread = new Thread(new ServerThread());
    this.serverThread.start();


    if (BackButton){

        PORT = 4848;
        IP = IPBoard;

        Missatge.setEnabled(true);
        Enviar.setEnabled(true);
        BotoSC.setEnabled(true);
        Connectar.setEnabled(false);
        // this.serverThread.start();
        StartServer();
    }
}

@Override
public void onClick(View view) {

    switch (view.getId()){

        case R.id.ButtonServerClient:
            switch (SC){
                case 0: // client
                    //StartServer();
                    StartClient();

                    break;
```



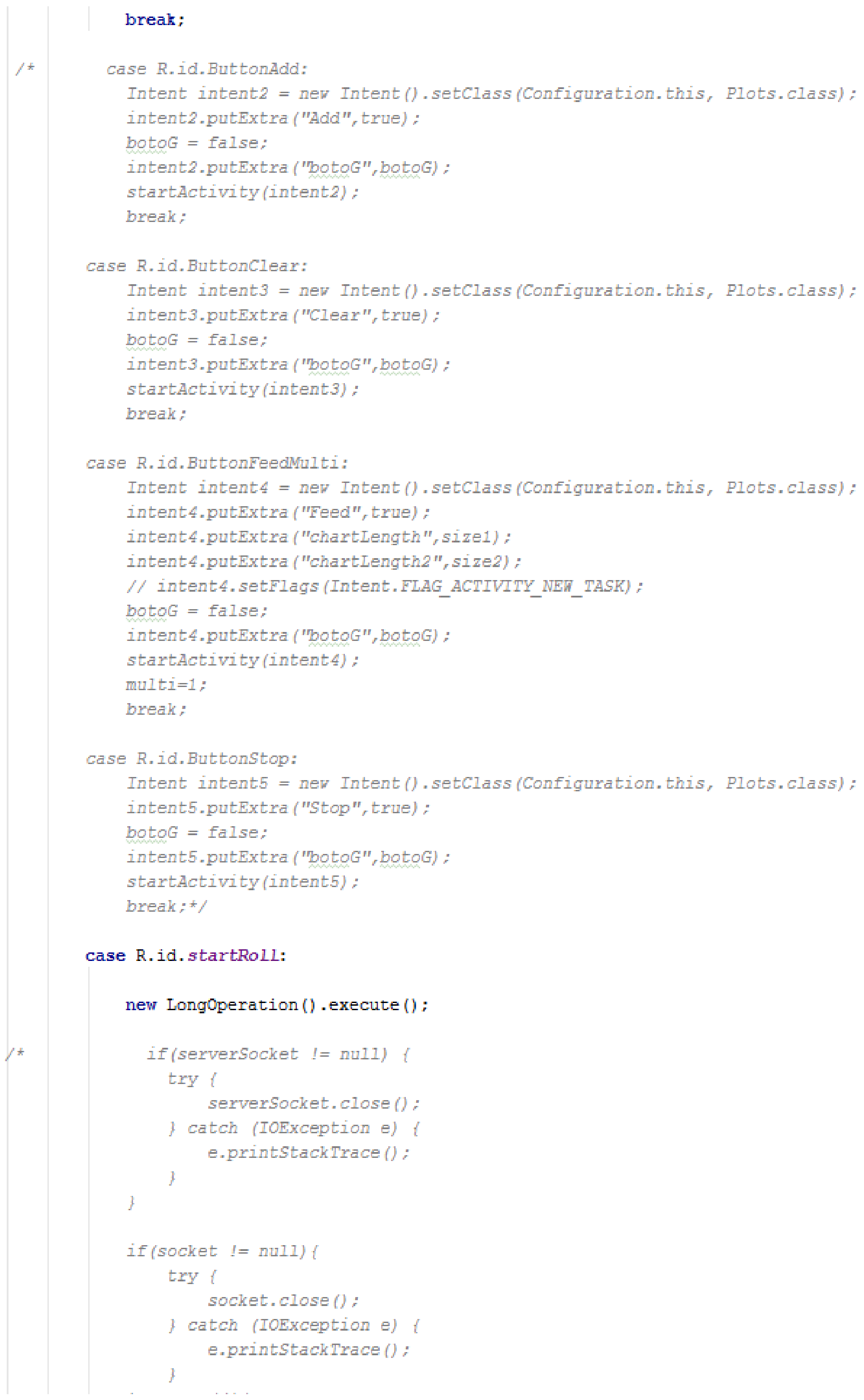
```
                    default:
                        break;

                }
                break;

            case R.id.ButtonConnect:

//                  String port = Port.getText().toString();
//                  PORT = Integer.parseInt(port);
//                  IP = Ip.getText().toString();
                PORT = 4848;
                IP = IPBoard;

                Missatge.setEnabled(true);
                Enviar.setEnabled(true);
                BotoSC.setEnabled(true);
                Connectar.setEnabled(false);
                // this.serverThread.start();
                StartServer();

                break;

            case R.id.ButtonSend:
                try{
                    //   clientThread.start();
                    //new Thread(new ClientThread()).start();
                    str = Missatge.getText().toString();
                    PrintWriter out = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()))
                            ,true);
                    out.println(str);
                } catch (UnknownHostException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                }catch (Exception e) {
                    e.printStackTrace();
                }
  /*           try {                              //despres del primer Send sempre entra aqui
                    socket.close();
                    this.serverThread.start();
                    //clientThread.start();
                } catch (IOException e) {
                    e.printStackTrace();
                }
*/
                // new Thread(new ClientThread());      /**/

                break;
```



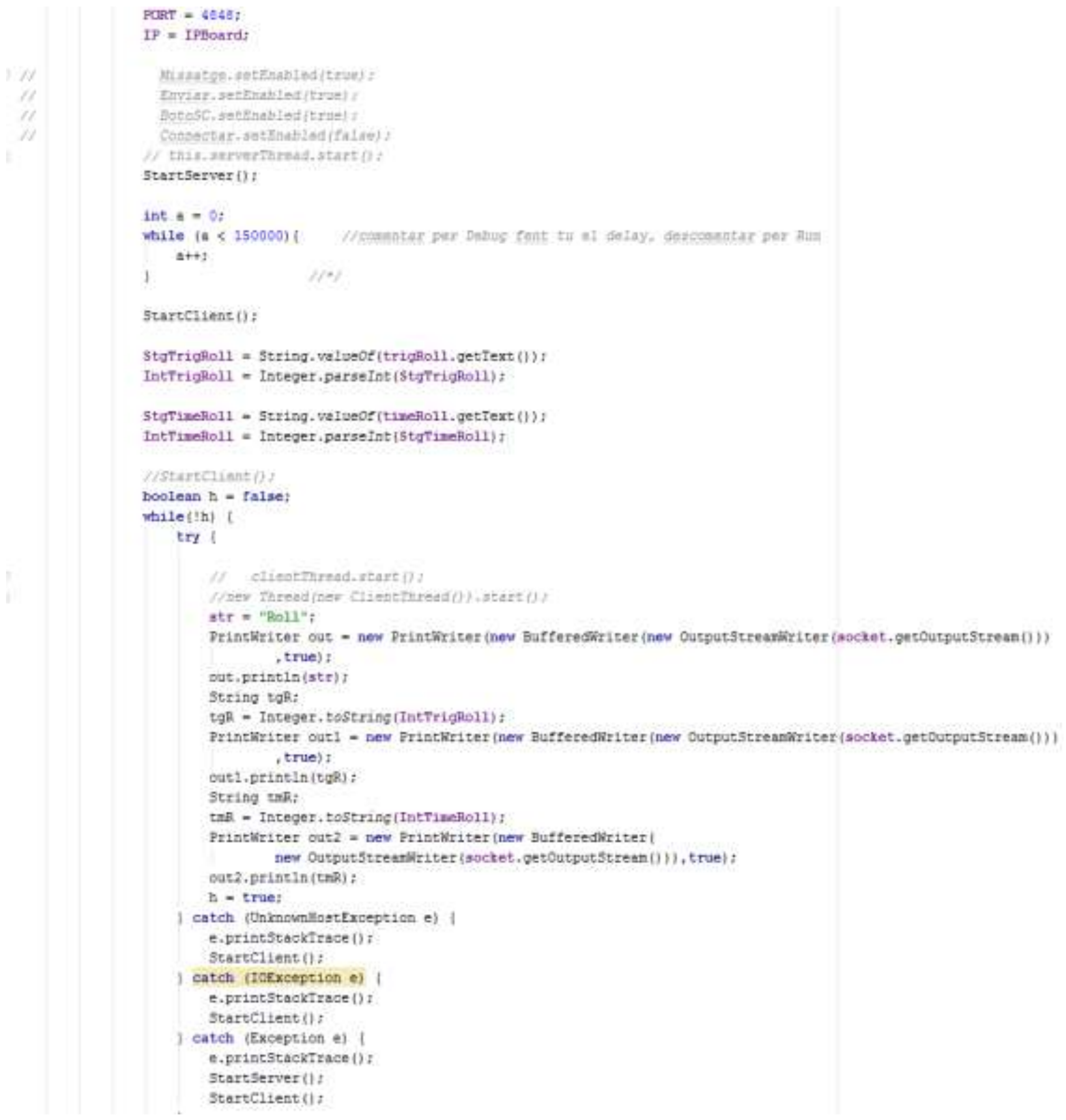
```
case R.id.botograph:

    if(serverSocket != null) {
        try {
            serverSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    if(socket != null){
        try {
            socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    boolean verific = getIntent().getExtras().getBoolean("verif");
    boolean RS = getIntent().getExtras().getBoolean("RS",false);
    int triggerS = getIntent().getExtras().getInt("triggS");
    int timeS = getIntent().getExtras().getInt("timeS");
    int triggerR = getIntent().getExtras().getInt("triggR");
    int timeR = getIntent().getExtras().getInt("timeR");
    //int ReNum = getIntent().getExtras().getInt("numpoints");

    boolean botoG = true;
    if(verific==false){

        Intent intent = new Intent().setClass(Configuration.this, Plots.class);
        intent.putExtra("verif",false);
        intent.putExtra("PUNTS",0);
        intent.putExtra("colorBack",colorBack);
        intent.putExtra("PUNTS2",0);
        intent.putExtra("botoG",botoG);
        intent.putExtra("UNITAT1",unitat1);
        intent.putExtra("UNITAT2",unitat2);
        //intent.putExtra("chartLength",size1);
        //intent.putExtra("chartLength2",size2);
//        intent.putExtra("ReNum",0);
        onBackPressed();
        startActivity(intent);

    }
    if (verific == true){
        ok = getIntent().getExtras().getString("ok");
//      float Offset1 = getIntent().getExtras().getFloat("Offset1",0);
//      float Offset2 = getIntent().getExtras().getFloat("Offset2",0);

        punts = getIntent().getExtras().getFloatArray("punts");
        punts2 = getIntent().getExtras().getFloatArray("punts2");
        Intent intent = new Intent().setClass(Configuration.this, Plots.class);
        intent.putExtra("OffsetTop",Offset1);
        intent.putExtra("OffsetBott",Offset2);
        intent.putExtra("colorBack",colorBack);
```



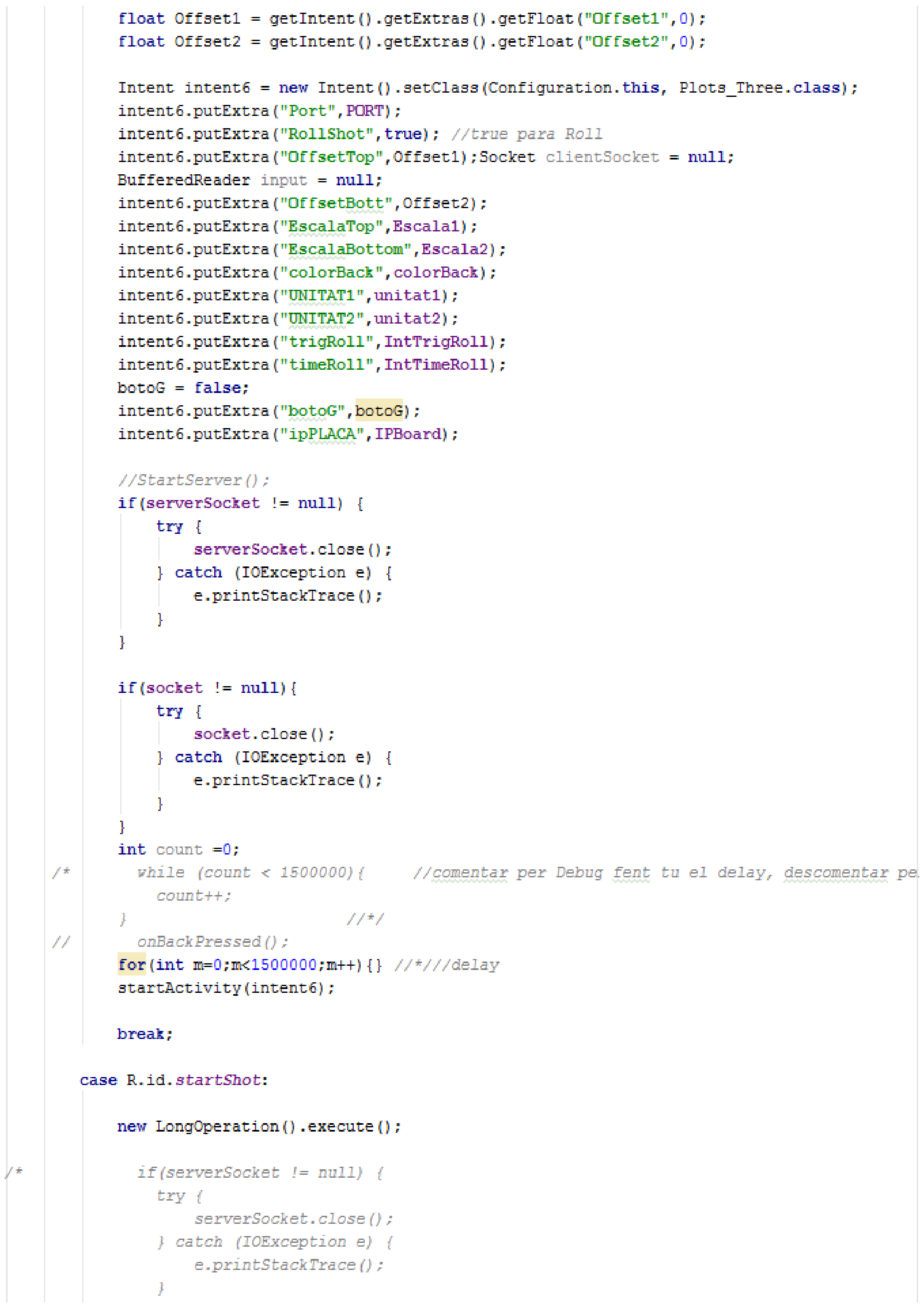
```
                intent.putExtra("UNITAT2",unitat2);
                intent.putExtra("OK",ok);
                intent.putExtra("PUNTS",punts);
                intent.putExtra("PUNTS2",punts2);
               // intent.putExtra("chartLength",size1);
               // intent.putExtra("chartLength2",size2)
                intent.putExtra("verif",true);
                intent.putExtra("RollShot",RS);
                intent.putExtra("trigShot",triggerS);
                intent.putExtra("timeShot",timeS);
                intent.putExtra("trigRoll",triggerR);
               intent.putExtra("timeRoll",timeR);
                //intent.putExtra("ReNum",ReNum);
                intent.putExtra("botoG",botoG);

                onBackPressed();
                startActivity(intent);

            }

            break;

        case R.id.botocomunic:

            if(serverSocket != null) {
                try {
                    serverSocket.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            if(socket != null){
                try {
                    socket.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            Intent intent1 = new Intent().setClass(Configuration.this, comunication.class);
            botoG = false;
            intent1.putExtra("IPBOARD",IPBoard);
            intent1.putExtra("botoG",botoG);
            intent1.putExtra("OffsetTop",Offset1);
            intent1.putExtra("OffsetBott",Offset2);
            intent1.putExtra("UNITAT1",unitat1);
            intent1.putExtra("UNITAT2",unitat2);
            intent1.putExtra("EscalaTop",Escala1);
            intent1.putExtra("EscalaBottom",Escala2);
            intent1.putExtra("GrafCar1",grafcaract1);
            intent1.putExtra("GrafCar2",grafcaract2);
            // intent7.putExtra("showShot",IntShowShot);

            onBackPressed();
            startActivity(intent1);
```



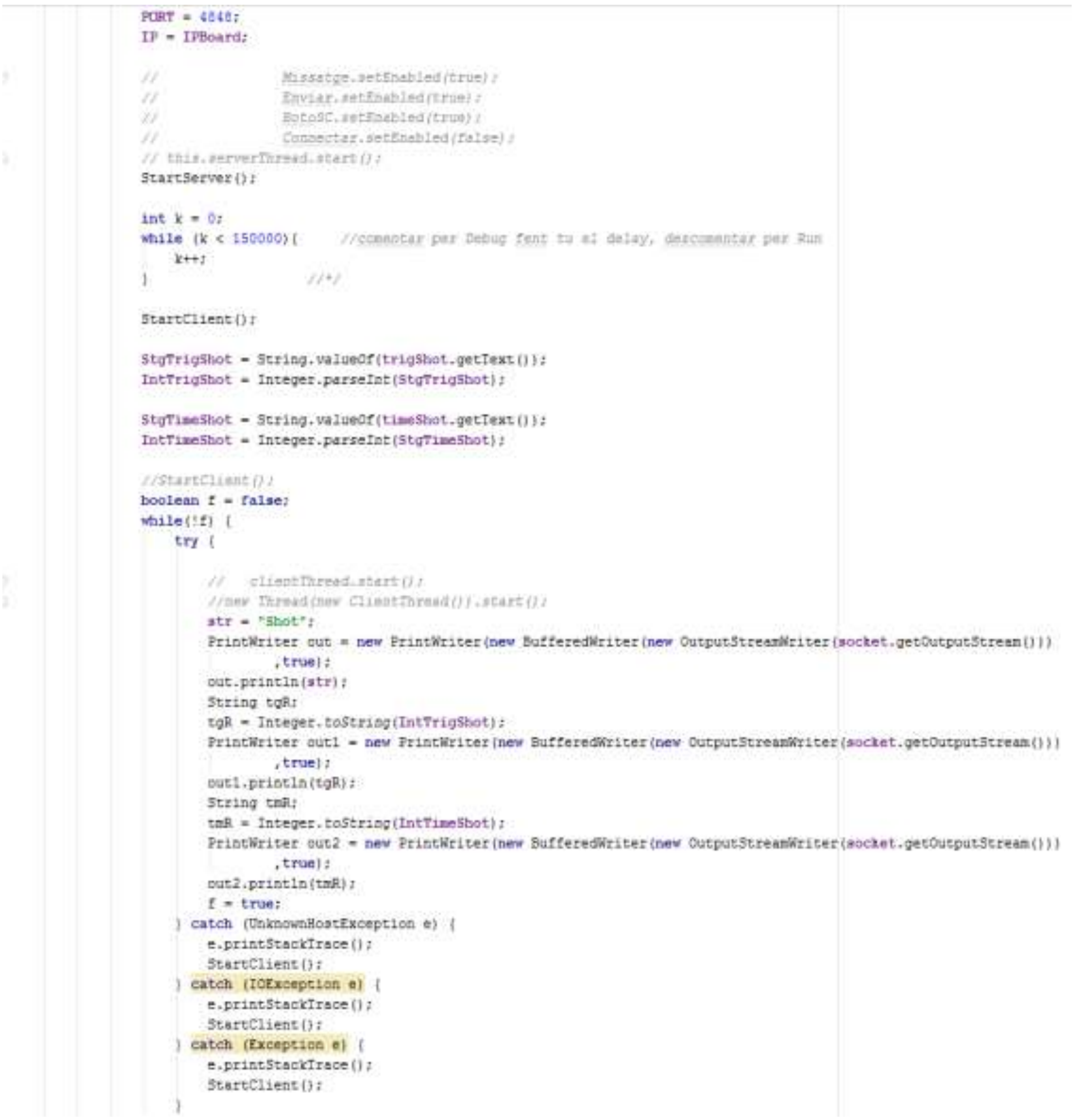
```
        break;

/*       case R.id.ButtonAdd:
            Intent intent2 = new Intent().setClass(Configuration.this, Plots.class);
            intent2.putExtra("Add",true);
            botoG = false;
            intent2.putExtra("botoG",botoG);
            startActivity(intent2);
            break;

        case R.id.ButtonClear:
            Intent intent3 = new Intent().setClass(Configuration.this, Plots.class);
            intent3.putExtra("Clear",true);
            botoG = false;
            intent3.putExtra("botoG",botoG);
            startActivity(intent3);
            break;

        case R.id.ButtonFeedMulti:
            Intent intent4 = new Intent().setClass(Configuration.this, Plots.class);
            intent4.putExtra("Feed",true);
            intent4.putExtra("chartLength",size1);
            intent4.putExtra("chartLength2",size2);
            // intent4.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            botoG = false;
            intent4.putExtra("botoG",botoG);
            startActivity(intent4);
            multi=1;
            break;

        case R.id.ButtonStop:
            Intent intent5 = new Intent().setClass(Configuration.this, Plots.class);
            intent5.putExtra("Stop",true);
            botoG = false;
            intent5.putExtra("botoG",botoG);
            startActivity(intent5);
            break;*/

        case R.id.startRoll:

            new LongOperation().execute();

/*            if(serverSocket != null) {
                try {
                    serverSocket.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            if(socket != null){
                try {
                    socket.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
```



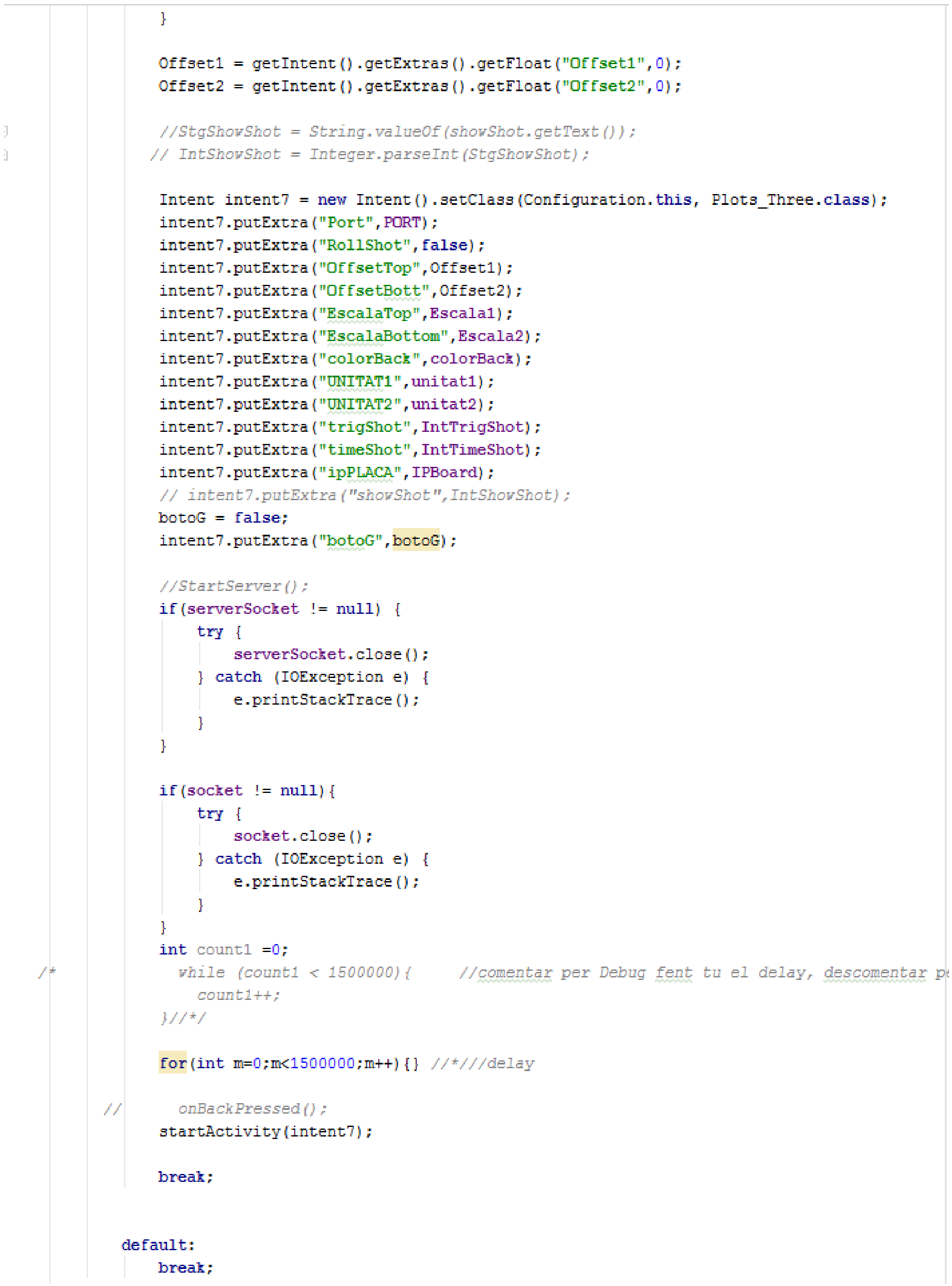
```
PORT = 4848;
IP = IPBoard;

//    Missatge.setEnabled(true);
//    Enviar.setEnabled(true);
//    BotoSC.setEnabled(true);
//    Connectar.setEnabled(false);
// this.serverThread.start();
StartServer();

int a = 0;
while (a < 150000){     //comentar per Debug fent tu el delay, descomentar per Run
    a++;
}                   //*/

StartClient();

StgTrigRoll = String.valueOf(trigRoll.getText());
IntTrigRoll = Integer.parseInt(StgTrigRoll);

StgTimeRoll = String.valueOf(timeRoll.getText());
IntTimeRoll = Integer.parseInt(StgTimeRoll);

//StartClient();
boolean h = false;
while(!h) {
    try {

        //   clientThread.start();
        //new Thread(new ClientThread()).start();
        str = "Roll";
        PrintWriter out = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()))
                ,true);
        out.println(str);
        String tgR;
        tgR = Integer.toString(IntTrigRoll);
        PrintWriter out1 = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()))
                ,true);
        out1.println(tgR);
        String tmR;
        tmR = Integer.toString(IntTimeRoll);
        PrintWriter out2 = new PrintWriter(new BufferedWriter(
                new OutputStreamWriter(socket.getOutputStream())),true);
        out2.println(tmR);
        h = true;
    } catch (UnknownHostException e) {
        e.printStackTrace();
        StartClient();
    } catch (IOException e) {
        e.printStackTrace();
        StartClient();
    } catch (Exception e) {
        e.printStackTrace();
        StartServer();
        StartClient();
```



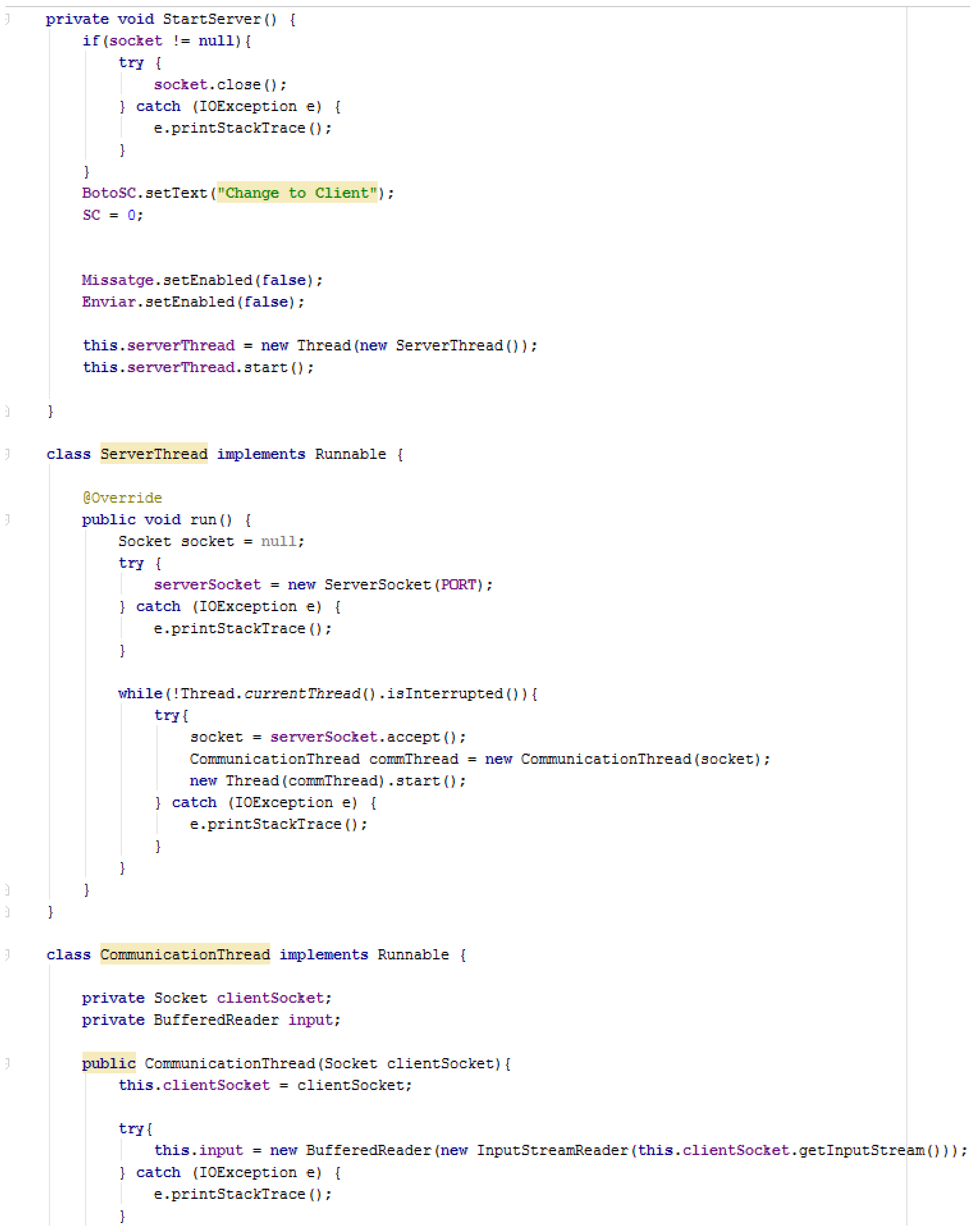
```
            float Offset1 = getIntent().getExtras().getFloat("Offset1",0);
            float Offset2 = getIntent().getExtras().getFloat("Offset2",0);

            Intent intent6 = new Intent().setClass(Configuration.this, Plots_Three.class);
            intent6.putExtra("Port",PORT);
            intent6.putExtra("RollShot",true); //true para Roll
            intent6.putExtra("OffsetTop",Offset1);Socket clientSocket = null;
            BufferedReader input = null;
            intent6.putExtra("OffsetBott",Offset2);
            intent6.putExtra("EscalaTop",Escala1);
            intent6.putExtra("EscalaBottom",Escala2);
            intent6.putExtra("colorBack",colorBack);
            intent6.putExtra("UNITAT1",unitat1);
            intent6.putExtra("UNITAT2",unitat2);
            intent6.putExtra("trigRoll",IntTrigRoll);
            intent6.putExtra("timeRoll",IntTimeRoll);
            botoG = false;
            intent6.putExtra("botoG",botoG);
            intent6.putExtra("ipPLACA",IPBoard);

            //StartServer();
            if(serverSocket != null) {
                try {
                    serverSocket.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            if(socket != null){
                try {
                    socket.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
            int count =0;
    /*        while (count < 1500000){      //comentar per Debug fent tu el delay, descomentar pe
                count++;
            }                       //*/
    //        onBackPressed();
            for(int m=0;m<1500000;m++){} //*///delay
            startActivity(intent6);

            break;

        case R.id.startShot:

            new LongOperation().execute();

/*            if(serverSocket != null) {
                try {
                    serverSocket.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
```



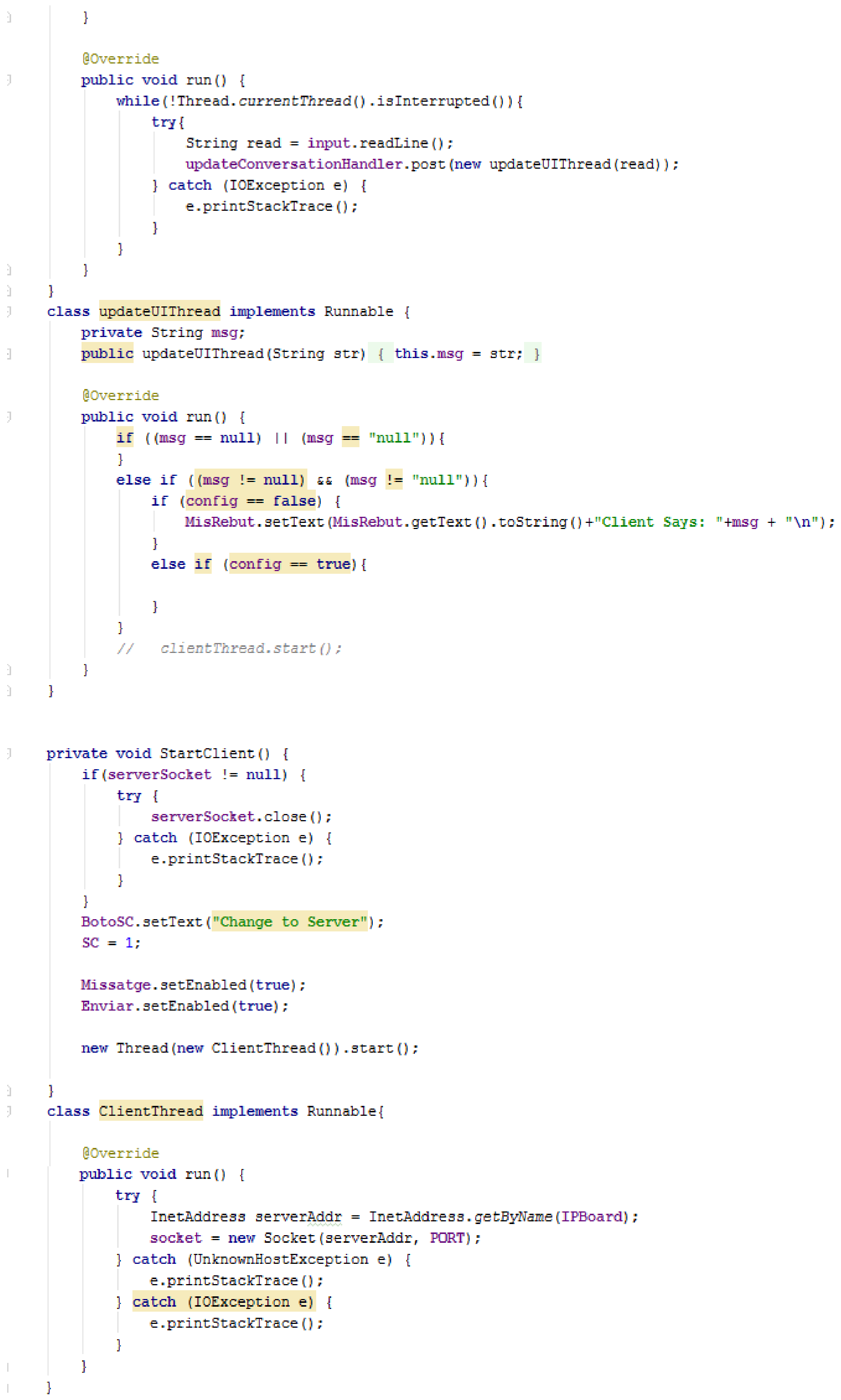
```
PORT = 4848;
IP = IPBoard;

//              Missatge.setEnabled(true);
//              Enviar.setEnabled(true);
//              BotoSC.setEnabled(true);
//              Connectar.setEnabled(false);
// this.serverThread.start();
StartServer();

int k = 0;
while (k < 150000){     //comentar per Debug fent tu el delay, descomentar per Run
    k++;
}                   //*/

StartClient();

StgTrigShot = String.valueOf(trigShot.getText());
IntTrigShot = Integer.parseInt(StgTrigShot);

StgTimeShot = String.valueOf(timeShot.getText());
IntTimeShot = Integer.parseInt(StgTimeShot);

//StartClient();
boolean f = false;
while(!f) {
    try {

        //   clientThread.start();
        //new Thread(new ClientThread()).start();
        str = "Shot";
        PrintWriter out = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()))
                ,true);
        out.println(str);
        String tgR;
        tgR = Integer.toString(IntTrigShot);
        PrintWriter out1 = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()))
                ,true);
        out1.println(tgR);
        String tmR;
        tmR = Integer.toString(IntTimeShot);
        PrintWriter out2 = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()))
                ,true);
        out2.println(tmR);
        f = true;
    } catch (UnknownHostException e) {
        e.printStackTrace();
        StartClient();
    } catch (IOException e) {
        e.printStackTrace();
        StartClient();
    } catch (Exception e) {
        e.printStackTrace();
        StartClient();
    }
```



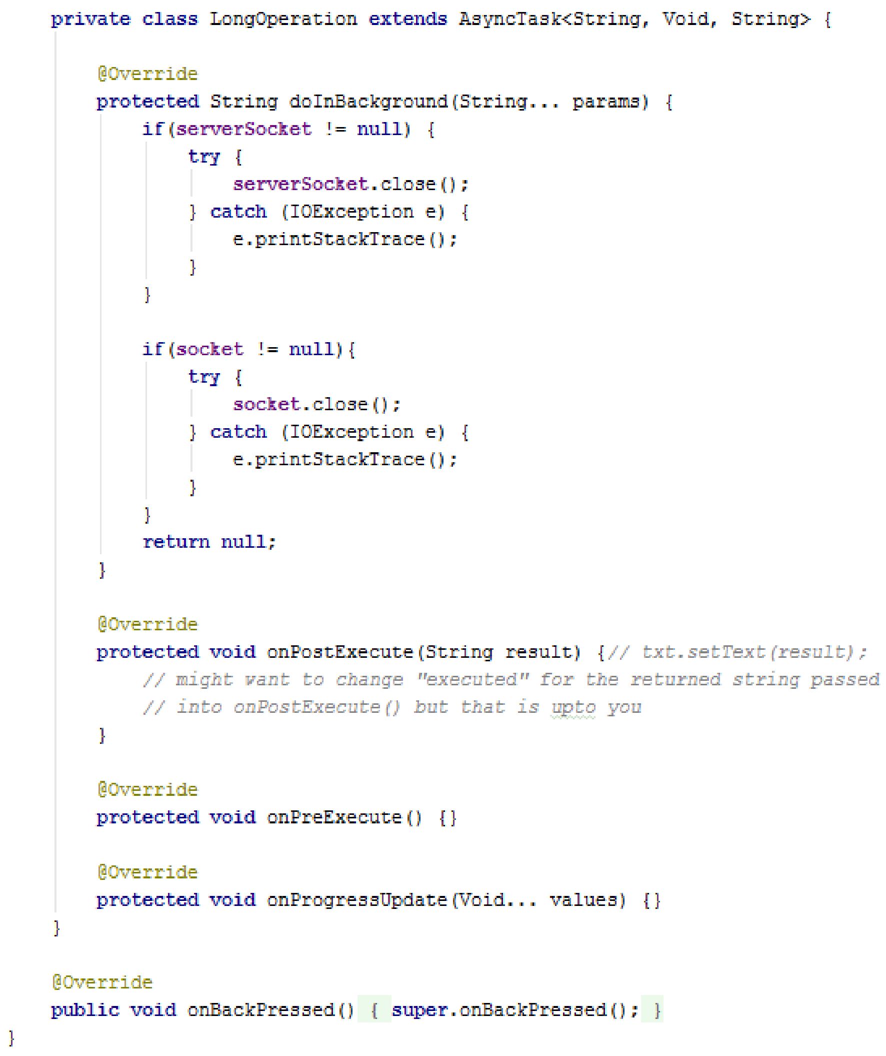
```
        }

        Offset1 = getIntent().getExtras().getFloat("Offset1",0);
        Offset2 = getIntent().getExtras().getFloat("Offset2",0);

        //StgShowShot = String.valueOf(showShot.getText());
       // IntShowShot = Integer.parseInt(StgShowShot);

        Intent intent7 = new Intent().setClass(Configuration.this, Plots_Three.class);
        intent7.putExtra("Port",PORT);
        intent7.putExtra("RollShot",false);
        intent7.putExtra("OffsetTop",Offset1);
        intent7.putExtra("OffsetBott",Offset2);
        intent7.putExtra("EscalaTop",Escala1);
        intent7.putExtra("EscalaBottom",Escala2);
        intent7.putExtra("colorBack",colorBack);
        intent7.putExtra("UNITAT1",unitat1);
        intent7.putExtra("UNITAT2",unitat2);
        intent7.putExtra("trigShot",IntTrigShot);
        intent7.putExtra("timeShot",IntTimeShot);
        intent7.putExtra("ipPLACA",IPBoard);
        // intent7.putExtra("showShot",IntShowShot);
        botoG = false;
        intent7.putExtra("botoG",botoG);

        //StartServer();
        if(serverSocket != null) {
            try {
                serverSocket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        if(socket != null){
            try {
                socket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        int count1 =0;
/*        while (count1 < 1500000){      //comentar per Debug fent tu el delay, descomentar p
            count1++;
        }//*/

        for(int m=0;m<1500000;m++){} //*///delay

//        onBackPressed();
        startActivity(intent7);

        break;


    default:
        break;
```



```java
private void StartServer() {
    if(socket != null){
        try {
            socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    BotoSC.setText("Change to Client");
    SC = 0;


    Missatge.setEnabled(false);
    Enviar.setEnabled(false);

    this.serverThread = new Thread(new ServerThread());
    this.serverThread.start();

}

class ServerThread implements Runnable {

    @Override
    public void run() {
        Socket socket = null;
        try {
            serverSocket = new ServerSocket(PORT);
        } catch (IOException e) {
            e.printStackTrace();
        }

        while(!Thread.currentThread().isInterrupted()){
            try{
                socket = serverSocket.accept();
                CommunicationThread commThread = new CommunicationThread(socket);
                new Thread(commThread).start();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

class CommunicationThread implements Runnable {

    private Socket clientSocket;
    private BufferedReader input;

    public CommunicationThread(Socket clientSocket){
        this.clientSocket = clientSocket;

        try{
            this.input = new BufferedReader(new InputStreamReader(this.clientSocket.getInputStream()));
        } catch (IOException e) {
            e.printStackTrace();
        }
```

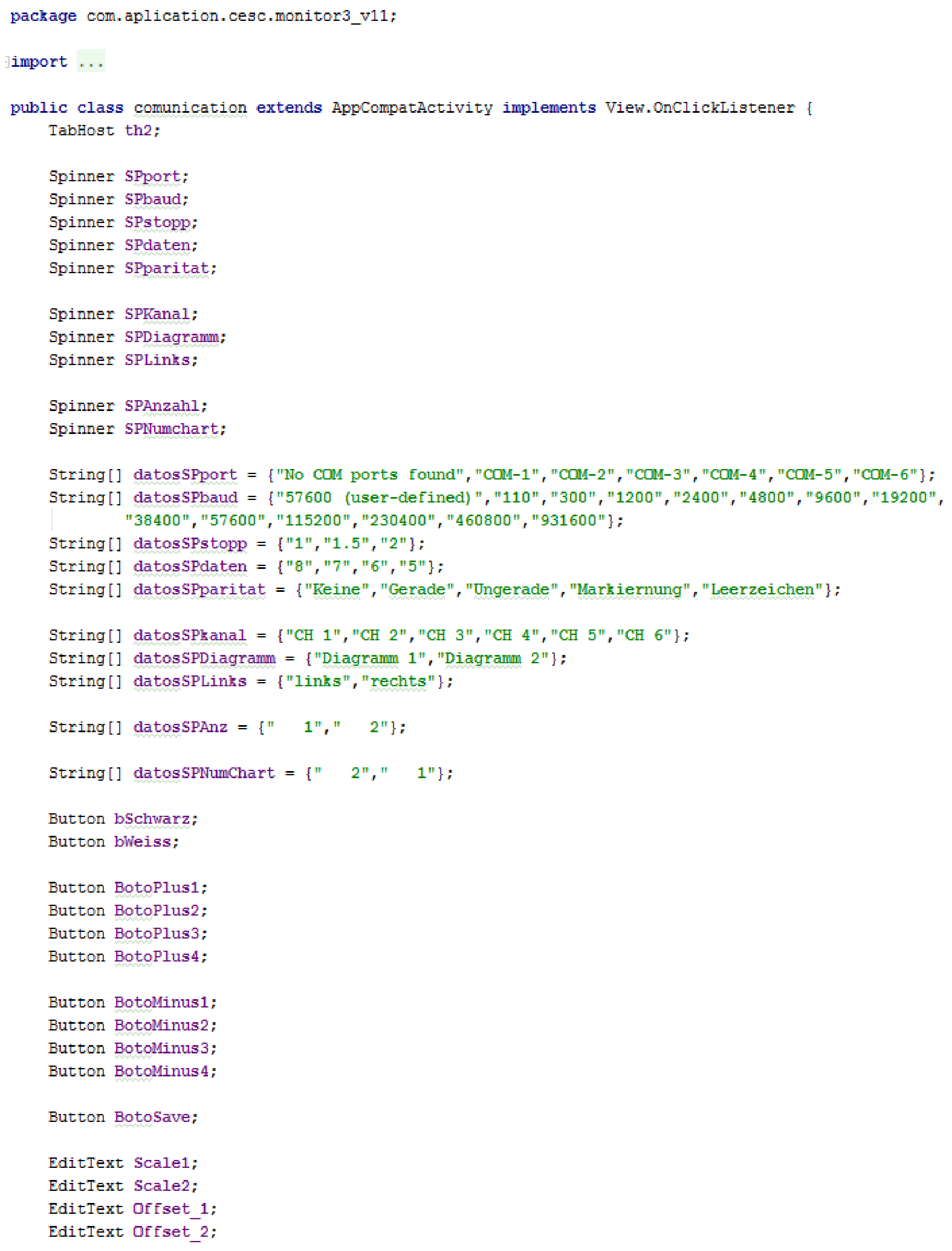


```
    }

    @Override
    public void run() {
        while(!Thread.currentThread().isInterrupted()){
            try{
                String read = input.readLine();
                updateConversationHandler.post(new updateUIThread(read));
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
class updateUIThread implements Runnable {
    private String msg;
    public updateUIThread(String str) { this.msg = str; }

    @Override
    public void run() {
        if ((msg == null) || (msg == "null")){
        }
        else if ((msg != null) && (msg != "null")){
            if (config == false) {
                MisRebut.setText(MisRebut.getText().toString()+"Client Says: "+msg + "\n");
            }
            else if (config == true){

            }
        }
        //   clientThread.start();
    }
}


private void StartClient() {
    if(serverSocket != null) {
        try {
            serverSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    BotoSC.setText("Change to Server");
    SC = 1;

    Missatge.setEnabled(true);
    Enviar.setEnabled(true);

    new Thread(new ClientThread()).start();

}
class ClientThread implements Runnable{

    @Override
    public void run() {
        try {
            InetAddress serverAddr = InetAddress.getByName(IPBoard);
            socket = new Socket(serverAddr, PORT);
        } catch (UnknownHostException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```



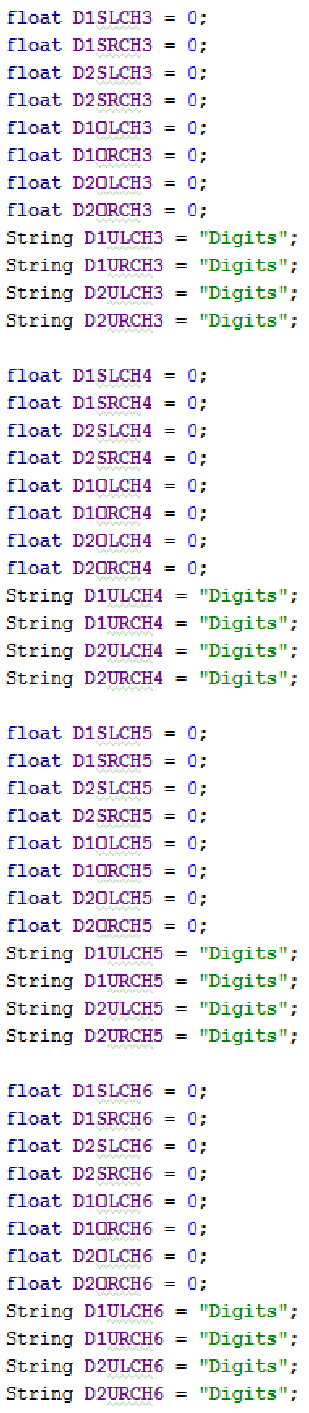
```
    private class LongOperation extends AsyncTask<String, Void, String> {

        @Override
        protected String doInBackground(String... params) {
            if(serverSocket != null) {
                try {
                    serverSocket.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            if(socket != null){
                try {
                    socket.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
            return null;
        }

        @Override
        protected void onPostExecute(String result) {// txt.setText(result);
            // might want to change "executed" for the returned string passed
            // into onPostExecute() but that is upto you
        }

        @Override
        protected void onPreExecute() {}

        @Override
        protected void onProgressUpdate(Void... values) {}
    }

    @Override
    public void onBackPressed() { super.onBackPressed(); }
}
```

Communication



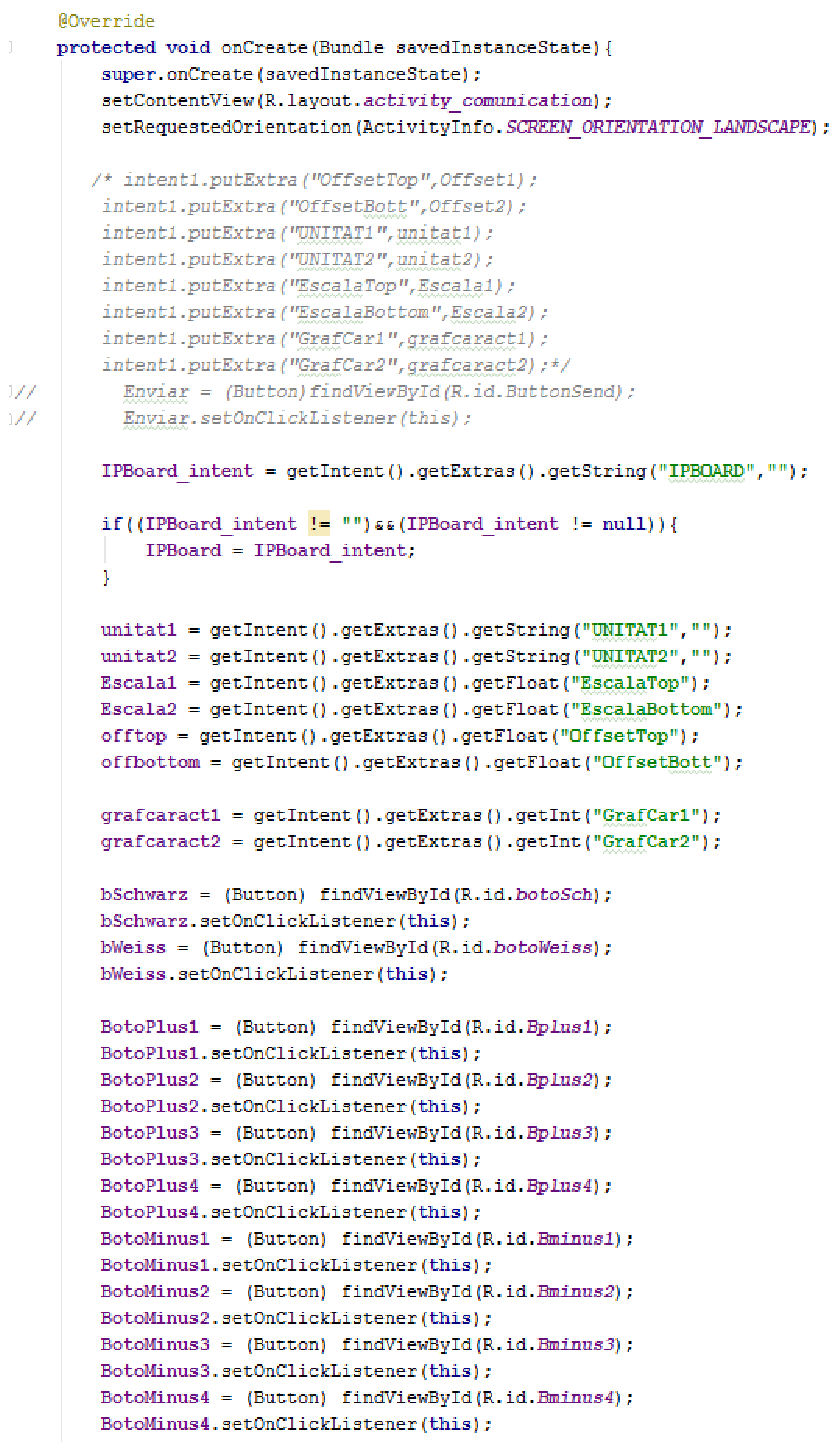
```
package com.aplication.cesc.monitor3_v11;

import ...

public class comunication extends AppCompatActivity implements View.OnClickListener {
    TabHost th2;

    Spinner SPport;
    Spinner SPbaud;
    Spinner SPstopp;
    Spinner SPdaten;
    Spinner SPparitat;

    Spinner SPKanal;
    Spinner SPDiagramm;
    Spinner SPLinks;

    Spinner SPAnzahl;
    Spinner SPNumchart;

    String[] datosSPport = {"No COM ports found","COM-1","COM-2","COM-3","COM-4","COM-5","COM-6"};
    String[] datosSPbaud = {"57600 (user-defined)","110","300","1200","2400","4800","9600","19200",
            "38400","57600","115200","230400","460800","931600"};
    String[] datosSPstopp = {"1","1.5","2"};
    String[] datosSPdaten = {"8","7","6","5"};
    String[] datosSPparitat = {"Keine","Gerade","Ungerade","Markiernung","Leerzeichen"};

    String[] datosSPkanal = {"CH 1","CH 2","CH 3","CH 4","CH 5","CH 6"};
    String[] datosSPDiagramm = {"Diagramm 1","Diagramm 2"};
    String[] datosSPLinks = {"links","rechts"};

    String[] datosSPAnz = {"   1","   2"};

    String[] datosSPNumChart = {"   2","   1"};

    Button bSchwarz;
    Button bWeiss;

    Button BotoPlus1;
    Button BotoPlus2;
    Button BotoPlus3;
    Button BotoPlus4;

    Button BotoMinus1;
    Button BotoMinus2;
    Button BotoMinus3;
    Button BotoMinus4;

    Button BotoSave;

    EditText Scale1;
    EditText Scale2;
    EditText Offset_1;
    EditText Offset_2;
```

```
String Sc1;
String Sc2;
String Off1;
String Off2;

float IntScale1 = 0;
float IntScale2 = 0;
float IntOffset1 = 0;
float IntOffset2 = 0;

EditText Unit1;
EditText Unit2;
String Units1;
String Units2;

float D1SL = 0;
float D1SR = 0;

float D2SL = 0;
float D2SR = 0;

float D1OL = 0;
float D1OR = 0;

float D2OL = 0;
float D2OR = 0;
String UL = "Digits";
String UR = "Digits";

float D1SLCH1 = 0;
float D1SRCH1 = 0;
float D2SLCH1 = 0;
float D2SRCH1 = 0;
float D1OLCH1 = 0;
float D1ORCH1 = 0;
float D2OLCH1 = 0;
float D2ORCH1 = 0;
String D1ULCH1 = "Digits";
String D1URCH1 = "Digits";
String D2ULCH1 = "Digits";
String D2URCH1 = "Digits";

float D1SLCH2 = 0;
float D1SRCH2 = 0;
float D2SLCH2 = 0;
float D2SRCH2 = 0;
float D1OLCH2 = 0;
float D1ORCH2 = 0;
float D2OLCH2 = 0;
float D2ORCH2 = 0;
String D1ULCH2 = "Digits";
String D1URCH2 = "Digits";
String D2ULCH2 = "Digits";
String D2URCH2 = "Digits";
```

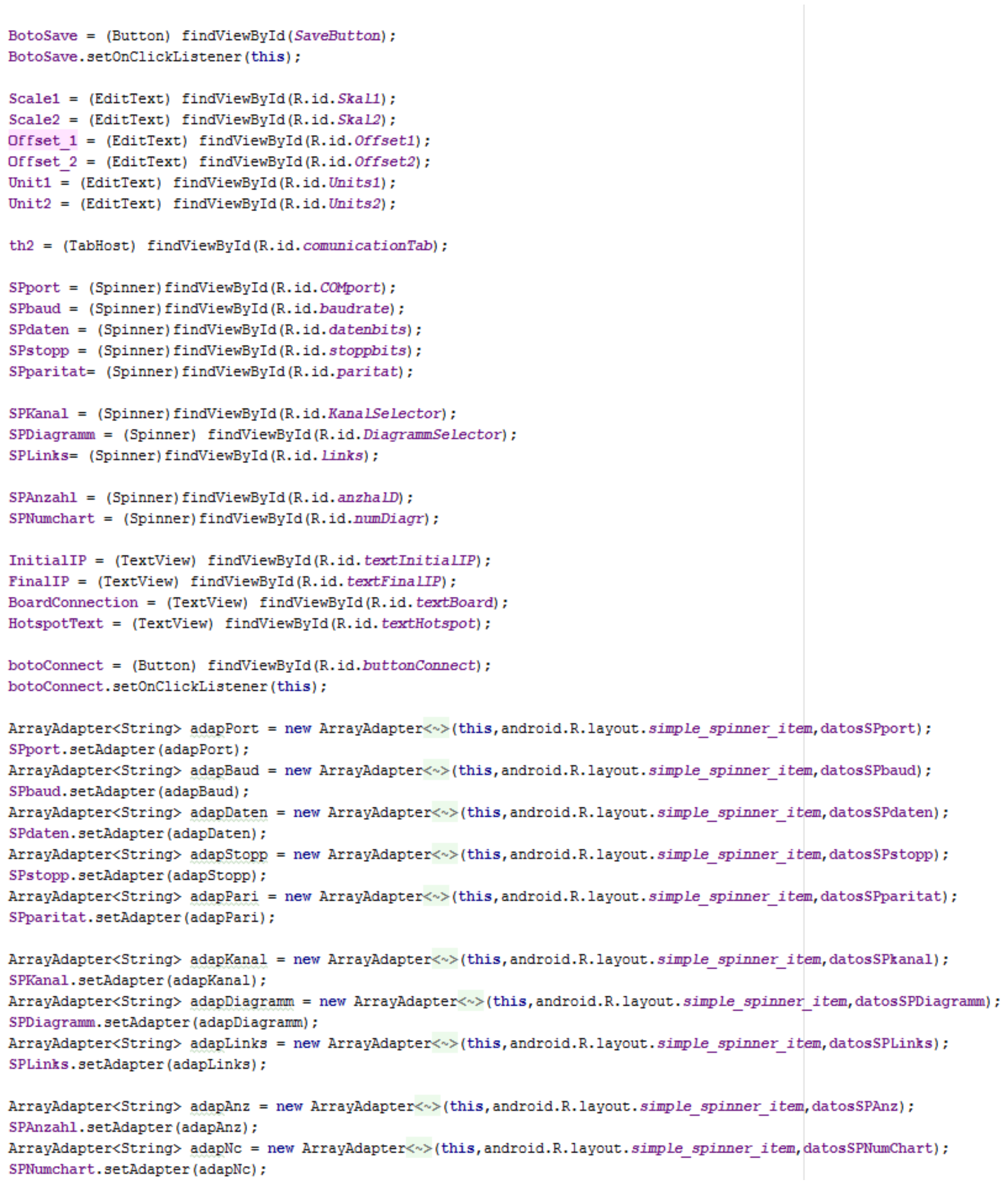


```
float D1SLCH3 = 0;
float D1SRCH3 = 0;
float D2SLCH3 = 0;
float D2SRCH3 = 0;
float D1OLCH3 = 0;
float D1ORCH3 = 0;
float D2OLCH3 = 0;
float D2ORCH3 = 0;
String D1ULCH3 = "Digits";
String D1URCH3 = "Digits";
String D2ULCH3 = "Digits";
String D2URCH3 = "Digits";

float D1SLCH4 = 0;
float D1SRCH4 = 0;
float D2SLCH4 = 0;
float D2SRCH4 = 0;
float D1OLCH4 = 0;
float D1ORCH4 = 0;
float D2OLCH4 = 0;
float D2ORCH4 = 0;
String D1ULCH4 = "Digits";
String D1URCH4 = "Digits";
String D2ULCH4 = "Digits";
String D2URCH4 = "Digits";

float D1SLCH5 = 0;
float D1SRCH5 = 0;
float D2SLCH5 = 0;
float D2SRCH5 = 0;
float D1OLCH5 = 0;
float D1ORCH5 = 0;
float D2OLCH5 = 0;
float D2ORCH5 = 0;
String D1ULCH5 = "Digits";
String D1URCH5 = "Digits";
String D2ULCH5 = "Digits";
String D2URCH5 = "Digits";

float D1SLCH6 = 0;
float D1SRCH6 = 0;
float D2SLCH6 = 0;
float D2SRCH6 = 0;
float D1OLCH6 = 0;
float D1ORCH6 = 0;
float D2OLCH6 = 0;
float D2ORCH6 = 0;
String D1ULCH6 = "Digits";
String D1URCH6 = "Digits";
String D2ULCH6 = "Digits";
String D2URCH6 = "Digits";
```

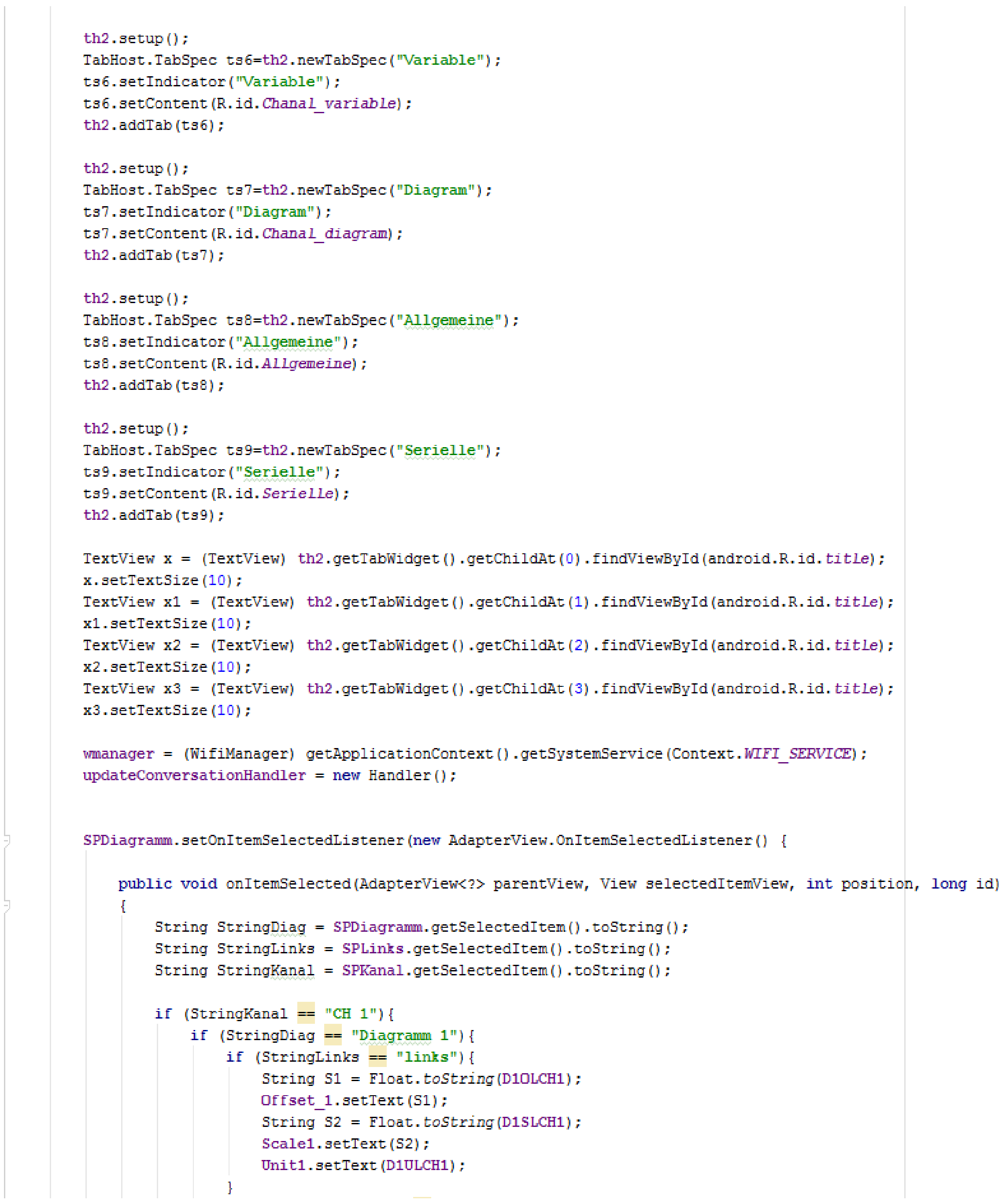
```
int COMport = 0;
int Baud = 0;
int Dbits = 0;
float Stoppbits = 0;
String Paritat;

int GrafCaract1 =0;
int GrafCaract2 =0;
int ChanSelect =0;

boolean colorgraf = false; //boolean for set the background of the graf white (true) or black (f

float Escala1;
float Escala2;
String unitat1;
String unitat2;
float offtop;
float offbottom;

int grafcaract1;
int grafcaract2;

WifiManager wmanager;

int PORT = 4848;
String IP;

private Socket socket;
private ServerSocket serverSocket;

Handler updateConversationHandler;

Thread serverThread = null;
Thread clientThread = null;
int SC = 0;
String str;

TextView InitialIP;
TextView FinalIP;
TextView BoardConnection;
TextView HotspotText;
Button botoConnect;

String IP1;
String IP2;
boolean BoardDetected;
String IPBoard;
String IPBoard_intent;

String WifiId;
```



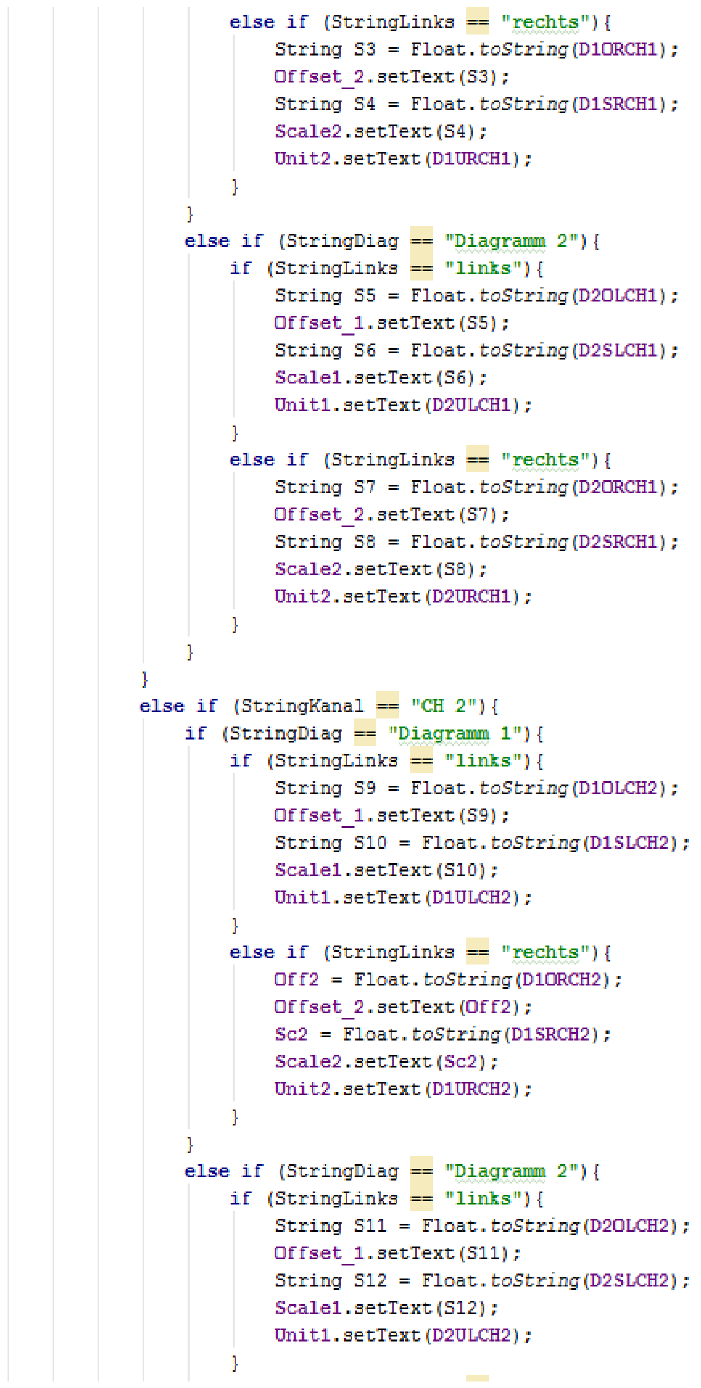
```
@Override
protected void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_comunication);
    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);

   /* intent1.putExtra("OffsetTop",Offset1);
    intent1.putExtra("OffsetBott",Offset2);
    intent1.putExtra("UNITAT1",unitat1);
    intent1.putExtra("UNITAT2",unitat2);
    intent1.putExtra("EscalaTop",Escala1);
    intent1.putExtra("EscalaBottom",Escala2);
    intent1.putExtra("GrafCar1",grafcaract1);
    intent1.putExtra("GrafCar2",grafcaract2);*/
//        Enviar = (Button)findViewById(R.id.ButtonSend);
//        Enviar.setOnClickListener(this);

    IPBoard_intent = getIntent().getExtras().getString("IPBOARD","");

    if((IPBoard_intent != "")&&(IPBoard_intent != null)){
        IPBoard = IPBoard_intent;
    }

    unitat1 = getIntent().getExtras().getString("UNITAT1","");
    unitat2 = getIntent().getExtras().getString("UNITAT2","");
    Escala1 = getIntent().getExtras().getFloat("EscalaTop");
    Escala2 = getIntent().getExtras().getFloat("EscalaBottom");
    offtop = getIntent().getExtras().getFloat("OffsetTop");
    offbottom = getIntent().getExtras().getFloat("OffsetBott");

    grafcaract1 = getIntent().getExtras().getInt("GrafCar1");
    grafcaract2 = getIntent().getExtras().getInt("GrafCar2");

    bSchwarz = (Button) findViewById(R.id.botoSch);
    bSchwarz.setOnClickListener(this);
    bWeiss = (Button) findViewById(R.id.botoWeiss);
    bWeiss.setOnClickListener(this);

    BotoPlus1 = (Button) findViewById(R.id.Bplus1);
    BotoPlus1.setOnClickListener(this);
    BotoPlus2 = (Button) findViewById(R.id.Bplus2);
    BotoPlus2.setOnClickListener(this);
    BotoPlus3 = (Button) findViewById(R.id.Bplus3);
    BotoPlus3.setOnClickListener(this);
    BotoPlus4 = (Button) findViewById(R.id.Bplus4);
    BotoPlus4.setOnClickListener(this);
    BotoMinus1 = (Button) findViewById(R.id.Bminus1);
    BotoMinus1.setOnClickListener(this);
    BotoMinus2 = (Button) findViewById(R.id.Bminus2);
    BotoMinus2.setOnClickListener(this);
    BotoMinus3 = (Button) findViewById(R.id.Bminus3);
    BotoMinus3.setOnClickListener(this);
    BotoMinus4 = (Button) findViewById(R.id.Bminus4);
    BotoMinus4.setOnClickListener(this);
```



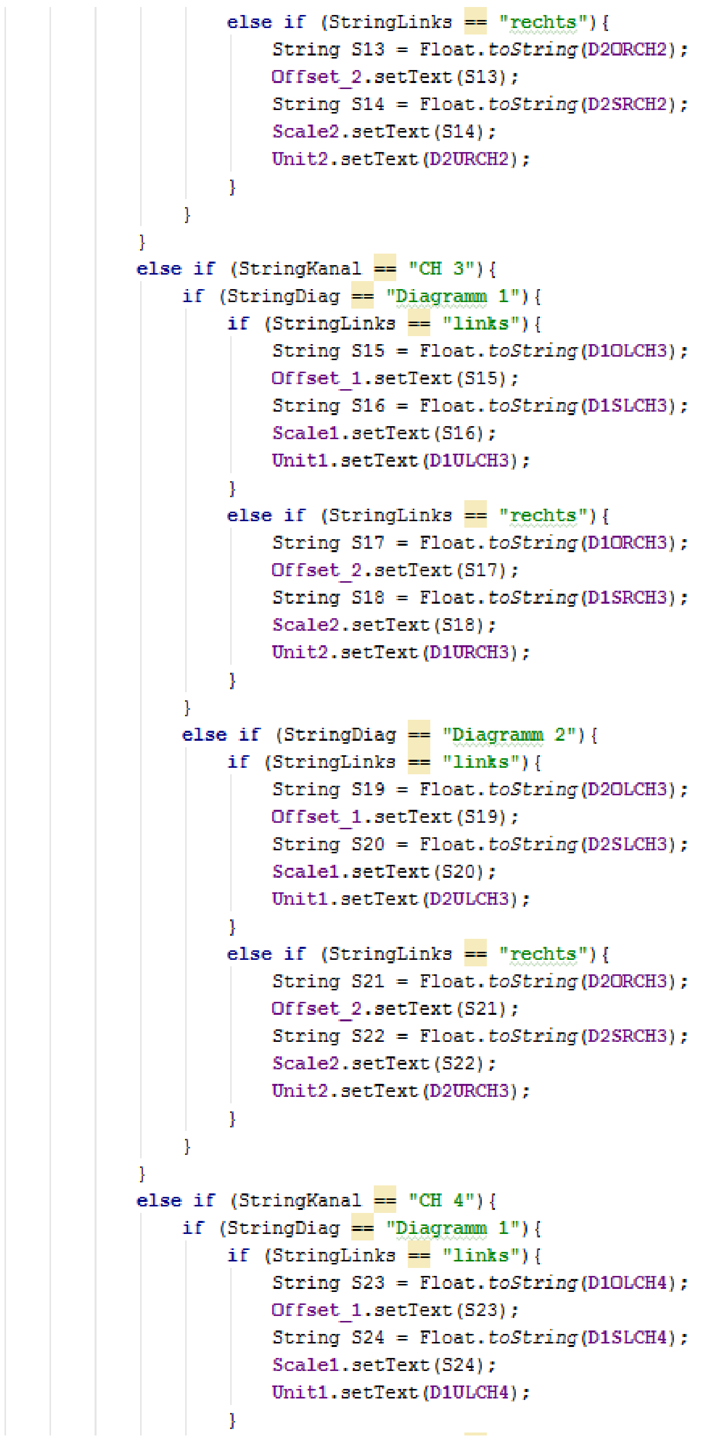
```
BotoSave = (Button) findViewById(SaveButton);
BotoSave.setOnClickListener(this);

Scale1 = (EditText) findViewById(R.id.Skal1);
Scale2 = (EditText) findViewById(R.id.Skal2);
Offset_1 = (EditText) findViewById(R.id.Offset1);
Offset_2 = (EditText) findViewById(R.id.Offset2);
Unit1 = (EditText) findViewById(R.id.Units1);
Unit2 = (EditText) findViewById(R.id.Units2);

th2 = (TabHost) findViewById(R.id.comunicationTab);

SPport = (Spinner)findViewById(R.id.COMport);
SPbaud = (Spinner)findViewById(R.id.baudrate);
SPdaten = (Spinner)findViewById(R.id.datenbits);
SPstopp = (Spinner)findViewById(R.id.stoppbits);
SPparitat= (Spinner)findViewById(R.id.paritat);

SPKanal = (Spinner)findViewById(R.id.KanalSelector);
SPDiagramm = (Spinner) findViewById(R.id.DiagrammSelector);
SPLinks= (Spinner)findViewById(R.id.links);

SPAnzahl = (Spinner)findViewById(R.id.anzhalD);
SPNumchart = (Spinner)findViewById(R.id.numDiagr);

InitialIP = (TextView) findViewById(R.id.textInitialIP);
FinalIP = (TextView) findViewById(R.id.textFinalIP);
BoardConnection = (TextView) findViewById(R.id.textBoard);
HotspotText = (TextView) findViewById(R.id.textHotspot);

botoConnect = (Button) findViewById(R.id.buttonConnect);
botoConnect.setOnClickListener(this);

ArrayAdapter<String> adapPort = new ArrayAdapter<~>(this,android.R.layout.simple_spinner_item,datosSPport);
SPport.setAdapter(adapPort);
ArrayAdapter<String> adapBaud = new ArrayAdapter<~>(this,android.R.layout.simple_spinner_item,datosSPbaud);
SPbaud.setAdapter(adapBaud);
ArrayAdapter<String> adapDaten = new ArrayAdapter<~>(this,android.R.layout.simple_spinner_item,datosSPdaten);
SPdaten.setAdapter(adapDaten);
ArrayAdapter<String> adapStopp = new ArrayAdapter<~>(this,android.R.layout.simple_spinner_item,datosSPstopp);
SPstopp.setAdapter(adapStopp);
ArrayAdapter<String> adapPari = new ArrayAdapter<~>(this,android.R.layout.simple_spinner_item,datosSPparitat);
SPparitat.setAdapter(adapPari);

ArrayAdapter<String> adapKanal = new ArrayAdapter<~>(this,android.R.layout.simple_spinner_item,datosSPkanal);
SPKanal.setAdapter(adapKanal);
ArrayAdapter<String> adapDiagramm = new ArrayAdapter<~>(this,android.R.layout.simple_spinner_item,datosSPDiagramm);
SPDiagramm.setAdapter(adapDiagramm);
ArrayAdapter<String> adapLinks = new ArrayAdapter<~>(this,android.R.layout.simple_spinner_item,datosSPLinks);
SPLinks.setAdapter(adapLinks);

ArrayAdapter<String> adapAnz = new ArrayAdapter<~>(this,android.R.layout.simple_spinner_item,datosSPAnz);
SPAnzahl.setAdapter(adapAnz);
ArrayAdapter<String> adapNc = new ArrayAdapter<~>(this,android.R.layout.simple_spinner_item,datosSPNumChart);
SPNumchart.setAdapter(adapNc);
```



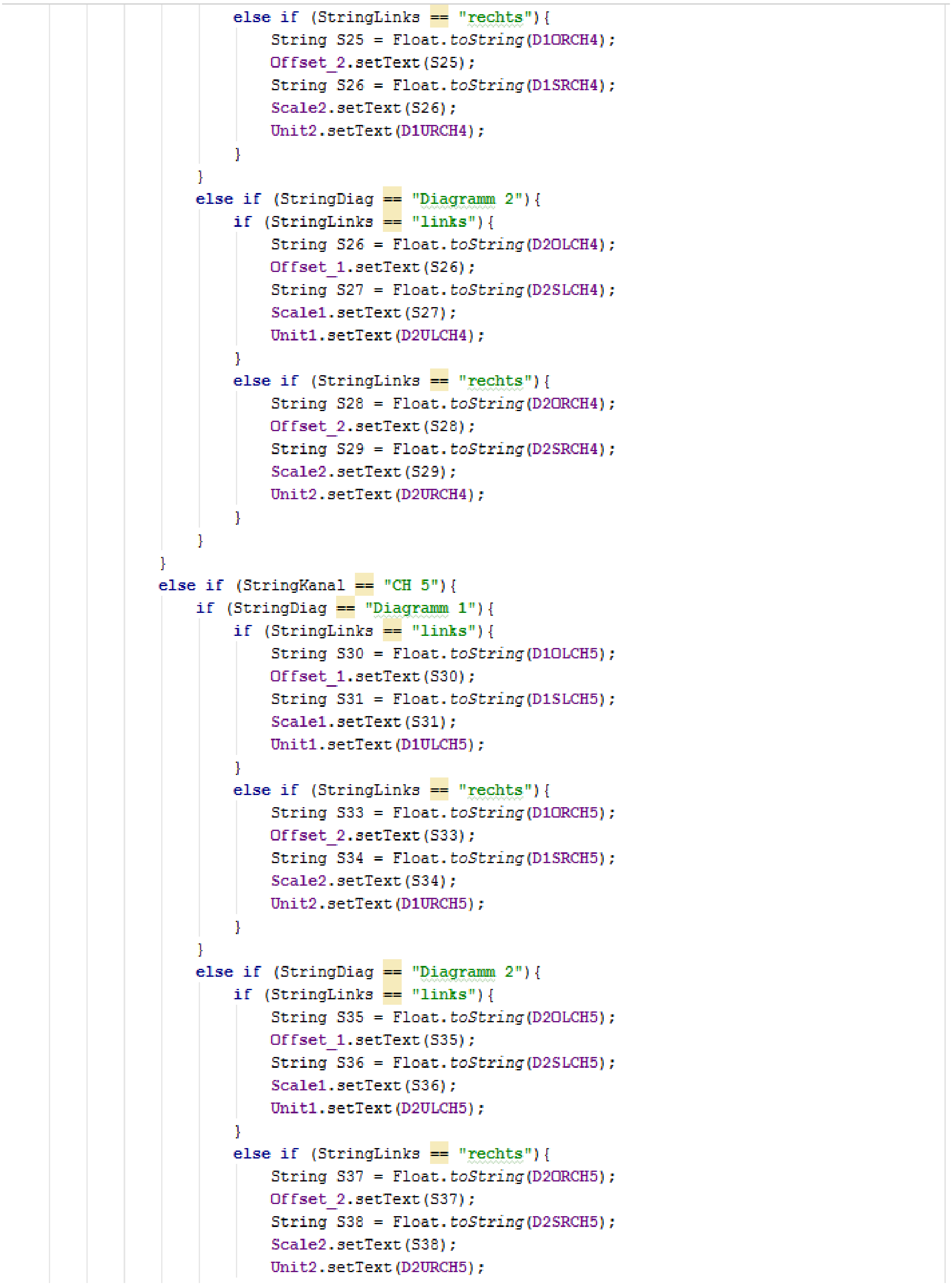
```
th2.setup();
TabHost.TabSpec ts6=th2.newTabSpec("Variable");
ts6.setIndicator("Variable");
ts6.setContent(R.id.Chanal_variable);
th2.addTab(ts6);

th2.setup();
TabHost.TabSpec ts7=th2.newTabSpec("Diagram");
ts7.setIndicator("Diagram");
ts7.setContent(R.id.Chanal_diagram);
th2.addTab(ts7);

th2.setup();
TabHost.TabSpec ts8=th2.newTabSpec("Allgemeine");
ts8.setIndicator("Allgemeine");
ts8.setContent(R.id.Allgemeine);
th2.addTab(ts8);

th2.setup();
TabHost.TabSpec ts9=th2.newTabSpec("Serielle");
ts9.setIndicator("Serielle");
ts9.setContent(R.id.Serielle);
th2.addTab(ts9);

TextView x = (TextView) th2.getTabWidget().getChildAt(0).findViewById(android.R.id.title);
x.setTextSize(10);
TextView x1 = (TextView) th2.getTabWidget().getChildAt(1).findViewById(android.R.id.title);
x1.setTextSize(10);
TextView x2 = (TextView) th2.getTabWidget().getChildAt(2).findViewById(android.R.id.title);
x2.setTextSize(10);
TextView x3 = (TextView) th2.getTabWidget().getChildAt(3).findViewById(android.R.id.title);
x3.setTextSize(10);

wmanager = (WifiManager) getApplicationContext().getSystemService(Context.WIFI_SERVICE);
updateConversationHandler = new Handler();


SPDiagramm.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

    public void onItemSelected(AdapterView<?> parentView, View selectedItemView, int position, long id)
    {
        String StringDiag = SPDiagramm.getSelectedItem().toString();
        String StringLinks = SPLinks.getSelectedItem().toString();
        String StringKanal = SPKanal.getSelectedItem().toString();

        if (StringKanal == "CH 1"){
            if (StringDiag == "Diagramm 1"){
                if (StringLinks == "links"){
                    String S1 = Float.toString(D1OLCH1);
                    Offset_1.setText(S1);
                    String S2 = Float.toString(D1SLCH1);
                    Scale1.setText(S2);
                    Unit1.setText(D1ULCH1);
                }
```



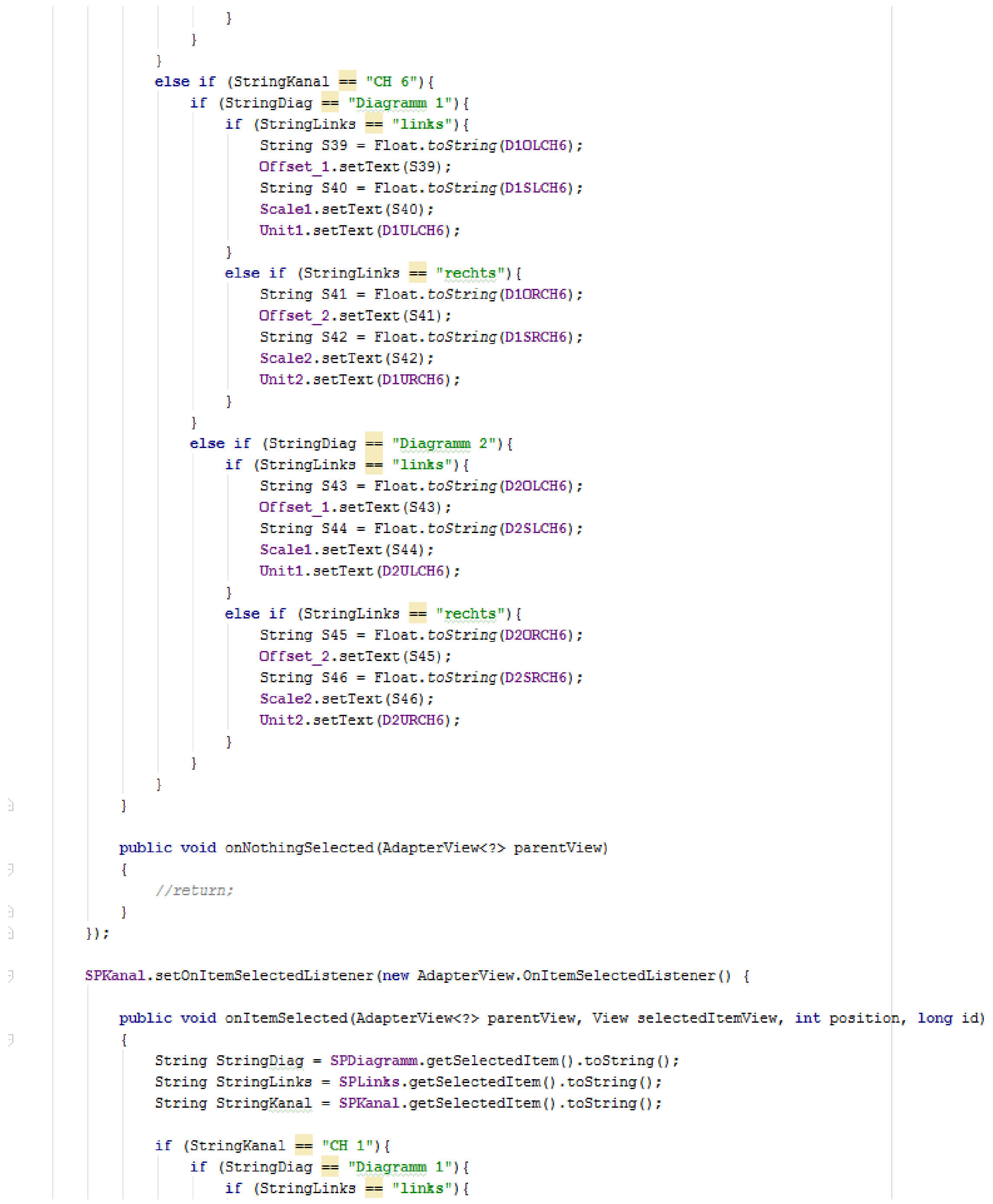
```
                    else if (StringLinks == "rechts"){
                        String S3 = Float.toString(D1ORCH1);
                        Offset_2.setText(S3);
                        String S4 = Float.toString(D1SRCH1);
                        Scale2.setText(S4);
                        Unit2.setText(D1URCH1);
                    }
                }
                else if (StringDiag == "Diagramm 2"){
                    if (StringLinks == "links"){
                        String S5 = Float.toString(D2OLCH1);
                        Offset_1.setText(S5);
                        String S6 = Float.toString(D2SLCH1);
                        Scale1.setText(S6);
                        Unit1.setText(D2ULCH1);
                    }
                    else if (StringLinks == "rechts"){
                        String S7 = Float.toString(D2ORCH1);
                        Offset_2.setText(S7);
                        String S8 = Float.toString(D2SRCH1);
                        Scale2.setText(S8);
                        Unit2.setText(D2URCH1);
                    }
                }
            }
            else if (StringKanal == "CH 2"){
                if (StringDiag == "Diagramm 1"){
                    if (StringLinks == "links"){
                        String S9 = Float.toString(D1OLCH2);
                        Offset_1.setText(S9);
                        String S10 = Float.toString(D1SLCH2);
                        Scale1.setText(S10);
                        Unit1.setText(D1ULCH2);
                    }
                    else if (StringLinks == "rechts"){
                        Off2 = Float.toString(D1ORCH2);
                        Offset_2.setText(Off2);
                        Sc2 = Float.toString(D1SRCH2);
                        Scale2.setText(Sc2);
                        Unit2.setText(D1URCH2);
                    }
                }
                else if (StringDiag == "Diagramm 2"){
                    if (StringLinks == "links"){
                        String S11 = Float.toString(D2OLCH2);
                        Offset_1.setText(S11);
                        String S12 = Float.toString(D2SLCH2);
                        Scale1.setText(S12);
                        Unit1.setText(D2ULCH2);
                    }
```



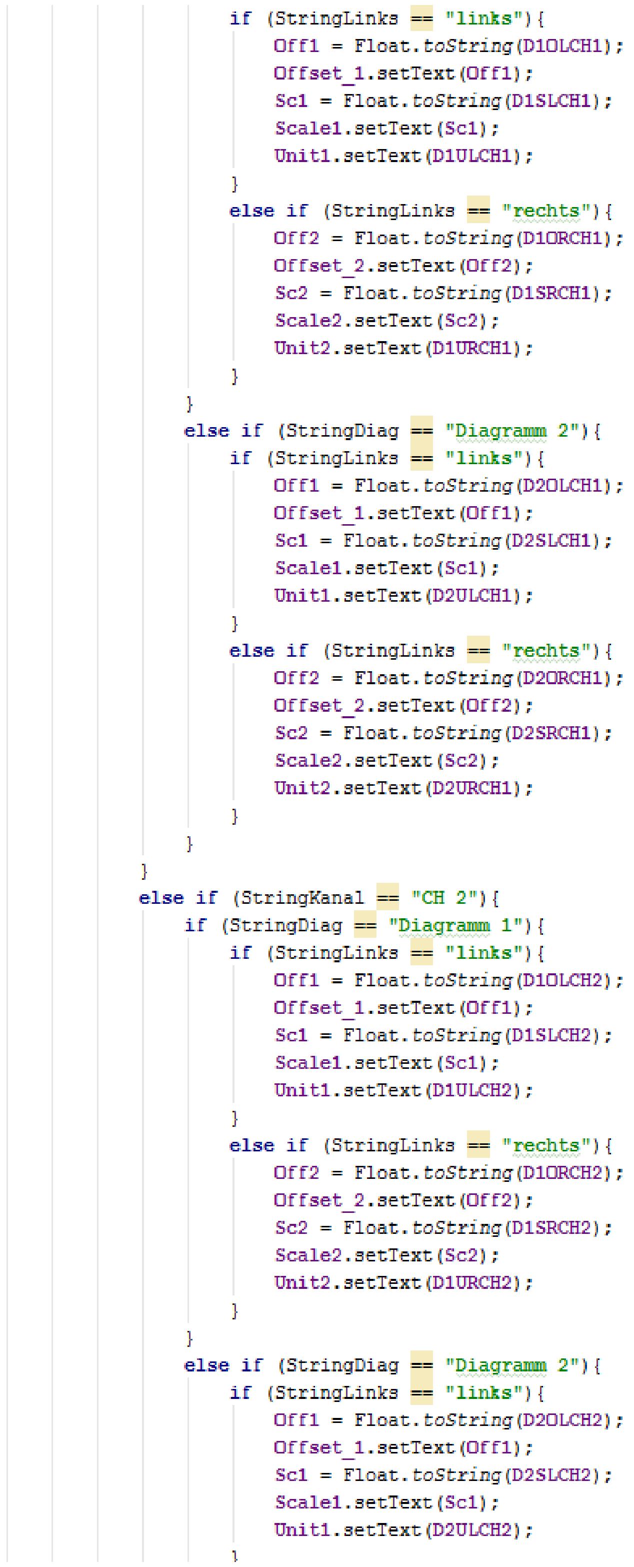
```
                    else if (StringLinks == "rechts"){
                        String S13 = Float.toString(D2ORCH2);
                        Offset_2.setText(S13);
                        String S14 = Float.toString(D2SRCH2);
                        Scale2.setText(S14);
                        Unit2.setText(D2URCH2);
                    }
                }
            }
            else if (StringKanal == "CH 3"){
                if (StringDiag == "Diagramm 1"){
                    if (StringLinks == "links"){
                        String S15 = Float.toString(D1OLCH3);
                        Offset_1.setText(S15);
                        String S16 = Float.toString(D1SLCH3);
                        Scale1.setText(S16);
                        Unit1.setText(D1ULCH3);
                    }
                    else if (StringLinks == "rechts"){
                        String S17 = Float.toString(D1ORCH3);
                        Offset_2.setText(S17);
                        String S18 = Float.toString(D1SRCH3);
                        Scale2.setText(S18);
                        Unit2.setText(D1URCH3);
                    }
                }
                else if (StringDiag == "Diagramm 2"){
                    if (StringLinks == "links"){
                        String S19 = Float.toString(D2OLCH3);
                        Offset_1.setText(S19);
                        String S20 = Float.toString(D2SLCH3);
                        Scale1.setText(S20);
                        Unit1.setText(D2ULCH3);
                    }
                    else if (StringLinks == "rechts"){
                        String S21 = Float.toString(D2ORCH3);
                        Offset_2.setText(S21);
                        String S22 = Float.toString(D2SRCH3);
                        Scale2.setText(S22);
                        Unit2.setText(D2URCH3);
                    }
                }
            }
            else if (StringKanal == "CH 4"){
                if (StringDiag == "Diagramm 1"){
                    if (StringLinks == "links"){
                        String S23 = Float.toString(D1OLCH4);
                        Offset_1.setText(S23);
                        String S24 = Float.toString(D1SLCH4);
                        Scale1.setText(S24);
                        Unit1.setText(D1ULCH4);
                    }
```



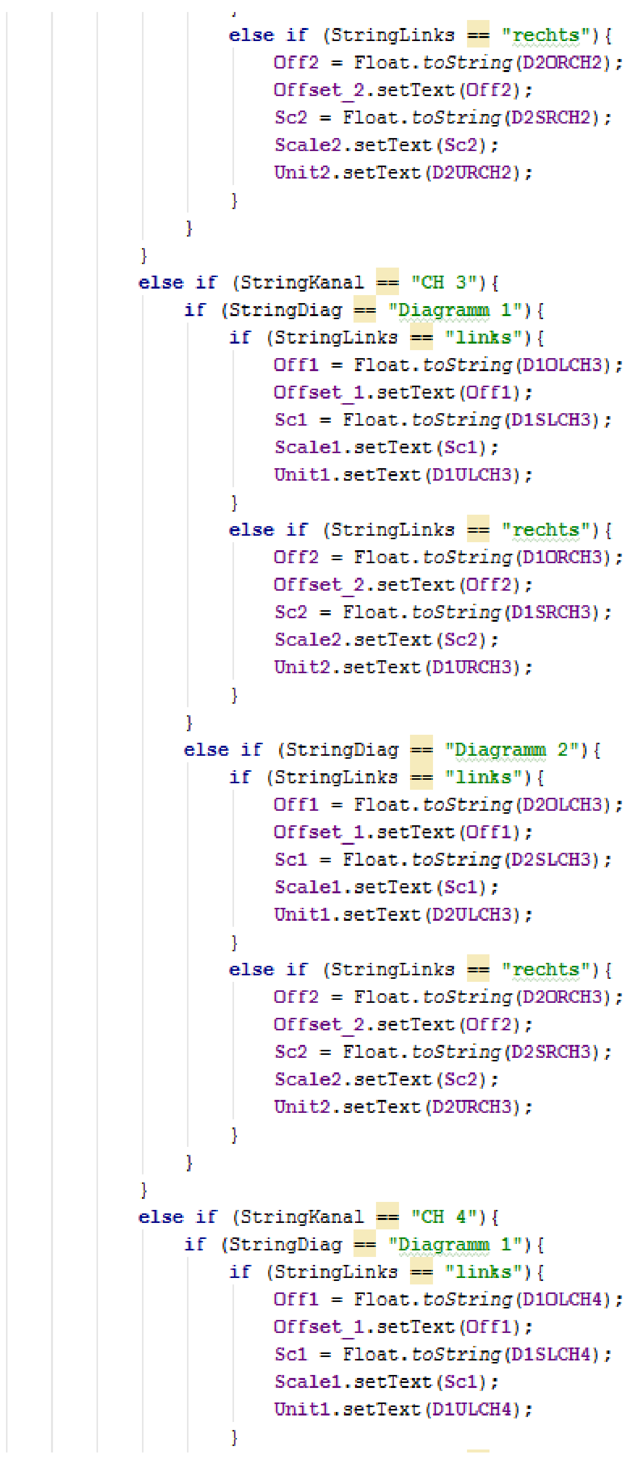
```
                        else if (StringLinks == "rechts"){
                            String S25 = Float.toString(D1ORCH4);
                            Offset_2.setText(S25);
                            String S26 = Float.toString(D1SRCH4);
                            Scale2.setText(S26);
                            Unit2.setText(D1URCH4);
                        }
                    }
                    else if (StringDiag == "Diagramm 2"){
                        if (StringLinks == "links"){
                            String S26 = Float.toString(D2OLCH4);
                            Offset_1.setText(S26);
                            String S27 = Float.toString(D2SLCH4);
                            Scale1.setText(S27);
                            Unit1.setText(D2ULCH4);
                        }
                        else if (StringLinks == "rechts"){
                            String S28 = Float.toString(D2ORCH4);
                            Offset_2.setText(S28);
                            String S29 = Float.toString(D2SRCH4);
                            Scale2.setText(S29);
                            Unit2.setText(D2URCH4);
                        }
                    }
                }
                else if (StringKanal == "CH 5"){
                    if (StringDiag == "Diagramm 1"){
                        if (StringLinks == "links"){
                            String S30 = Float.toString(D1OLCH5);
                            Offset_1.setText(S30);
                            String S31 = Float.toString(D1SLCH5);
                            Scale1.setText(S31);
                            Unit1.setText(D1ULCH5);
                        }
                        else if (StringLinks == "rechts"){
                            String S33 = Float.toString(D1ORCH5);
                            Offset_2.setText(S33);
                            String S34 = Float.toString(D1SRCH5);
                            Scale2.setText(S34);
                            Unit2.setText(D1URCH5);
                        }
                    }
                    else if (StringDiag == "Diagramm 2"){
                        if (StringLinks == "links"){
                            String S35 = Float.toString(D2OLCH5);
                            Offset_1.setText(S35);
                            String S36 = Float.toString(D2SLCH5);
                            Scale1.setText(S36);
                            Unit1.setText(D2ULCH5);
                        }
                        else if (StringLinks == "rechts"){
                            String S37 = Float.toString(D2ORCH5);
                            Offset_2.setText(S37);
                            String S38 = Float.toString(D2SRCH5);
                            Scale2.setText(S38);
                            Unit2.setText(D2URCH5);
```



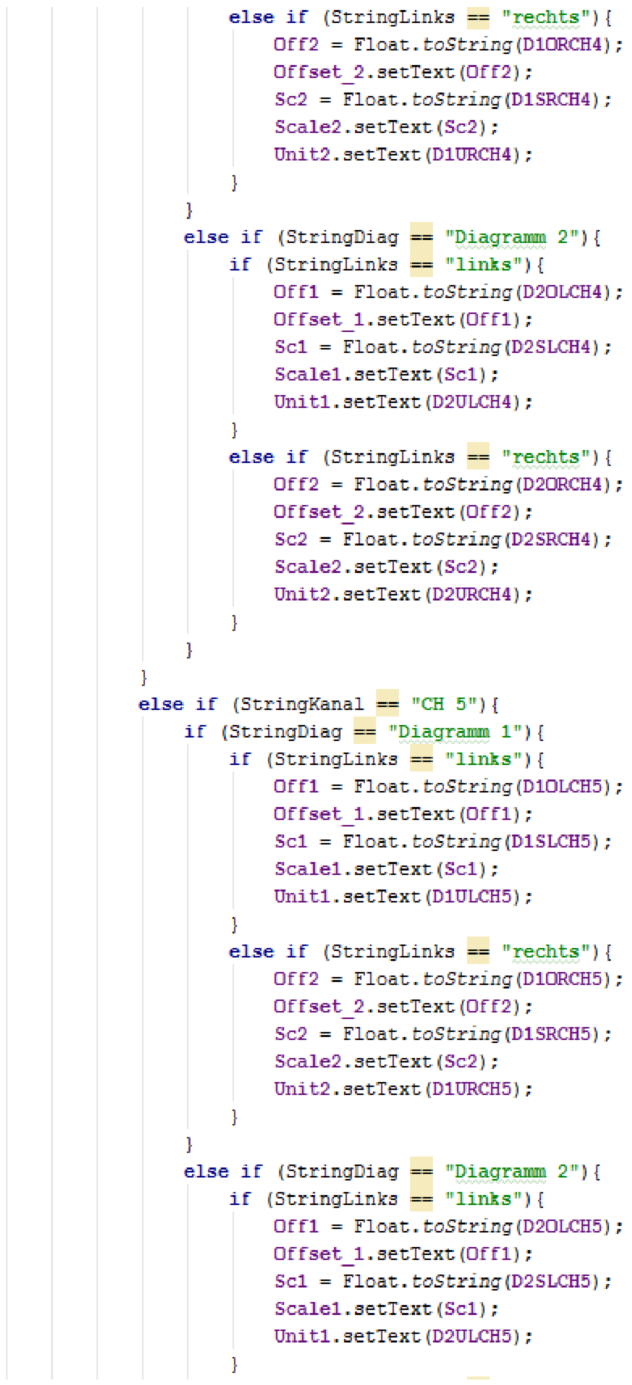
```
                }
            }
        }
        else if (StringKanal == "CH 6"){
            if (StringDiag == "Diagramm 1"){
                if (StringLinks == "links"){
                    String S39 = Float.toString(D1OLCH6);
                    Offset_1.setText(S39);
                    String S40 = Float.toString(D1SLCH6);
                    Scale1.setText(S40);
                    Unit1.setText(D1ULCH6);
                }
                else if (StringLinks == "rechts"){
                    String S41 = Float.toString(D1ORCH6);
                    Offset_2.setText(S41);
                    String S42 = Float.toString(D1SRCH6);
                    Scale2.setText(S42);
                    Unit2.setText(D1URCH6);
                }
            }
            else if (StringDiag == "Diagramm 2"){
                if (StringLinks == "links"){
                    String S43 = Float.toString(D2OLCH6);
                    Offset_1.setText(S43);
                    String S44 = Float.toString(D2SLCH6);
                    Scale1.setText(S44);
                    Unit1.setText(D2ULCH6);
                }
                else if (StringLinks == "rechts"){
                    String S45 = Float.toString(D2ORCH6);
                    Offset_2.setText(S45);
                    String S46 = Float.toString(D2SRCH6);
                    Scale2.setText(S46);
                    Unit2.setText(D2URCH6);
                }
            }
        }
    }

    public void onNothingSelected(AdapterView<?> parentView)
    {
        //return;
    }
});

SPKanal.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

    public void onItemSelected(AdapterView<?> parentView, View selectedItemView, int position, long id)
    {
        String StringDiag = SPDiagramm.getSelectedItem().toString();
        String StringLinks = SPLinks.getSelectedItem().toString();
        String StringKanal = SPKanal.getSelectedItem().toString();

        if (StringKanal == "CH 1"){
            if (StringDiag == "Diagramm 1"){
                if (StringLinks == "links"){
```



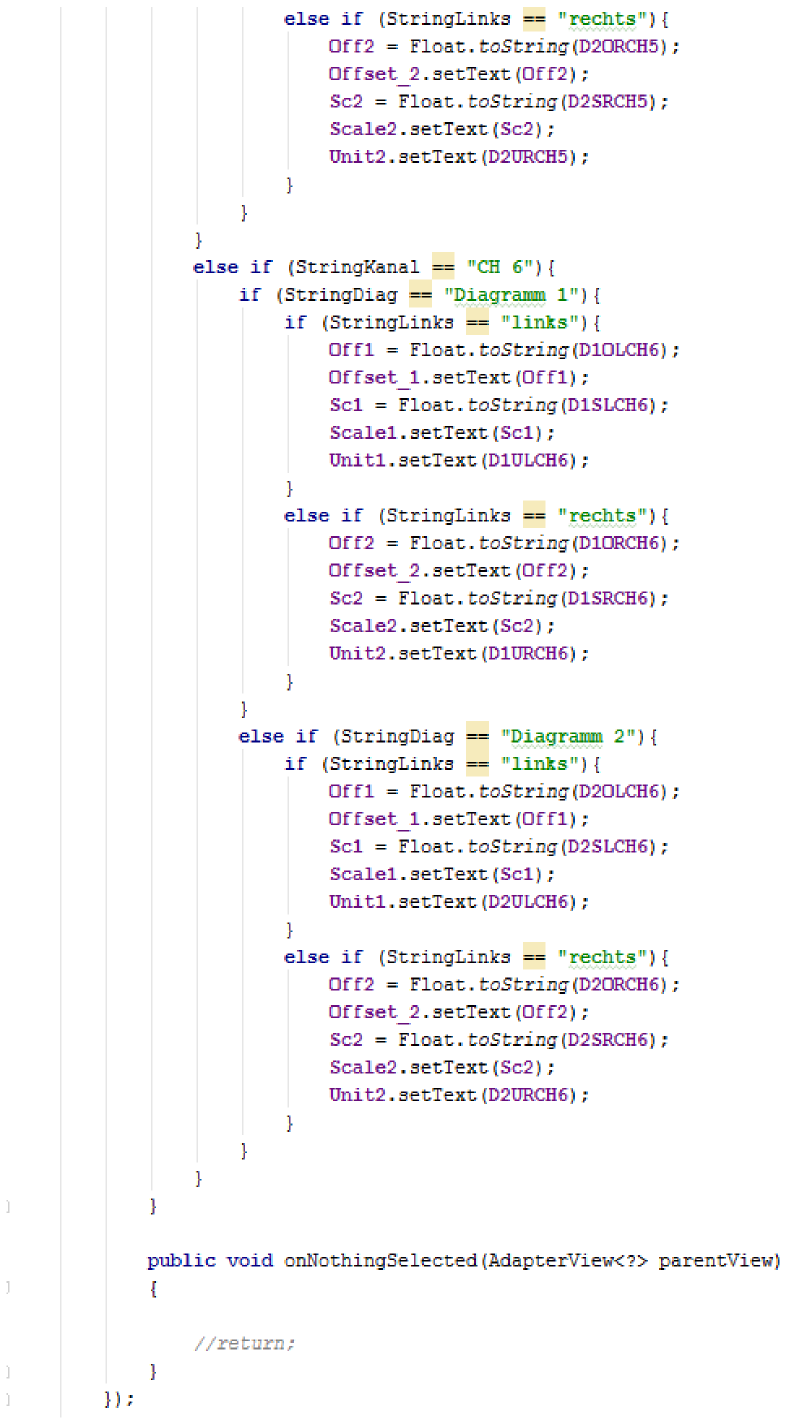
```
        if (StringLinks == "links"){
            Off1 = Float.toString(D1OLCH1);
            Offset_1.setText(Off1);
            Sc1 = Float.toString(D1SLCH1);
            Scale1.setText(Sc1);
            Unit1.setText(D1ULCH1);
        }
        else if (StringLinks == "rechts"){
            Off2 = Float.toString(D1ORCH1);
            Offset_2.setText(Off2);
            Sc2 = Float.toString(D1SRCH1);
            Scale2.setText(Sc2);
            Unit2.setText(D1URCH1);
        }
    }
    else if (StringDiag == "Diagramm 2"){
        if (StringLinks == "links"){
            Off1 = Float.toString(D2OLCH1);
            Offset_1.setText(Off1);
            Sc1 = Float.toString(D2SLCH1);
            Scale1.setText(Sc1);
            Unit1.setText(D2ULCH1);
        }
        else if (StringLinks == "rechts"){
            Off2 = Float.toString(D2ORCH1);
            Offset_2.setText(Off2);
            Sc2 = Float.toString(D2SRCH1);
            Scale2.setText(Sc2);
            Unit2.setText(D2URCH1);
        }
    }
}
else if (StringKanal == "CH 2"){
    if (StringDiag == "Diagramm 1"){
        if (StringLinks == "links"){
            Off1 = Float.toString(D1OLCH2);
            Offset_1.setText(Off1);
            Sc1 = Float.toString(D1SLCH2);
            Scale1.setText(Sc1);
            Unit1.setText(D1ULCH2);
        }
        else if (StringLinks == "rechts"){
            Off2 = Float.toString(D1ORCH2);
            Offset_2.setText(Off2);
            Sc2 = Float.toString(D1SRCH2);
            Scale2.setText(Sc2);
            Unit2.setText(D1URCH2);
        }
    }
    else if (StringDiag == "Diagramm 2"){
        if (StringLinks == "links"){
            Off1 = Float.toString(D2OLCH2);
            Offset_1.setText(Off1);
            Sc1 = Float.toString(D2SLCH2);
            Scale1.setText(Sc1);
            Unit1.setText(D2ULCH2);
        }
```



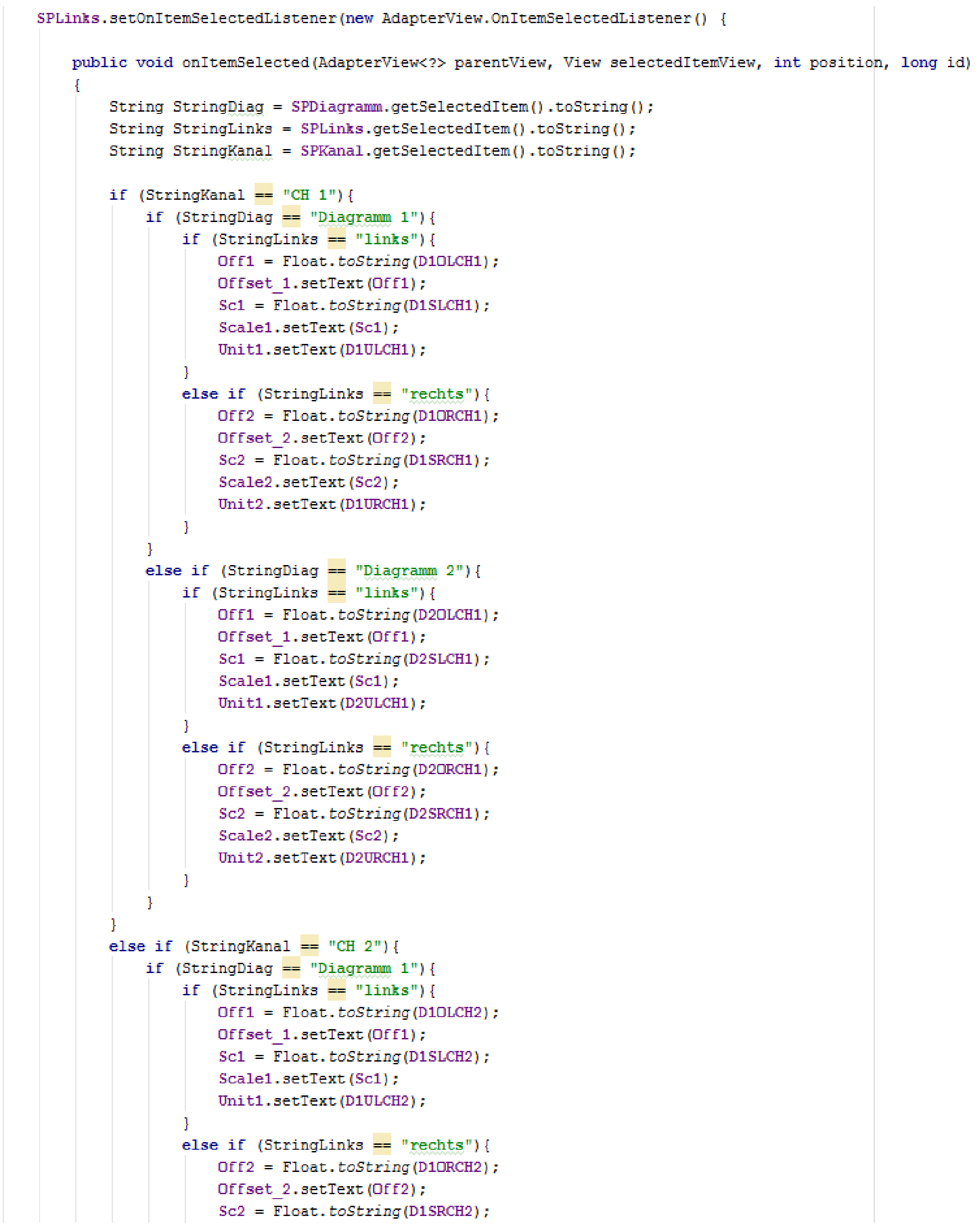
```
                    else if (StringLinks == "rechts"){
                        Off2 = Float.toString(D2ORCH2);
                        Offset_2.setText(Off2);
                        Sc2 = Float.toString(D2SRCH2);
                        Scale2.setText(Sc2);
                        Unit2.setText(D2URCH2);
                    }
                }
            }
            else if (StringKanal == "CH 3"){
                if (StringDiag == "Diagramm 1"){
                    if (StringLinks == "links"){
                        Off1 = Float.toString(D1OLCH3);
                        Offset_1.setText(Off1);
                        Sc1 = Float.toString(D1SLCH3);
                        Scale1.setText(Sc1);
                        Unit1.setText(D1ULCH3);
                    }
                    else if (StringLinks == "rechts"){
                        Off2 = Float.toString(D1ORCH3);
                        Offset_2.setText(Off2);
                        Sc2 = Float.toString(D1SRCH3);
                        Scale2.setText(Sc2);
                        Unit2.setText(D1URCH3);
                    }
                }
                else if (StringDiag == "Diagramm 2"){
                    if (StringLinks == "links"){
                        Off1 = Float.toString(D2OLCH3);
                        Offset_1.setText(Off1);
                        Sc1 = Float.toString(D2SLCH3);
                        Scale1.setText(Sc1);
                        Unit1.setText(D2ULCH3);
                    }
                    else if (StringLinks == "rechts"){
                        Off2 = Float.toString(D2ORCH3);
                        Offset_2.setText(Off2);
                        Sc2 = Float.toString(D2SRCH3);
                        Scale2.setText(Sc2);
                        Unit2.setText(D2URCH3);
                    }
                }
            }
            else if (StringKanal == "CH 4"){
                if (StringDiag == "Diagramm 1"){
                    if (StringLinks == "links"){
                        Off1 = Float.toString(D1OLCH4);
                        Offset_1.setText(Off1);
                        Sc1 = Float.toString(D1SLCH4);
                        Scale1.setText(Sc1);
                        Unit1.setText(D1ULCH4);
                    }
```



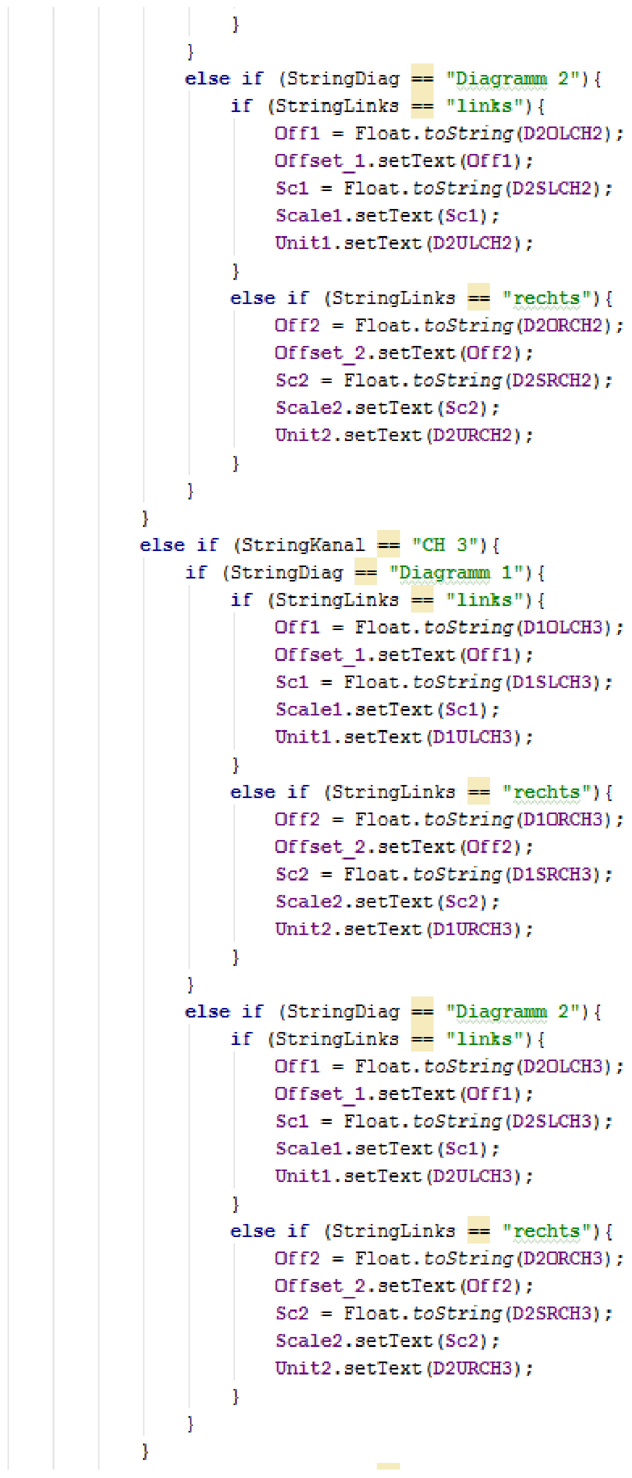
```
                    else if (StringLinks == "rechts"){
                        Off2 = Float.toString(D1ORCH4);
                        Offset_2.setText(Off2);
                        Sc2 = Float.toString(D1SRCH4);
                        Scale2.setText(Sc2);
                        Unit2.setText(D1URCH4);
                    }
                }
                else if (StringDiag == "Diagramm 2"){
                    if (StringLinks == "links"){
                        Off1 = Float.toString(D2OLCH4);
                        Offset_1.setText(Off1);
                        Sc1 = Float.toString(D2SLCH4);
                        Scale1.setText(Sc1);
                        Unit1.setText(D2ULCH4);
                    }
                    else if (StringLinks == "rechts"){
                        Off2 = Float.toString(D2ORCH4);
                        Offset_2.setText(Off2);
                        Sc2 = Float.toString(D2SRCH4);
                        Scale2.setText(Sc2);
                        Unit2.setText(D2URCH4);
                    }
                }
            }
            else if (StringKanal == "CH 5"){
                if (StringDiag == "Diagramm 1"){
                    if (StringLinks == "links"){
                        Off1 = Float.toString(D1OLCH5);
                        Offset_1.setText(Off1);
                        Sc1 = Float.toString(D1SLCH5);
                        Scale1.setText(Sc1);
                        Unit1.setText(D1ULCH5);
                    }
                    else if (StringLinks == "rechts"){
                        Off2 = Float.toString(D1ORCH5);
                        Offset_2.setText(Off2);
                        Sc2 = Float.toString(D1SRCH5);
                        Scale2.setText(Sc2);
                        Unit2.setText(D1URCH5);
                    }
                }
                else if (StringDiag == "Diagramm 2"){
                    if (StringLinks == "links"){
                        Off1 = Float.toString(D2OLCH5);
                        Offset_1.setText(Off1);
                        Sc1 = Float.toString(D2SLCH5);
                        Scale1.setText(Sc1);
                        Unit1.setText(D2ULCH5);
                    }
```



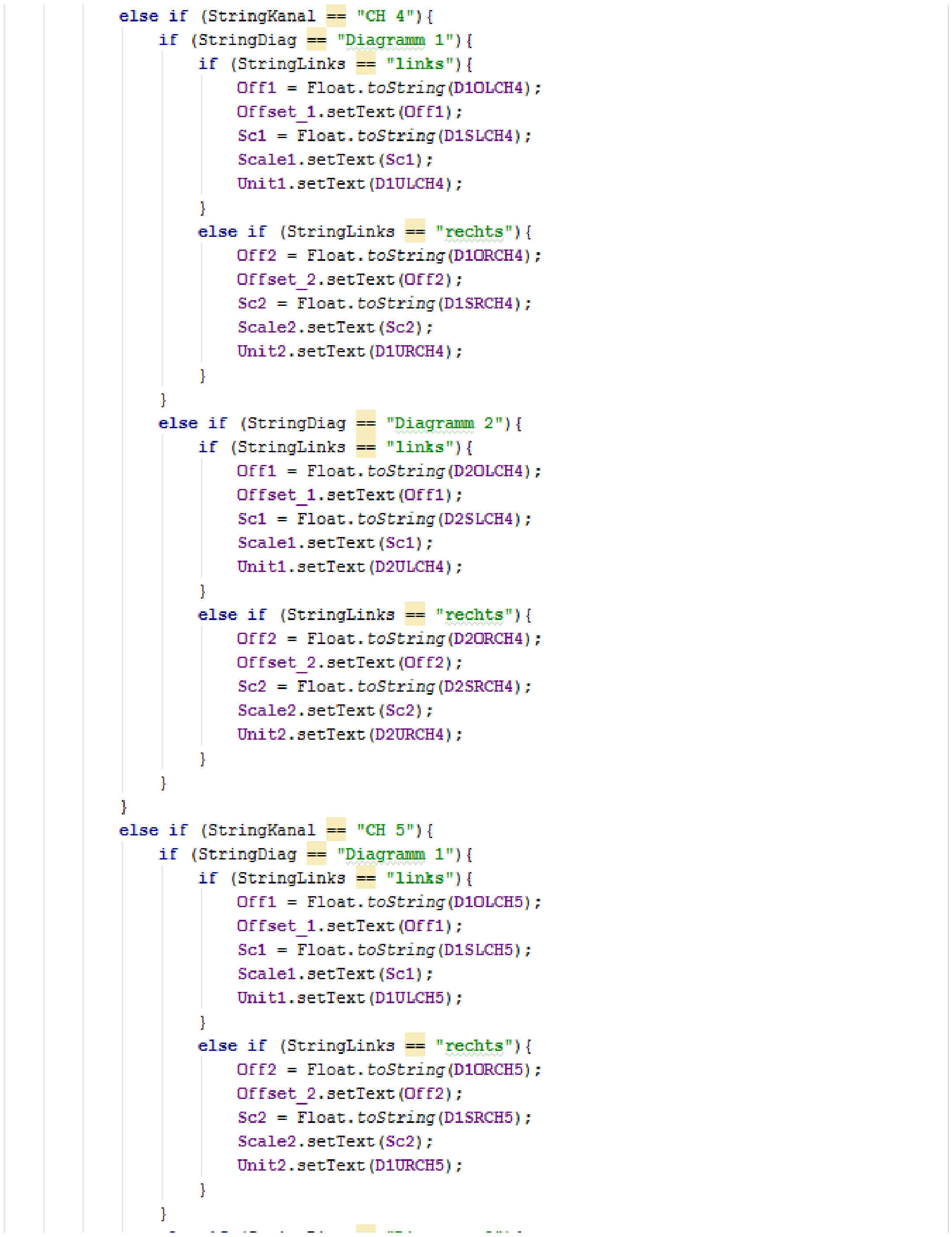
```
                else if (StringLinks == "rechts"){
                    Off2 = Float.toString(D2ORCH5);
                    Offset_2.setText(Off2);
                    Sc2 = Float.toString(D2SRCH5);
                    Scale2.setText(Sc2);
                    Unit2.setText(D2URCH5);
                }
            }
        }
        else if (StringKanal == "CH 6"){
            if (StringDiag == "Diagramm 1"){
                if (StringLinks == "links"){
                    Off1 = Float.toString(D1OLCH6);
                    Offset_1.setText(Off1);
                    Sc1 = Float.toString(D1SLCH6);
                    Scale1.setText(Sc1);
                    Unit1.setText(D1ULCH6);
                }
                else if (StringLinks == "rechts"){
                    Off2 = Float.toString(D1ORCH6);
                    Offset_2.setText(Off2);
                    Sc2 = Float.toString(D1SRCH6);
                    Scale2.setText(Sc2);
                    Unit2.setText(D1URCH6);
                }
            }
            else if (StringDiag == "Diagramm 2"){
                if (StringLinks == "links"){
                    Off1 = Float.toString(D2OLCH6);
                    Offset_1.setText(Off1);
                    Sc1 = Float.toString(D2SLCH6);
                    Scale1.setText(Sc1);
                    Unit1.setText(D2ULCH6);
                }
                else if (StringLinks == "rechts"){
                    Off2 = Float.toString(D2ORCH6);
                    Offset_2.setText(Off2);
                    Sc2 = Float.toString(D2SRCH6);
                    Scale2.setText(Sc2);
                    Unit2.setText(D2URCH6);
                }
            }
        }
    }

    public void onNothingSelected(AdapterView<?> parentView)
    {

        //return;
    }
});
```



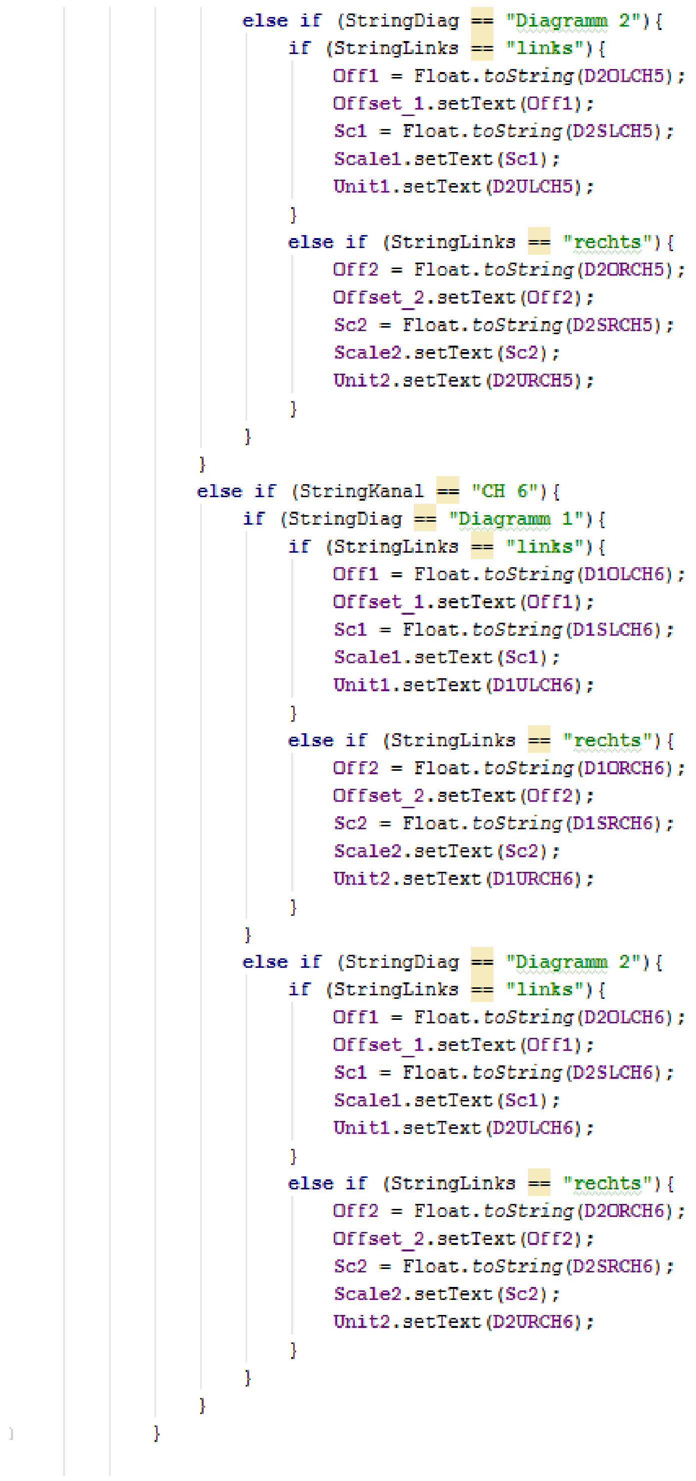
```
SPLinks.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

    public void onItemSelected(AdapterView<?> parentView, View selectedItemView, int position, long id)
    {
        String StringDiag = SPDiagramm.getSelectedItem().toString();
        String StringLinks = SPLinks.getSelectedItem().toString();
        String StringKanal = SPKanal.getSelectedItem().toString();

        if (StringKanal == "CH 1"){
            if (StringDiag == "Diagramm 1"){
                if (StringLinks == "links"){
                    Off1 = Float.toString(D1OLCH1);
                    Offset_1.setText(Off1);
                    Sc1 = Float.toString(D1SLCH1);
                    Scale1.setText(Sc1);
                    Unit1.setText(D1ULCH1);
                }
                else if (StringLinks == "rechts"){
                    Off2 = Float.toString(D1ORCH1);
                    Offset_2.setText(Off2);
                    Sc2 = Float.toString(D1SRCH1);
                    Scale2.setText(Sc2);
                    Unit2.setText(D1URCH1);
                }
            }
            else if (StringDiag == "Diagramm 2"){
                if (StringLinks == "links"){
                    Off1 = Float.toString(D2OLCH1);
                    Offset_1.setText(Off1);
                    Sc1 = Float.toString(D2SLCH1);
                    Scale1.setText(Sc1);
                    Unit1.setText(D2ULCH1);
                }
                else if (StringLinks == "rechts"){
                    Off2 = Float.toString(D2ORCH1);
                    Offset_2.setText(Off2);
                    Sc2 = Float.toString(D2SRCH1);
                    Scale2.setText(Sc2);
                    Unit2.setText(D2URCH1);
                }
            }
        }
        else if (StringKanal == "CH 2"){
            if (StringDiag == "Diagramm 1"){
                if (StringLinks == "links"){
                    Off1 = Float.toString(D1OLCH2);
                    Offset_1.setText(Off1);
                    Sc1 = Float.toString(D1SLCH2);
                    Scale1.setText(Sc1);
                    Unit1.setText(D1ULCH2);
                }
                else if (StringLinks == "rechts"){
                    Off2 = Float.toString(D1ORCH2);
                    Offset_2.setText(Off2);
                    Sc2 = Float.toString(D1SRCH2);
```



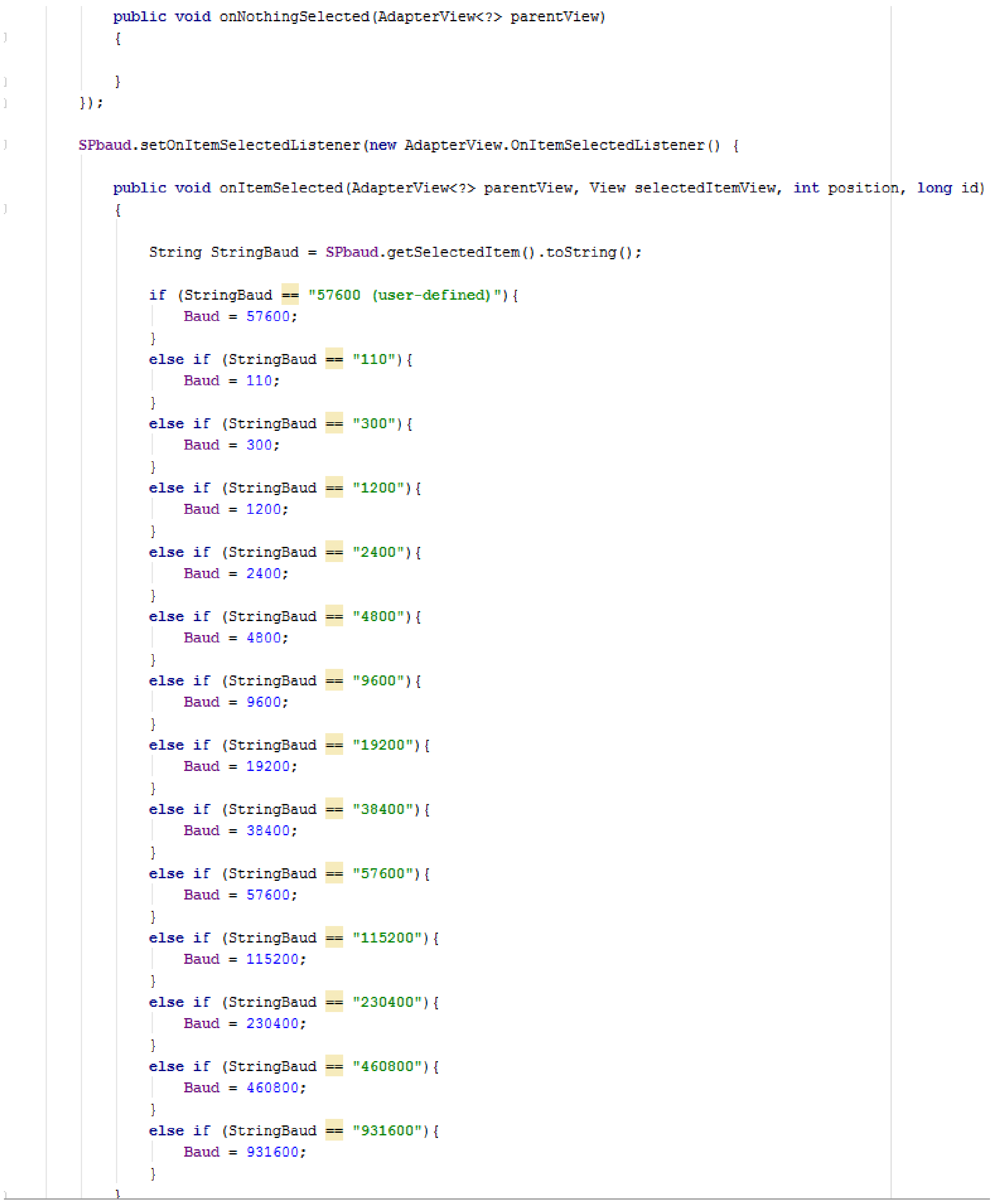
```
        }
    }
    else if (StringDiag == "Diagramm 2"){
        if (StringLinks == "links"){
            Off1 = Float.toString(D2OLCH2);
            Offset_1.setText(Off1);
            Sc1 = Float.toString(D2SLCH2);
            Scale1.setText(Sc1);
            Unit1.setText(D2ULCH2);
        }
        else if (StringLinks == "rechts"){
            Off2 = Float.toString(D2ORCH2);
            Offset_2.setText(Off2);
            Sc2 = Float.toString(D2SRCH2);
            Scale2.setText(Sc2);
            Unit2.setText(D2URCH2);
        }
    }
}
else if (StringKanal == "CH 3"){
    if (StringDiag == "Diagramm 1"){
        if (StringLinks == "links"){
            Off1 = Float.toString(D1OLCH3);
            Offset_1.setText(Off1);
            Sc1 = Float.toString(D1SLCH3);
            Scale1.setText(Sc1);
            Unit1.setText(D1ULCH3);
        }
        else if (StringLinks == "rechts"){
            Off2 = Float.toString(D1ORCH3);
            Offset_2.setText(Off2);
            Sc2 = Float.toString(D1SRCH3);
            Scale2.setText(Sc2);
            Unit2.setText(D1URCH3);
        }
    }
    else if (StringDiag == "Diagramm 2"){
        if (StringLinks == "links"){
            Off1 = Float.toString(D2OLCH3);
            Offset_1.setText(Off1);
            Sc1 = Float.toString(D2SLCH3);
            Scale1.setText(Sc1);
            Unit1.setText(D2ULCH3);
        }
        else if (StringLinks == "rechts"){
            Off2 = Float.toString(D2ORCH3);
            Offset_2.setText(Off2);
            Sc2 = Float.toString(D2SRCH3);
            Scale2.setText(Sc2);
            Unit2.setText(D2URCH3);
        }
    }
}
```



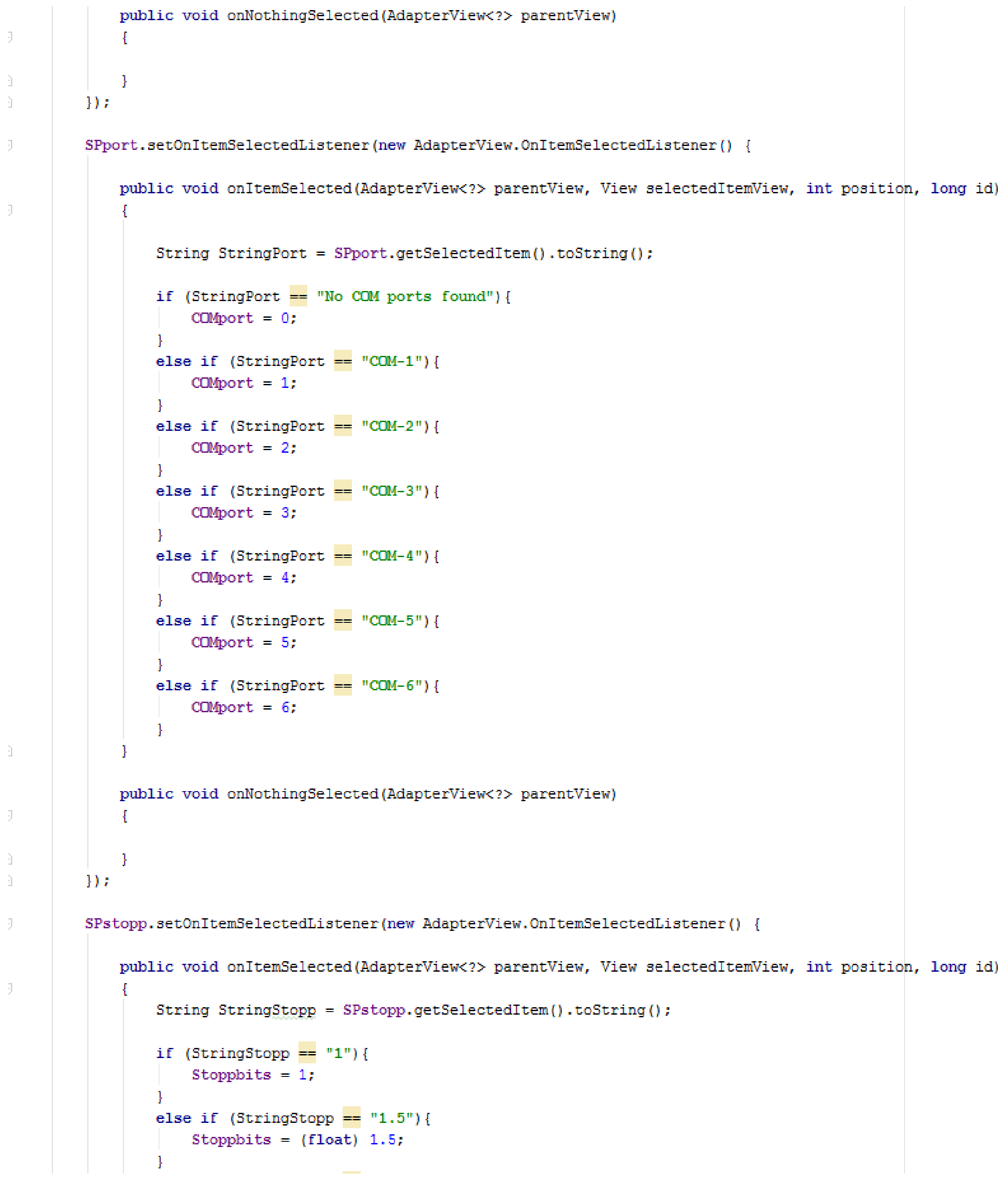
```
else if (StringKanal == "CH 4"){
    if (StringDiag == "Diagramm 1"){
        if (StringLinks == "links"){
            Off1 = Float.toString(D1OLCH4);
            Offset_1.setText(Off1);
            Sc1 = Float.toString(D1SLCH4);
            Scale1.setText(Sc1);
            Unit1.setText(D1ULCH4);
        }
        else if (StringLinks == "rechts"){
            Off2 = Float.toString(D1ORCH4);
            Offset_2.setText(Off2);
            Sc2 = Float.toString(D1SRCH4);
            Scale2.setText(Sc2);
            Unit2.setText(D1URCH4);
        }
    }
    else if (StringDiag == "Diagramm 2"){
        if (StringLinks == "links"){
            Off1 = Float.toString(D2OLCH4);
            Offset_1.setText(Off1);
            Sc1 = Float.toString(D2SLCH4);
            Scale1.setText(Sc1);
            Unit1.setText(D2ULCH4);
        }
        else if (StringLinks == "rechts"){
            Off2 = Float.toString(D2ORCH4);
            Offset_2.setText(Off2);
            Sc2 = Float.toString(D2SRCH4);
            Scale2.setText(Sc2);
            Unit2.setText(D2URCH4);
        }
    }
}
else if (StringKanal == "CH 5"){
    if (StringDiag == "Diagramm 1"){
        if (StringLinks == "links"){
            Off1 = Float.toString(D1OLCH5);
            Offset_1.setText(Off1);
            Sc1 = Float.toString(D1SLCH5);
            Scale1.setText(Sc1);
            Unit1.setText(D1ULCH5);
        }
        else if (StringLinks == "rechts"){
            Off2 = Float.toString(D1ORCH5);
            Offset_2.setText(Off2);
            Sc2 = Float.toString(D1SRCH5);
            Scale2.setText(Sc2);
            Unit2.setText(D1URCH5);
        }
    }
```



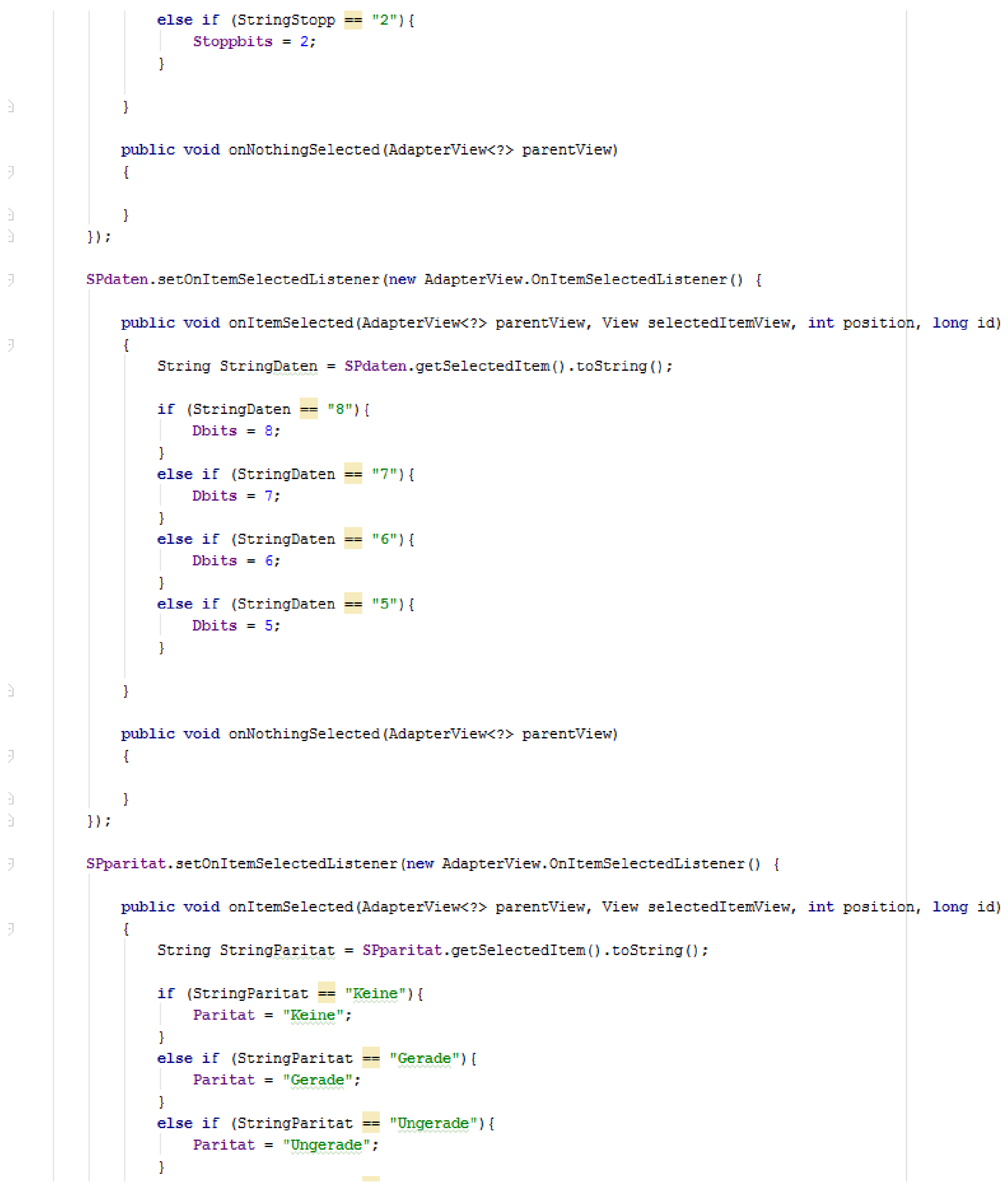
```
                    else if (StringDiag == "Diagramm 2"){
                        if (StringLinks == "links"){
                            Off1 = Float.toString(D2OLCH5);
                            Offset_1.setText(Off1);
                            Sc1 = Float.toString(D2SLCH5);
                            Scale1.setText(Sc1);
                            Unit1.setText(D2ULCH5);
                        }
                        else if (StringLinks == "rechts"){
                            Off2 = Float.toString(D2ORCH5);
                            Offset_2.setText(Off2);
                            Sc2 = Float.toString(D2SRCH5);
                            Scale2.setText(Sc2);
                            Unit2.setText(D2URCH5);
                        }
                    }
                }
                else if (StringKanal == "CH 6"){
                    if (StringDiag == "Diagramm 1"){
                        if (StringLinks == "links"){
                            Off1 = Float.toString(D1OLCH6);
                            Offset_1.setText(Off1);
                            Sc1 = Float.toString(D1SLCH6);
                            Scale1.setText(Sc1);
                            Unit1.setText(D1ULCH6);
                        }
                        else if (StringLinks == "rechts"){
                            Off2 = Float.toString(D1ORCH6);
                            Offset_2.setText(Off2);
                            Sc2 = Float.toString(D1SRCH6);
                            Scale2.setText(Sc2);
                            Unit2.setText(D1URCH6);
                        }
                    }
                    else if (StringDiag == "Diagramm 2"){
                        if (StringLinks == "links"){
                            Off1 = Float.toString(D2OLCH6);
                            Offset_1.setText(Off1);
                            Sc1 = Float.toString(D2SLCH6);
                            Scale1.setText(Sc1);
                            Unit1.setText(D2ULCH6);
                        }
                        else if (StringLinks == "rechts"){
                            Off2 = Float.toString(D2ORCH6);
                            Offset_2.setText(Off2);
                            Sc2 = Float.toString(D2SRCH6);
                            Scale2.setText(Sc2);
                            Unit2.setText(D2URCH6);
                        }
                    }
                }
}           }
```



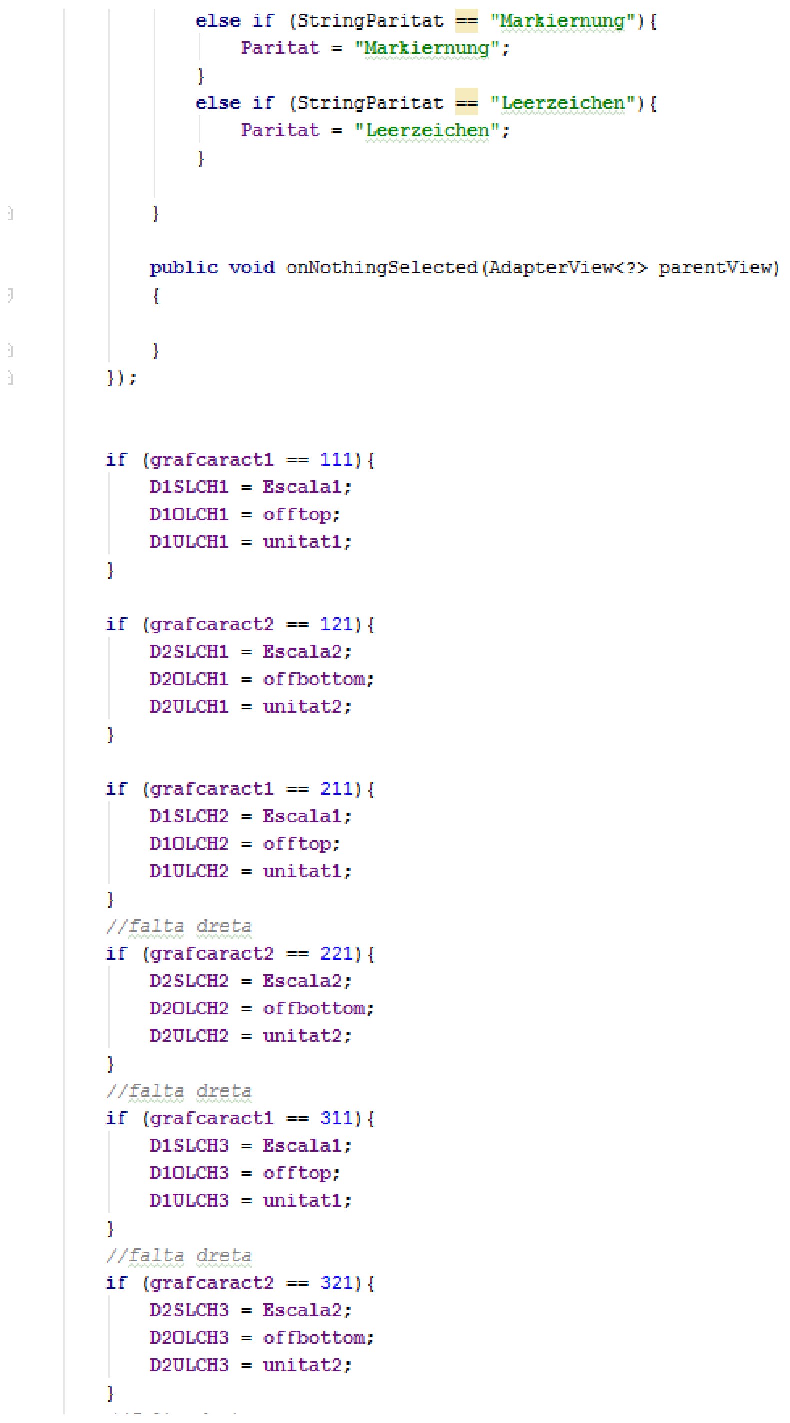
```
        public void onNothingSelected(AdapterView<?> parentView)
        {

        }
    });

    SPbaud.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

        public void onItemSelected(AdapterView<?> parentView, View selectedItemView, int position, long id)
        {

            String StringBaud = SPbaud.getSelectedItem().toString();

            if (StringBaud == "57600 (user-defined)"){
                Baud = 57600;
            }
            else if (StringBaud == "110"){
                Baud = 110;
            }
            else if (StringBaud == "300"){
                Baud = 300;
            }
            else if (StringBaud == "1200"){
                Baud = 1200;
            }
            else if (StringBaud == "2400"){
                Baud = 2400;
            }
            else if (StringBaud == "4800"){
                Baud = 4800;
            }
            else if (StringBaud == "9600"){
                Baud = 9600;
            }
            else if (StringBaud == "19200"){
                Baud = 19200;
            }
            else if (StringBaud == "38400"){
                Baud = 38400;
            }
            else if (StringBaud == "57600"){
                Baud = 57600;
            }
            else if (StringBaud == "115200"){
                Baud = 115200;
            }
            else if (StringBaud == "230400"){
                Baud = 230400;
            }
            else if (StringBaud == "460800"){
                Baud = 460800;
            }
            else if (StringBaud == "931600"){
                Baud = 931600;
            }
        }
```



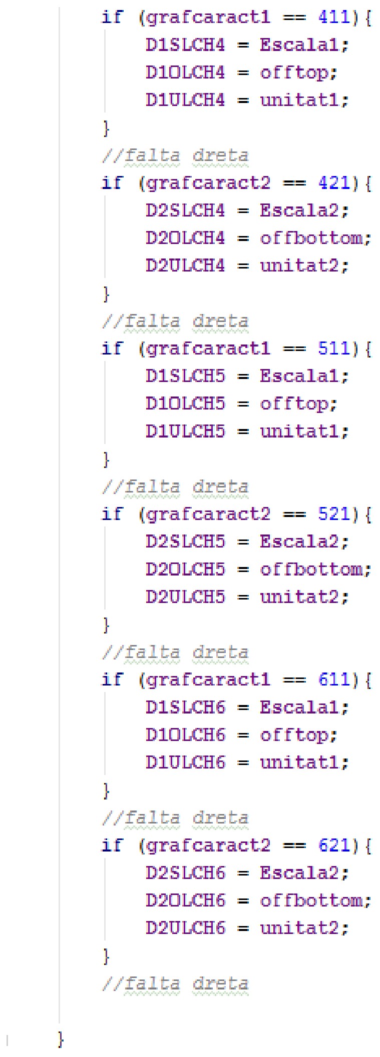
```
        public void onNothingSelected(AdapterView<?> parentView)
        {

        }
    });

    SPport.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

        public void onItemSelected(AdapterView<?> parentView, View selectedItemView, int position, long id)
        {

            String StringPort = SPport.getSelectedItem().toString();

            if (StringPort == "No COM ports found"){
                COMport = 0;
            }
            else if (StringPort == "COM-1"){
                COMport = 1;
            }
            else if (StringPort == "COM-2"){
                COMport = 2;
            }
            else if (StringPort == "COM-3"){
                COMport = 3;
            }
            else if (StringPort == "COM-4"){
                COMport = 4;
            }
            else if (StringPort == "COM-5"){
                COMport = 5;
            }
            else if (StringPort == "COM-6"){
                COMport = 6;
            }
        }

        public void onNothingSelected(AdapterView<?> parentView)
        {

        }
    });

    SPstopp.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

        public void onItemSelected(AdapterView<?> parentView, View selectedItemView, int position, long id)
        {
            String StringStopp = SPstopp.getSelectedItem().toString();

            if (StringStopp == "1"){
                Stoppbits = 1;
            }
            else if (StringStopp == "1.5"){
                Stoppbits = (float) 1.5;
            }
```



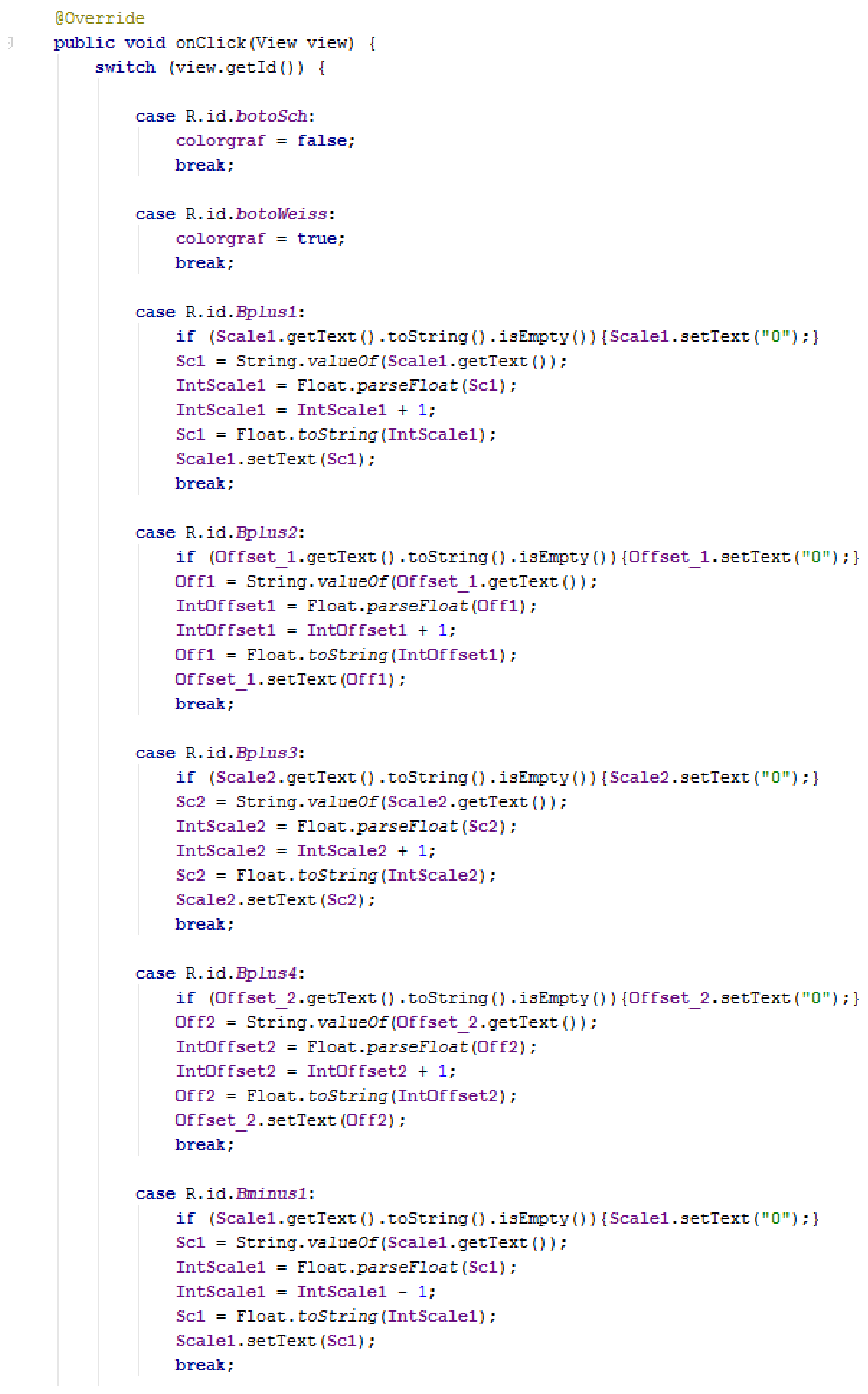
```
        else if (StringStopp == "2"){
            Stoppbits = 2;
        }

    }

    public void onNothingSelected(AdapterView<?> parentView)
    {

    }
});

SPdaten.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

    public void onItemSelected(AdapterView<?> parentView, View selectedItemView, int position, long id)
    {
        String StringDaten = SPdaten.getSelectedItem().toString();

        if (StringDaten == "8"){
            Dbits = 8;
        }
        else if (StringDaten == "7"){
            Dbits = 7;
        }
        else if (StringDaten == "6"){
            Dbits = 6;
        }
        else if (StringDaten == "5"){
            Dbits = 5;
        }

    }

    public void onNothingSelected(AdapterView<?> parentView)
    {

    }
});

SPparitat.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

    public void onItemSelected(AdapterView<?> parentView, View selectedItemView, int position, long id)
    {
        String StringParitat = SPparitat.getSelectedItem().toString();

        if (StringParitat == "Keine"){
            Paritat = "Keine";
        }
        else if (StringParitat == "Gerade"){
            Paritat = "Gerade";
        }
        else if (StringParitat == "Ungerade"){
            Paritat = "Ungerade";
        }
```



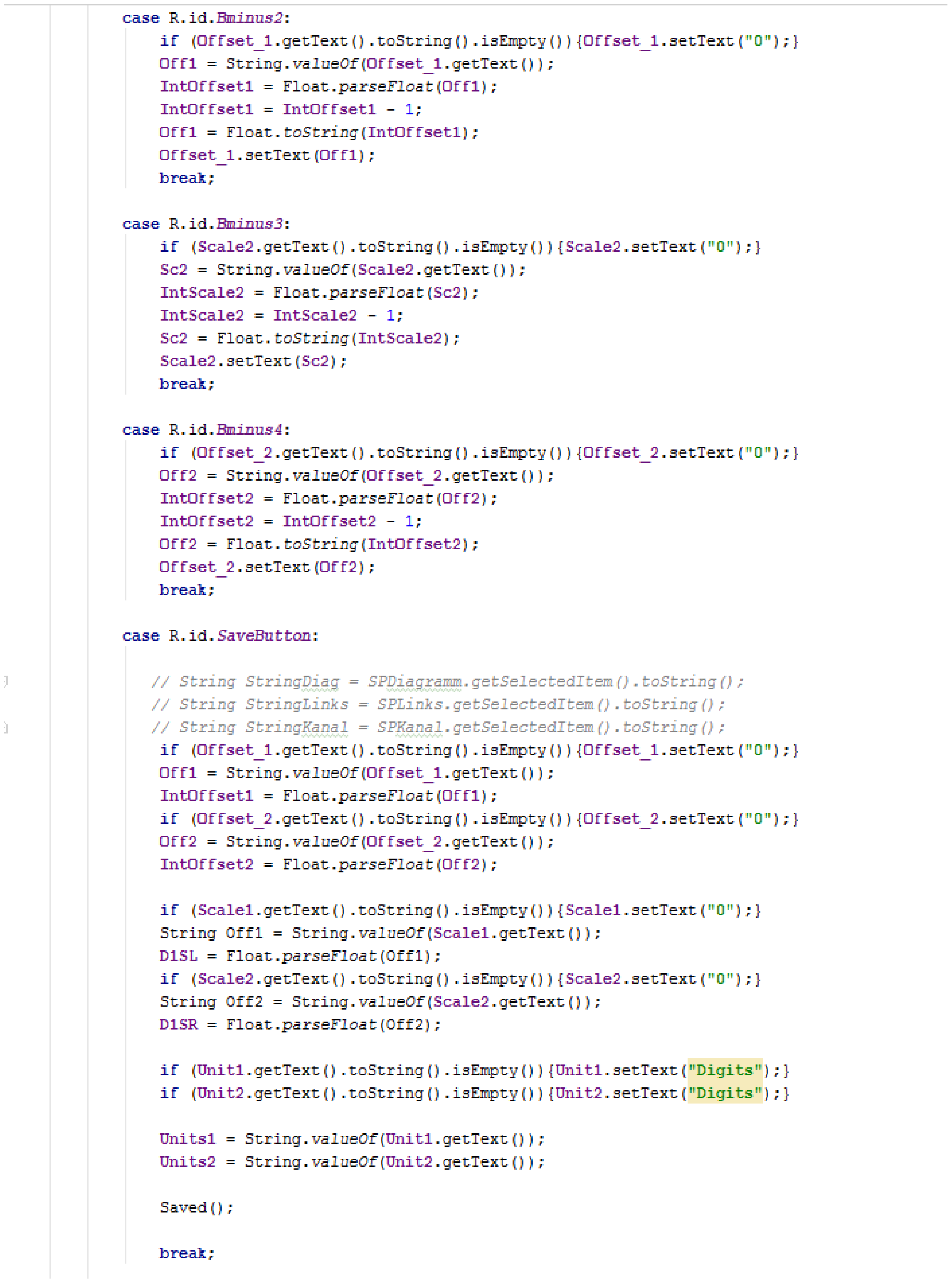
```
        else if (StringParitat == "Markiernung"){
            Paritat = "Markiernung";
        }
        else if (StringParitat == "Leerzeichen"){
            Paritat = "Leerzeichen";
        }

    }

    public void onNothingSelected(AdapterView<?> parentView)
    {

    }
});


if (grafcaract1 == 111){
    D1SLCH1 = Escala1;
    D1OLCH1 = offtop;
    D1ULCH1 = unitat1;
}

if (grafcaract2 == 121){
    D2SLCH1 = Escala2;
    D2OLCH1 = offbottom;
    D2ULCH1 = unitat2;
}

if (grafcaract1 == 211){
    D1SLCH2 = Escala1;
    D1OLCH2 = offtop;
    D1ULCH2 = unitat1;
}
//falta dreta
if (grafcaract2 == 221){
    D2SLCH2 = Escala2;
    D2OLCH2 = offbottom;
    D2ULCH2 = unitat2;
}
//falta dreta
if (grafcaract1 == 311){
    D1SLCH3 = Escala1;
    D1OLCH3 = offtop;
    D1ULCH3 = unitat1;
}
//falta dreta
if (grafcaract2 == 321){
    D2SLCH3 = Escala2;
    D2OLCH3 = offbottom;
    D2ULCH3 = unitat2;
}
```



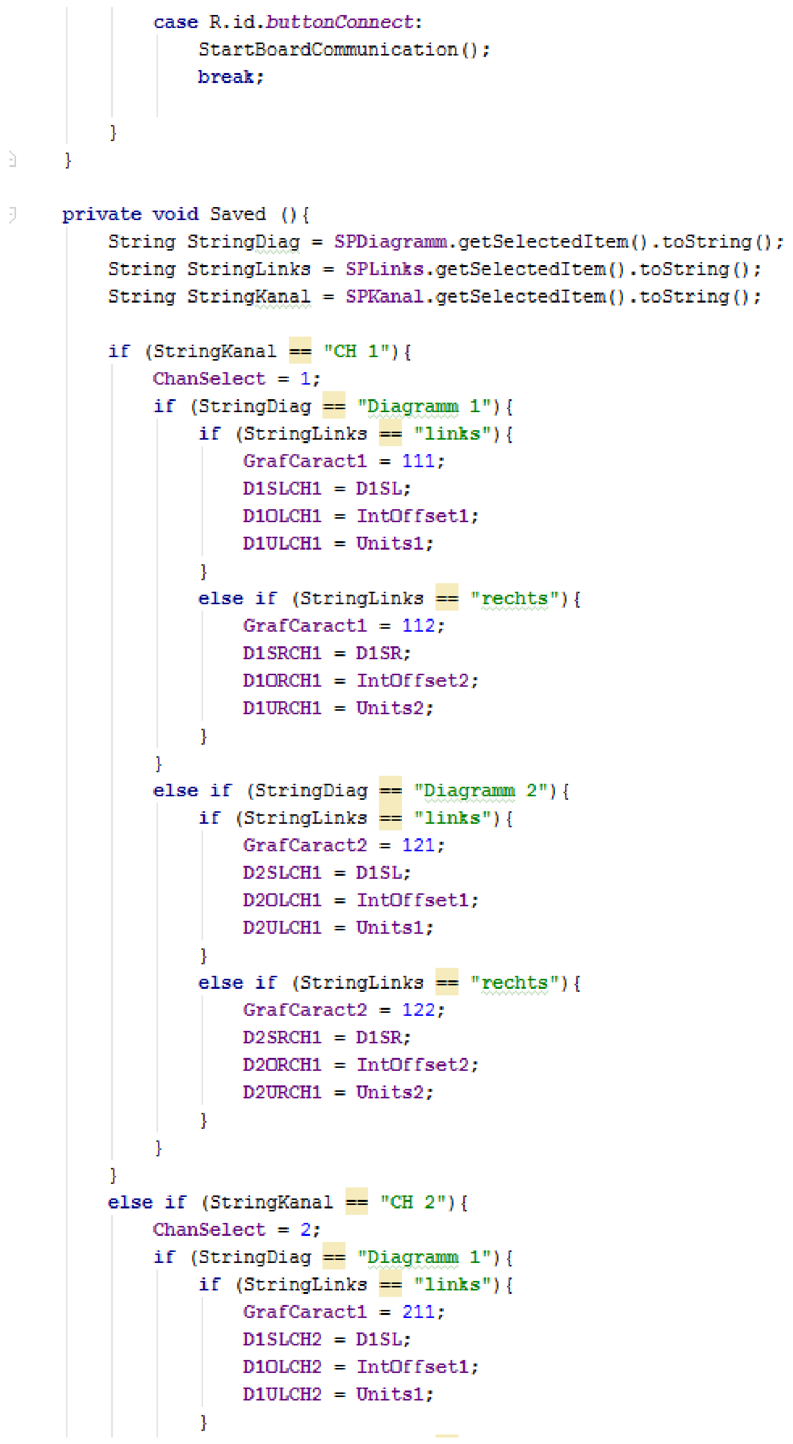
```
    if (grafcaract1 == 411){
        D1SLCH4 = Escala1;
        D1OLCH4 = offtop;
        D1ULCH4 = unitat1;
    }
    //falta dreta
    if (grafcaract2 == 421){
        D2SLCH4 = Escala2;
        D2OLCH4 = offbottom;
        D2ULCH4 = unitat2;
    }
    //falta dreta
    if (grafcaract1 == 511){
        D1SLCH5 = Escala1;
        D1OLCH5 = offtop;
        D1ULCH5 = unitat1;
    }
    //falta dreta
    if (grafcaract2 == 521){
        D2SLCH5 = Escala2;
        D2OLCH5 = offbottom;
        D2ULCH5 = unitat2;
    }
    //falta dreta
    if (grafcaract1 == 611){
        D1SLCH6 = Escala1;
        D1OLCH6 = offtop;
        D1ULCH6 = unitat1;
    }
    //falta dreta
    if (grafcaract2 == 621){
        D2SLCH6 = Escala2;
        D2OLCH6 = offbottom;
        D2ULCH6 = unitat2;
    }
    //falta dreta

}
```



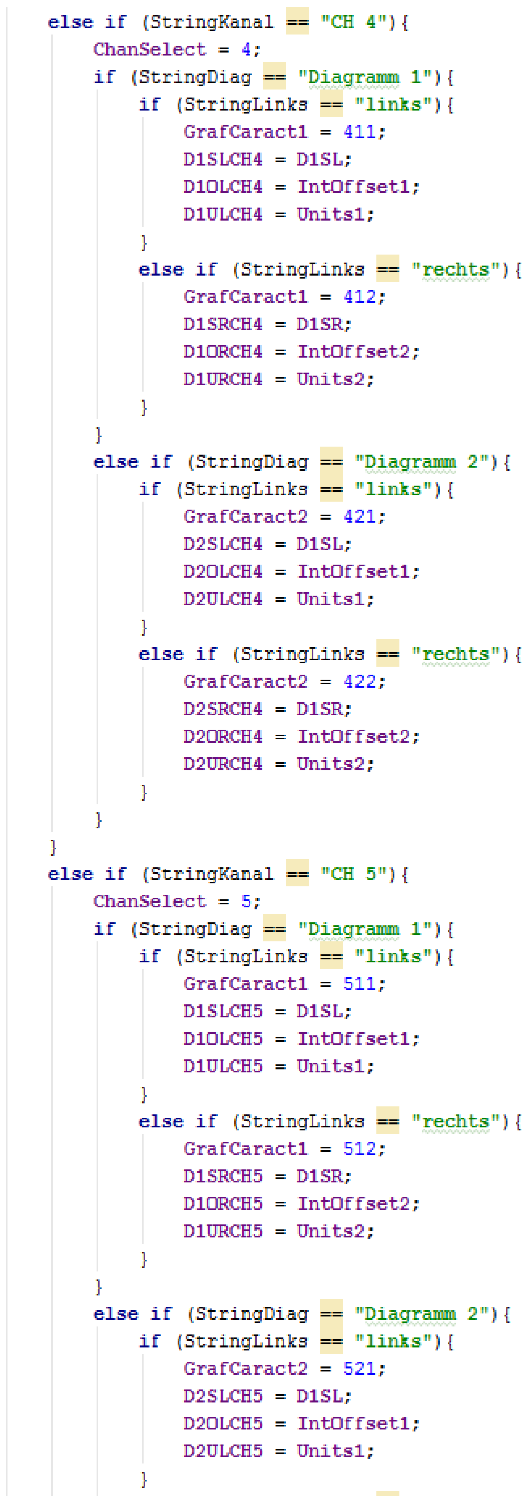
```java
@Override
public void onClick(View view) {
    switch (view.getId()) {

        case R.id.botoSch:
            colorgraf = false;
            break;

        case R.id.botoWeiss:
            colorgraf = true;
            break;

        case R.id.Bplus1:
            if (Scale1.getText().toString().isEmpty()){Scale1.setText("0");}
            Sc1 = String.valueOf(Scale1.getText());
            IntScale1 = Float.parseFloat(Sc1);
            IntScale1 = IntScale1 + 1;
            Sc1 = Float.toString(IntScale1);
            Scale1.setText(Sc1);
            break;

        case R.id.Bplus2:
            if (Offset_1.getText().toString().isEmpty()){Offset_1.setText("0");}
            Off1 = String.valueOf(Offset_1.getText());
            IntOffset1 = Float.parseFloat(Off1);
            IntOffset1 = IntOffset1 + 1;
            Off1 = Float.toString(IntOffset1);
            Offset_1.setText(Off1);
            break;

        case R.id.Bplus3:
            if (Scale2.getText().toString().isEmpty()){Scale2.setText("0");}
            Sc2 = String.valueOf(Scale2.getText());
            IntScale2 = Float.parseFloat(Sc2);
            IntScale2 = IntScale2 + 1;
            Sc2 = Float.toString(IntScale2);
            Scale2.setText(Sc2);
            break;

        case R.id.Bplus4:
            if (Offset_2.getText().toString().isEmpty()){Offset_2.setText("0");}
            Off2 = String.valueOf(Offset_2.getText());
            IntOffset2 = Float.parseFloat(Off2);
            IntOffset2 = IntOffset2 + 1;
            Off2 = Float.toString(IntOffset2);
            Offset_2.setText(Off2);
            break;

        case R.id.Bminus1:
            if (Scale1.getText().toString().isEmpty()){Scale1.setText("0");}
            Sc1 = String.valueOf(Scale1.getText());
            IntScale1 = Float.parseFloat(Sc1);
            IntScale1 = IntScale1 - 1;
            Sc1 = Float.toString(IntScale1);
            Scale1.setText(Sc1);
            break;
```



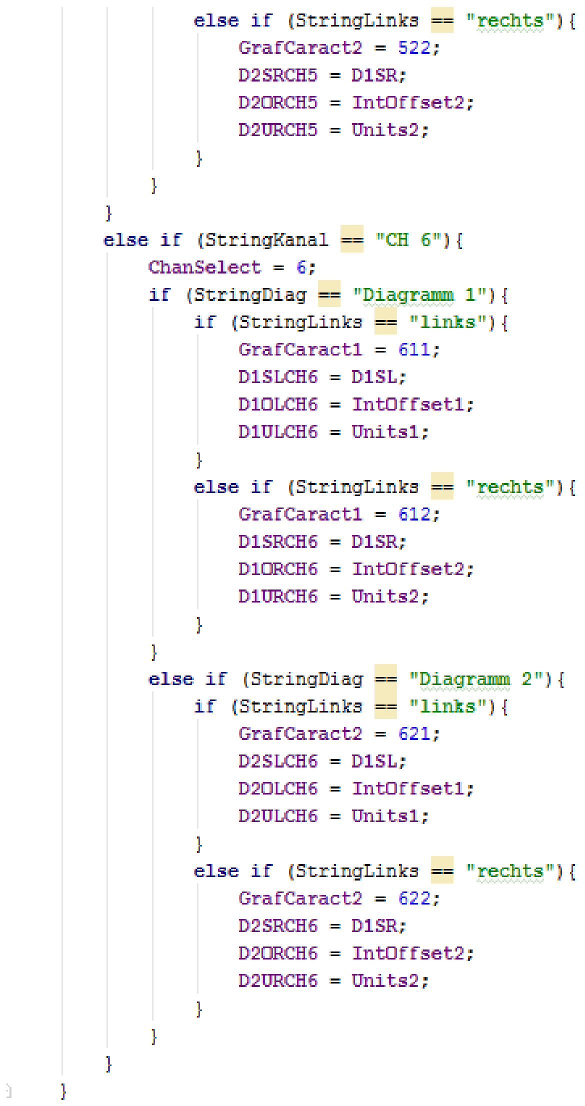
```
case R.id.Bminus2:
    if (Offset_1.getText().toString().isEmpty()){Offset_1.setText("0");}
    Off1 = String.valueOf(Offset_1.getText());
    IntOffset1 = Float.parseFloat(Off1);
    IntOffset1 = IntOffset1 - 1;
    Off1 = Float.toString(IntOffset1);
    Offset_1.setText(Off1);
    break;

case R.id.Bminus3:
    if (Scale2.getText().toString().isEmpty()){Scale2.setText("0");}
    Sc2 = String.valueOf(Scale2.getText());
    IntScale2 = Float.parseFloat(Sc2);
    IntScale2 = IntScale2 - 1;
    Sc2 = Float.toString(IntScale2);
    Scale2.setText(Sc2);
    break;

case R.id.Bminus4:
    if (Offset_2.getText().toString().isEmpty()){Offset_2.setText("0");}
    Off2 = String.valueOf(Offset_2.getText());
    IntOffset2 = Float.parseFloat(Off2);
    IntOffset2 = IntOffset2 - 1;
    Off2 = Float.toString(IntOffset2);
    Offset_2.setText(Off2);
    break;

case R.id.SaveButton:

   // String StringDiag = SPDiagramm.getSelectedItem().toString();
   // String StringLinks = SPLinks.getSelectedItem().toString();
   // String StringKanal = SPKanal.getSelectedItem().toString();
    if (Offset_1.getText().toString().isEmpty()){Offset_1.setText("0");}
    Off1 = String.valueOf(Offset_1.getText());
    IntOffset1 = Float.parseFloat(Off1);
    if (Offset_2.getText().toString().isEmpty()){Offset_2.setText("0");}
    Off2 = String.valueOf(Offset_2.getText());
    IntOffset2 = Float.parseFloat(Off2);

    if (Scale1.getText().toString().isEmpty()){Scale1.setText("0");}
    String Off1 = String.valueOf(Scale1.getText());
    D1SL = Float.parseFloat(Off1);
    if (Scale2.getText().toString().isEmpty()){Scale2.setText("0");}
    String Off2 = String.valueOf(Scale2.getText());
    D1SR = Float.parseFloat(Off2);

    if (Unit1.getText().toString().isEmpty()){Unit1.setText("Digits");}
    if (Unit2.getText().toString().isEmpty()){Unit2.setText("Digits");}

    Units1 = String.valueOf(Unit1.getText());
    Units2 = String.valueOf(Unit2.getText());

    Saved();

    break;
```



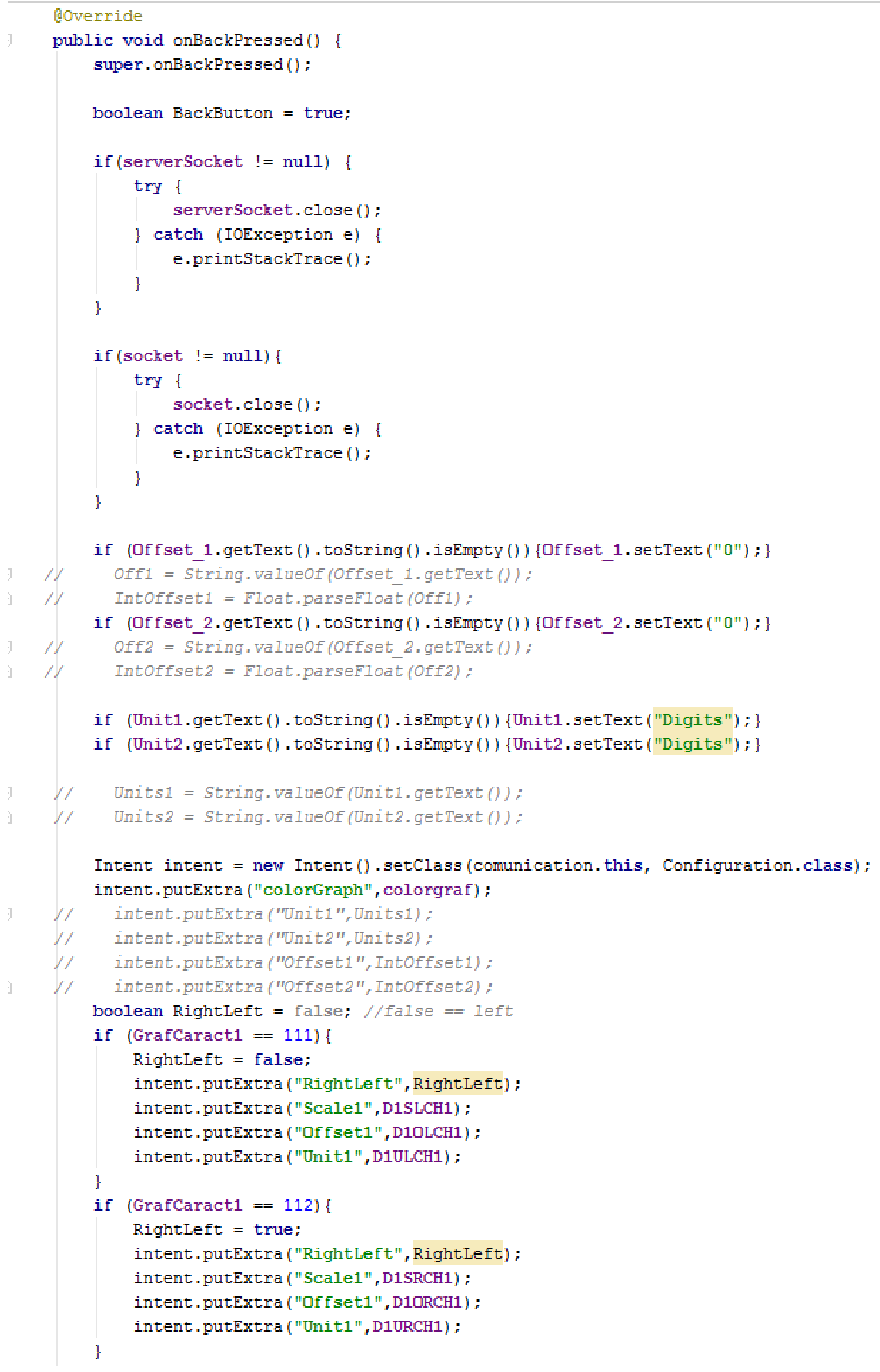
```
            case R.id.buttonConnect:
                StartBoardCommunication();
                break;

        }
    }

    private void Saved (){
        String StringDiag = SPDiagramm.getSelectedItem().toString();
        String StringLinks = SPLinks.getSelectedItem().toString();
        String StringKanal = SPKanal.getSelectedItem().toString();

        if (StringKanal == "CH 1"){
            ChanSelect = 1;
            if (StringDiag == "Diagramm 1"){
                if (StringLinks == "links"){
                    GrafCaract1 = 111;
                    D1SLCH1 = D1SL;
                    D1OLCH1 = IntOffset1;
                    D1ULCH1 = Units1;
                }
                else if (StringLinks == "rechts"){
                    GrafCaract1 = 112;
                    D1SRCH1 = D1SR;
                    D1ORCH1 = IntOffset2;
                    D1URCH1 = Units2;
                }
            }
            else if (StringDiag == "Diagramm 2"){
                if (StringLinks == "links"){
                    GrafCaract2 = 121;
                    D2SLCH1 = D1SL;
                    D2OLCH1 = IntOffset1;
                    D2ULCH1 = Units1;
                }
                else if (StringLinks == "rechts"){
                    GrafCaract2 = 122;
                    D2SRCH1 = D1SR;
                    D2ORCH1 = IntOffset2;
                    D2URCH1 = Units2;
                }
            }
        }
        else if (StringKanal == "CH 2"){
            ChanSelect = 2;
            if (StringDiag == "Diagramm 1"){
                if (StringLinks == "links"){
                    GrafCaract1 = 211;
                    D1SLCH2 = D1SL;
                    D1OLCH2 = IntOffset1;
                    D1ULCH2 = Units1;
                }
        else if (StringLinks == "rechts"){
            GrafCaract1 = 212;
            D1SRCH2 = D1SR;
            D1ORCH2 = IntOffset2;
            D1URCH2 = Units2;
        }
    }
    else if (StringDiag == "Diagramm 2"){
        if (StringLinks == "links"){
            GrafCaract2 = 221;
            D2SLCH2 = D1SL;
            D2OLCH2 = IntOffset1;
            D2ULCH2 = Units1;
        }
        else if (StringLinks == "rechts"){
            GrafCaract2 = 222;
            D2SRCH2 = D1SR;
            D2ORCH2 = IntOffset2;
            D2URCH2 = Units2;
        }
    }
}
else if (StringKanal == "CH 3"){
    ChanSelect = 3;
    if (StringDiag == "Diagramm 1"){
        if (StringLinks == "links"){
            GrafCaract1 = 311;
            D1SLCH3 = D1SL;
            D1OLCH3 = IntOffset1;
            D1ULCH3 = Units1;
        }
        else if (StringLinks == "rechts"){
            GrafCaract1 = 312;
            D1SRCH3 = D1SR;
            D1ORCH3 = IntOffset2;
            D1URCH3 = Units2;
        }
    }
    else if (StringDiag == "Diagramm 2"){
        if (StringLinks == "links"){
            GrafCaract2 = 321;
            D2SLCH3 = D1SL;
            D2OLCH3 = IntOffset1;
            D2ULCH3 = Units1;
        }
        else if (StringLinks == "rechts"){
            GrafCaract2 = 322;
            D2SRCH3 = D1SR;
            D2ORCH3 = IntOffset2;
            D2URCH3 = Units2;
        }
    }
}
```



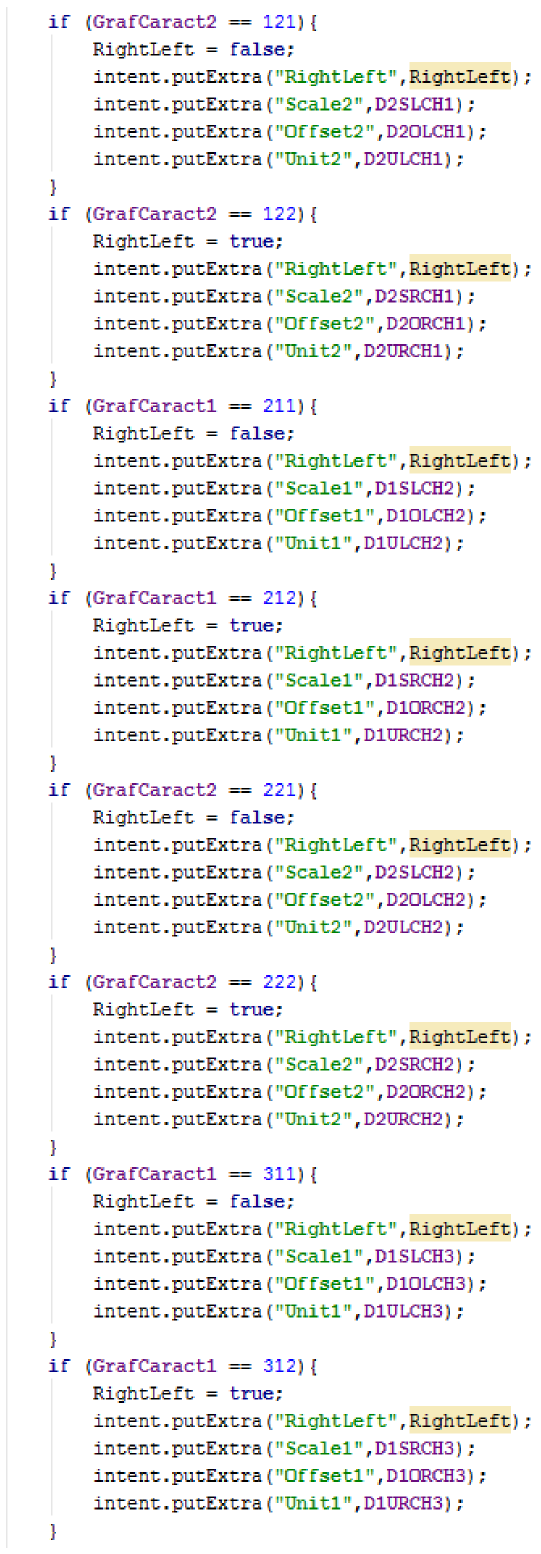
```
else if (StringKanal == "CH 4"){
    ChanSelect = 4;
    if (StringDiag == "Diagramm 1"){
        if (StringLinks == "links"){
            GrafCaract1 = 411;
            D1SLCH4 = D1SL;
            D1OLCH4 = IntOffset1;
            D1ULCH4 = Units1;
        }
        else if (StringLinks == "rechts"){
            GrafCaract1 = 412;
            D1SRCH4 = D1SR;
            D1ORCH4 = IntOffset2;
            D1URCH4 = Units2;
        }
    }
    else if (StringDiag == "Diagramm 2"){
        if (StringLinks == "links"){
            GrafCaract2 = 421;
            D2SLCH4 = D1SL;
            D2OLCH4 = IntOffset1;
            D2ULCH4 = Units1;
        }
        else if (StringLinks == "rechts"){
            GrafCaract2 = 422;
            D2SRCH4 = D1SR;
            D2ORCH4 = IntOffset2;
            D2URCH4 = Units2;
        }
    }
}
else if (StringKanal == "CH 5"){
    ChanSelect = 5;
    if (StringDiag == "Diagramm 1"){
        if (StringLinks == "links"){
            GrafCaract1 = 511;
            D1SLCH5 = D1SL;
            D1OLCH5 = IntOffset1;
            D1ULCH5 = Units1;
        }
        else if (StringLinks == "rechts"){
            GrafCaract1 = 512;
            D1SRCH5 = D1SR;
            D1ORCH5 = IntOffset2;
            D1URCH5 = Units2;
        }
    }
    else if (StringDiag == "Diagramm 2"){
        if (StringLinks == "links"){
            GrafCaract2 = 521;
            D2SLCH5 = D1SL;
            D2OLCH5 = IntOffset1;
            D2ULCH5 = Units1;
        }
```



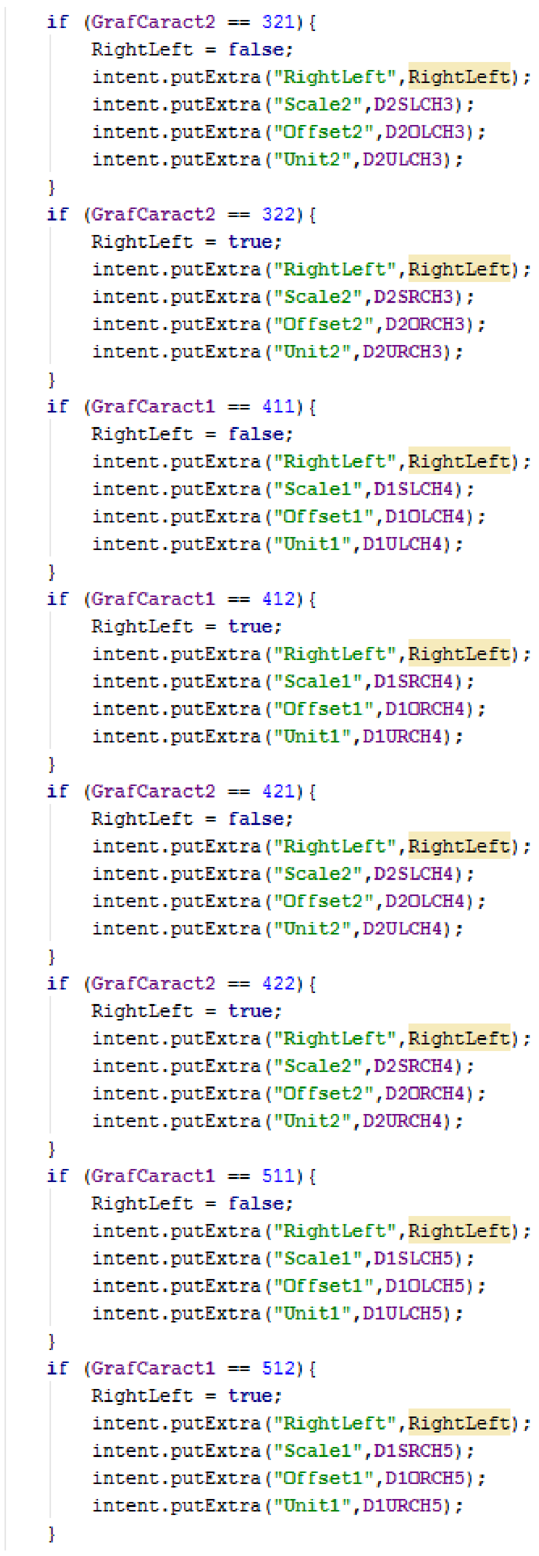
```
                else if (StringLinks == "rechts"){
                    GrafCaract2 = 522;
                    D2SRCH5 = D1SR;
                    D2ORCH5 = IntOffset2;
                    D2URCH5 = Units2;
                }
            }
        }
        else if (StringKanal == "CH 6"){
            ChanSelect = 6;
            if (StringDiag == "Diagramm 1"){
                if (StringLinks == "links"){
                    GrafCaract1 = 611;
                    D1SLCH6 = D1SL;
                    D1OLCH6 = IntOffset1;
                    D1ULCH6 = Units1;
                }
                else if (StringLinks == "rechts"){
                    GrafCaract1 = 612;
                    D1SRCH6 = D1SR;
                    D1ORCH6 = IntOffset2;
                    D1URCH6 = Units2;
                }
            }
            else if (StringDiag == "Diagramm 2"){
                if (StringLinks == "links"){
                    GrafCaract2 = 621;
                    D2SLCH6 = D1SL;
                    D2OLCH6 = IntOffset1;
                    D2ULCH6 = Units1;
                }
                else if (StringLinks == "rechts"){
                    GrafCaract2 = 622;
                    D2SRCH6 = D1SR;
                    D2ORCH6 = IntOffset2;
                    D2URCH6 = Units2;
                }
            }
        }
}   }
```



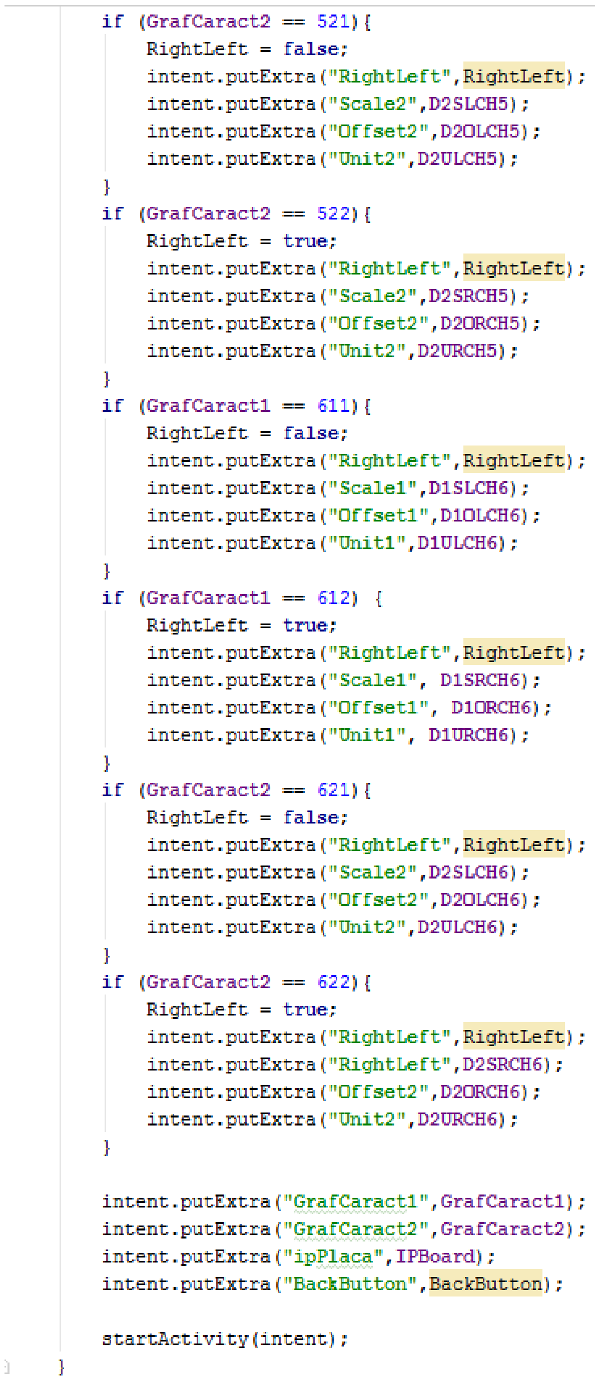
```java
@Override
public void onBackPressed() {
    super.onBackPressed();

    boolean BackButton = true;

    if(serverSocket != null) {
        try {
            serverSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    if(socket != null){
        try {
            socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    if (Offset_1.getText().toString().isEmpty()){Offset_1.setText("0");}
//      Off1 = String.valueOf(Offset_1.getText());
//      IntOffset1 = Float.parseFloat(Off1);
    if (Offset_2.getText().toString().isEmpty()){Offset_2.setText("0");}
//      Off2 = String.valueOf(Offset_2.getText());
//      IntOffset2 = Float.parseFloat(Off2);

    if (Unit1.getText().toString().isEmpty()){Unit1.setText("Digits");}
    if (Unit2.getText().toString().isEmpty()){Unit2.setText("Digits");}

//      Units1 = String.valueOf(Unit1.getText());
//      Units2 = String.valueOf(Unit2.getText());

    Intent intent = new Intent().setClass(comunication.this, Configuration.class);
    intent.putExtra("colorGraph",colorgraf);
//      intent.putExtra("Unit1",Units1);
//      intent.putExtra("Unit2",Units2);
//      intent.putExtra("Offset1",IntOffset1);
//      intent.putExtra("Offset2",IntOffset2);
    boolean RightLeft = false; //false == left
    if (GrafCaract1 == 111){
        RightLeft = false;
        intent.putExtra("RightLeft",RightLeft);
        intent.putExtra("Scale1",D1SLCH1);
        intent.putExtra("Offset1",D1OLCH1);
        intent.putExtra("Unit1",D1ULCH1);
    }
    if (GrafCaract1 == 112){
        RightLeft = true;
        intent.putExtra("RightLeft",RightLeft);
        intent.putExtra("Scale1",D1SRCH1);
        intent.putExtra("Offset1",D1ORCH1);
        intent.putExtra("Unit1",D1URCH1);
    }
```



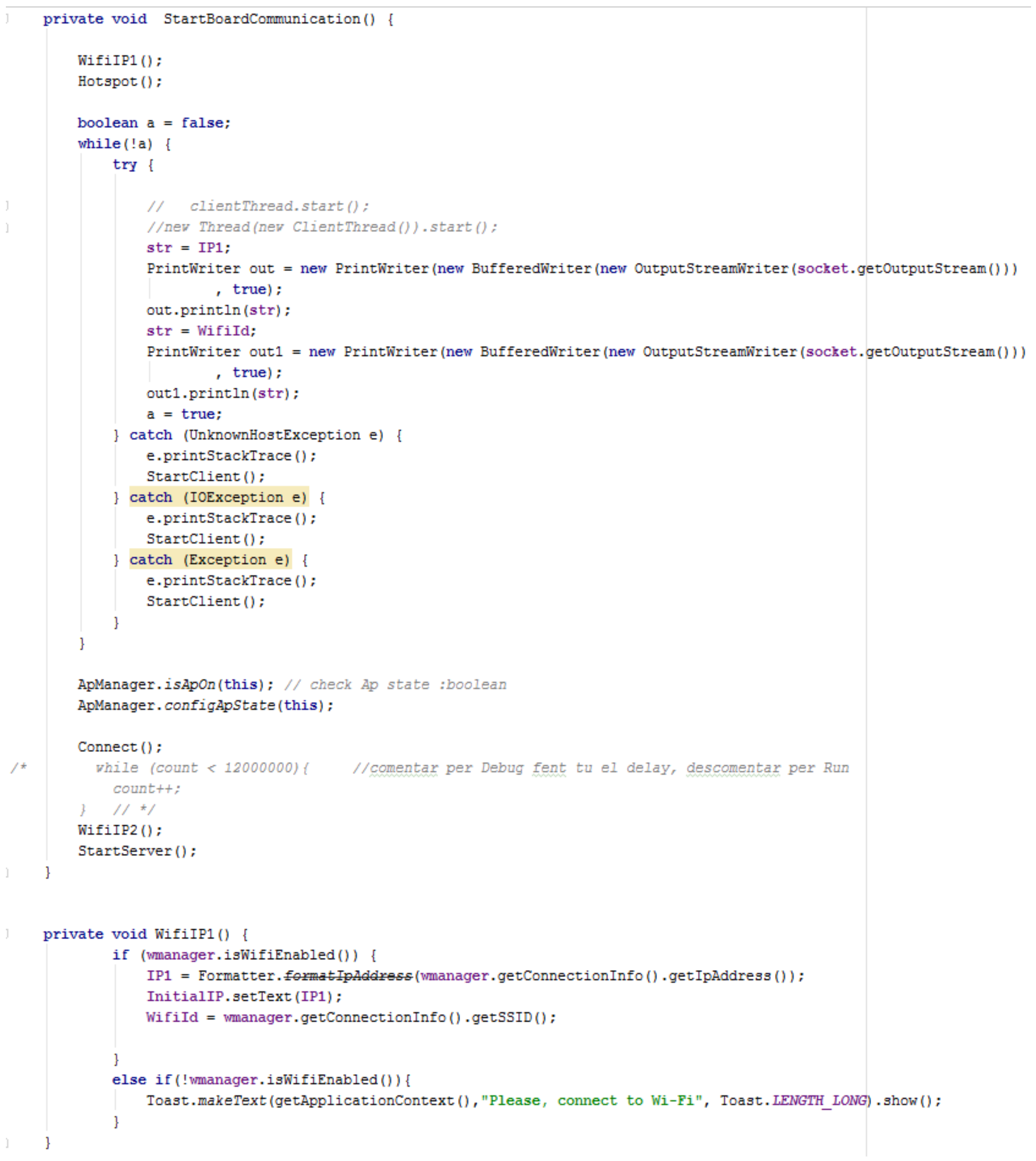
```
if (GrafCaract2 == 121){
    RightLeft = false;
    intent.putExtra("RightLeft",RightLeft);
    intent.putExtra("Scale2",D2SLCH1);
    intent.putExtra("Offset2",D2OLCH1);
    intent.putExtra("Unit2",D2ULCH1);
}
if (GrafCaract2 == 122){
    RightLeft = true;
    intent.putExtra("RightLeft",RightLeft);
    intent.putExtra("Scale2",D2SRCH1);
    intent.putExtra("Offset2",D2ORCH1);
    intent.putExtra("Unit2",D2URCH1);
}
if (GrafCaract1 == 211){
    RightLeft = false;
    intent.putExtra("RightLeft",RightLeft);
    intent.putExtra("Scale1",D1SLCH2);
    intent.putExtra("Offset1",D1OLCH2);
    intent.putExtra("Unit1",D1ULCH2);
}
if (GrafCaract1 == 212){
    RightLeft = true;
    intent.putExtra("RightLeft",RightLeft);
    intent.putExtra("Scale1",D1SRCH2);
    intent.putExtra("Offset1",D1ORCH2);
    intent.putExtra("Unit1",D1URCH2);
}
if (GrafCaract2 == 221){
    RightLeft = false;
    intent.putExtra("RightLeft",RightLeft);
    intent.putExtra("Scale2",D2SLCH2);
    intent.putExtra("Offset2",D2OLCH2);
    intent.putExtra("Unit2",D2ULCH2);
}
if (GrafCaract2 == 222){
    RightLeft = true;
    intent.putExtra("RightLeft",RightLeft);
    intent.putExtra("Scale2",D2SRCH2);
    intent.putExtra("Offset2",D2ORCH2);
    intent.putExtra("Unit2",D2URCH2);
}
if (GrafCaract1 == 311){
    RightLeft = false;
    intent.putExtra("RightLeft",RightLeft);
    intent.putExtra("Scale1",D1SLCH3);
    intent.putExtra("Offset1",D1OLCH3);
    intent.putExtra("Unit1",D1ULCH3);
}
if (GrafCaract1 == 312){
    RightLeft = true;
    intent.putExtra("RightLeft",RightLeft);
    intent.putExtra("Scale1",D1SRCH3);
    intent.putExtra("Offset1",D1ORCH3);
    intent.putExtra("Unit1",D1URCH3);
}
```



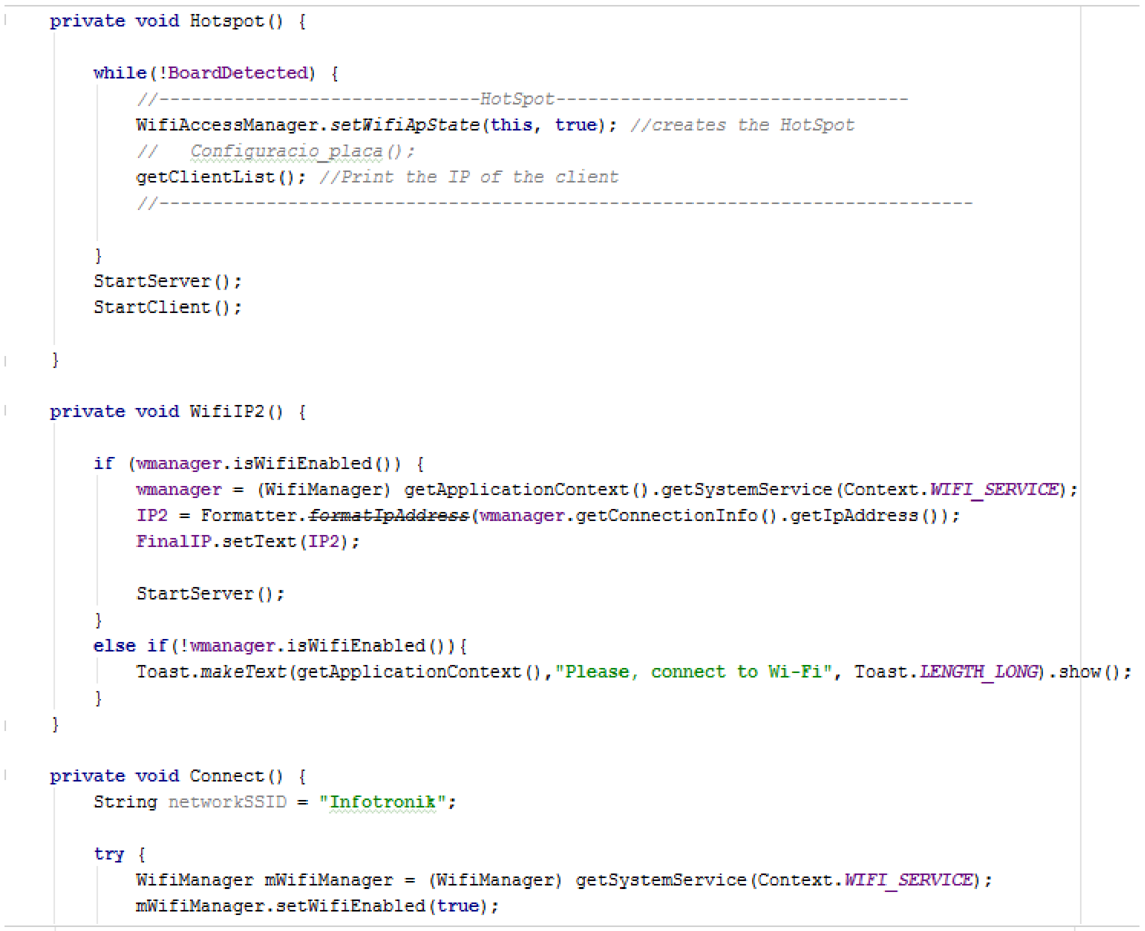
```
if (GrafCaract2 == 321){
    RightLeft = false;
    intent.putExtra("RightLeft",RightLeft);
    intent.putExtra("Scale2",D2SLCH3);
    intent.putExtra("Offset2",D2OLCH3);
    intent.putExtra("Unit2",D2ULCH3);
}
if (GrafCaract2 == 322){
    RightLeft = true;
    intent.putExtra("RightLeft",RightLeft);
    intent.putExtra("Scale2",D2SRCH3);
    intent.putExtra("Offset2",D2ORCH3);
    intent.putExtra("Unit2",D2URCH3);
}
if (GrafCaract1 == 411){
    RightLeft = false;
    intent.putExtra("RightLeft",RightLeft);
    intent.putExtra("Scale1",D1SLCH4);
    intent.putExtra("Offset1",D1OLCH4);
    intent.putExtra("Unit1",D1ULCH4);
}
if (GrafCaract1 == 412){
    RightLeft = true;
    intent.putExtra("RightLeft",RightLeft);
    intent.putExtra("Scale1",D1SRCH4);
    intent.putExtra("Offset1",D1ORCH4);
    intent.putExtra("Unit1",D1URCH4);
}
if (GrafCaract2 == 421){
    RightLeft = false;
    intent.putExtra("RightLeft",RightLeft);
    intent.putExtra("Scale2",D2SLCH4);
    intent.putExtra("Offset2",D2OLCH4);
    intent.putExtra("Unit2",D2ULCH4);
}
if (GrafCaract2 == 422){
    RightLeft = true;
    intent.putExtra("RightLeft",RightLeft);
    intent.putExtra("Scale2",D2SRCH4);
    intent.putExtra("Offset2",D2ORCH4);
    intent.putExtra("Unit2",D2URCH4);
}
if (GrafCaract1 == 511){
    RightLeft = false;
    intent.putExtra("RightLeft",RightLeft);
    intent.putExtra("Scale1",D1SLCH5);
    intent.putExtra("Offset1",D1OLCH5);
    intent.putExtra("Unit1",D1ULCH5);
}
if (GrafCaract1 == 512){
    RightLeft = true;
    intent.putExtra("RightLeft",RightLeft);
    intent.putExtra("Scale1",D1SRCH5);
    intent.putExtra("Offset1",D1ORCH5);
    intent.putExtra("Unit1",D1URCH5);
}
```



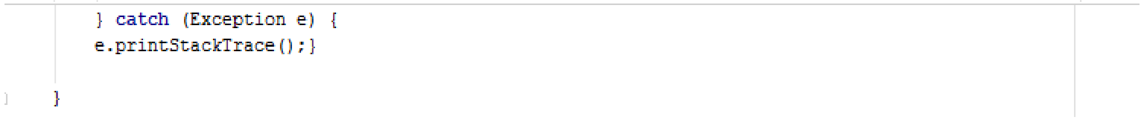
```
        if (GrafCaract2 == 521){
            RightLeft = false;
            intent.putExtra("RightLeft",RightLeft);
            intent.putExtra("Scale2",D2SLCH5);
            intent.putExtra("Offset2",D2OLCH5);
            intent.putExtra("Unit2",D2ULCH5);
        }
        if (GrafCaract2 == 522){
            RightLeft = true;
            intent.putExtra("RightLeft",RightLeft);
            intent.putExtra("Scale2",D2SRCH5);
            intent.putExtra("Offset2",D2ORCH5);
            intent.putExtra("Unit2",D2URCH5);
        }
        if (GrafCaract1 == 611){
            RightLeft = false;
            intent.putExtra("RightLeft",RightLeft);
            intent.putExtra("Scale1",D1SLCH6);
            intent.putExtra("Offset1",D1OLCH6);
            intent.putExtra("Unit1",D1ULCH6);
        }
        if (GrafCaract1 == 612) {
            RightLeft = true;
            intent.putExtra("RightLeft",RightLeft);
            intent.putExtra("Scale1", D1SRCH6);
            intent.putExtra("Offset1", D1ORCH6);
            intent.putExtra("Unit1", D1URCH6);
        }
        if (GrafCaract2 == 621){
            RightLeft = false;
            intent.putExtra("RightLeft",RightLeft);
            intent.putExtra("Scale2",D2SLCH6);
            intent.putExtra("Offset2",D2OLCH6);
            intent.putExtra("Unit2",D2ULCH6);
        }
        if (GrafCaract2 == 622){
            RightLeft = true;
            intent.putExtra("RightLeft",RightLeft);
            intent.putExtra("RightLeft",D2SRCH6);
            intent.putExtra("Offset2",D2ORCH6);
            intent.putExtra("Unit2",D2URCH6);
        }

        intent.putExtra("GrafCaract1",GrafCaract1);
        intent.putExtra("GrafCaract2",GrafCaract2);
        intent.putExtra("ipPlaca",IPBoard);
        intent.putExtra("BackButton",BackButton);

        startActivity(intent);
    }
```



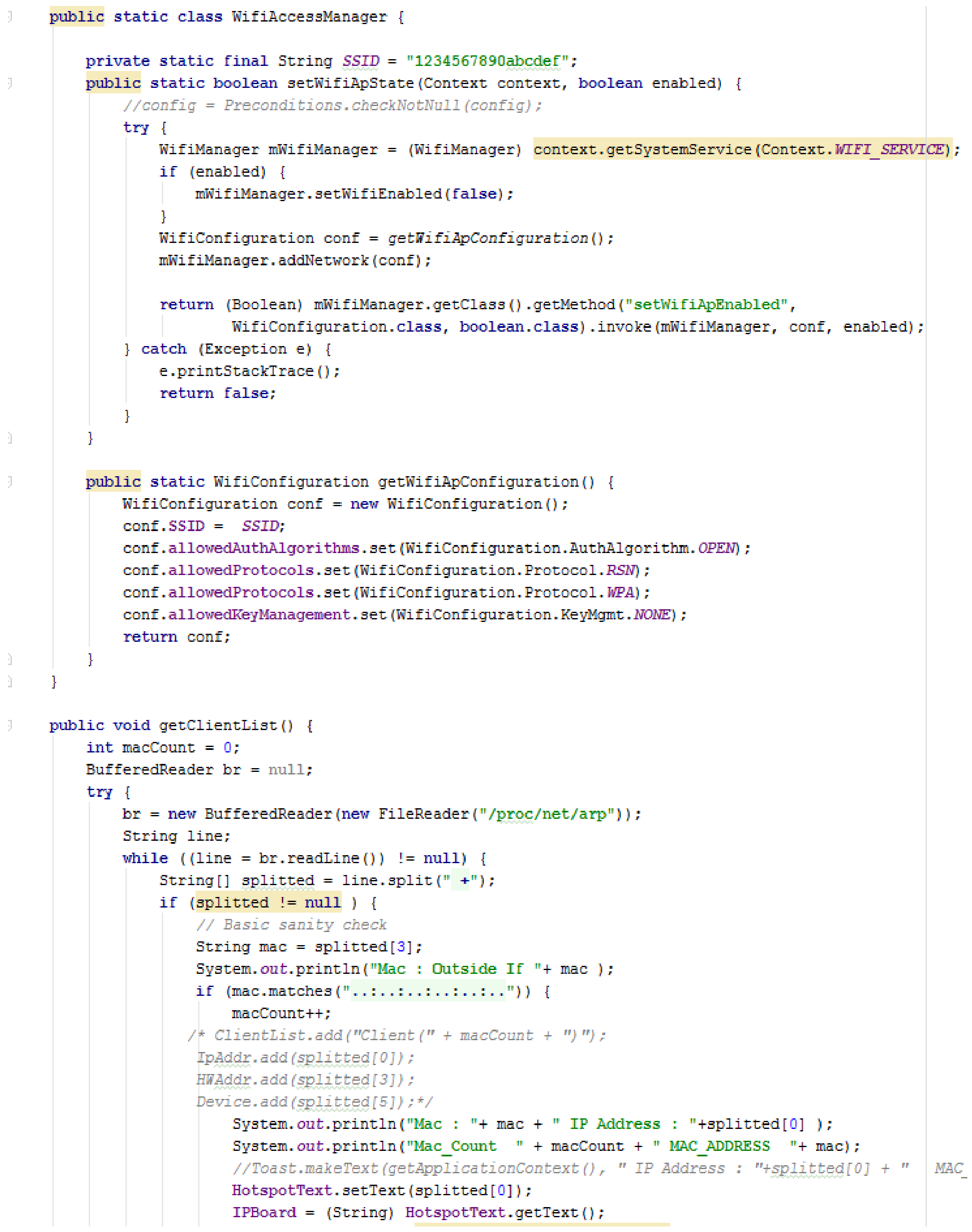
```
    private void  StartBoardCommunication() {

        WifiIP1();
        Hotspot();

        boolean a = false;
        while(!a) {
            try {

                //   clientThread.start();
                //new Thread(new ClientThread()).start();
                str = IP1;
                PrintWriter out = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()))
                        , true);
                out.println(str);
                str = WifiId;
                PrintWriter out1 = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()))
                        , true);
                out1.println(str);
                a = true;
            } catch (UnknownHostException e) {
                e.printStackTrace();
                StartClient();
            } catch (IOException e) {
                e.printStackTrace();
                StartClient();
            } catch (Exception e) {
                e.printStackTrace();
                StartClient();
            }
        }

        ApManager.isApOn(this); // check Ap state :boolean
        ApManager.configApState(this);

        Connect();
/*        while (count < 12000000){     //comentar per Debug fent tu el delay, descomentar per Run
            count++;
        }   // */
        WifiIP2();
        StartServer();
    }


    private void WifiIP1() {
            if (wmanager.isWifiEnabled()) {
                IP1 = Formatter.~~formatIpAddress~~(wmanager.getConnectionInfo().getIpAddress());
                InitialIP.setText(IP1);
                WifiId = wmanager.getConnectionInfo().getSSID();

            }
            else if(!wmanager.isWifiEnabled()){
                Toast.makeText(getApplicationContext(),"Please, connect to Wi-Fi", Toast.LENGTH_LONG).show();
            }
    }
```



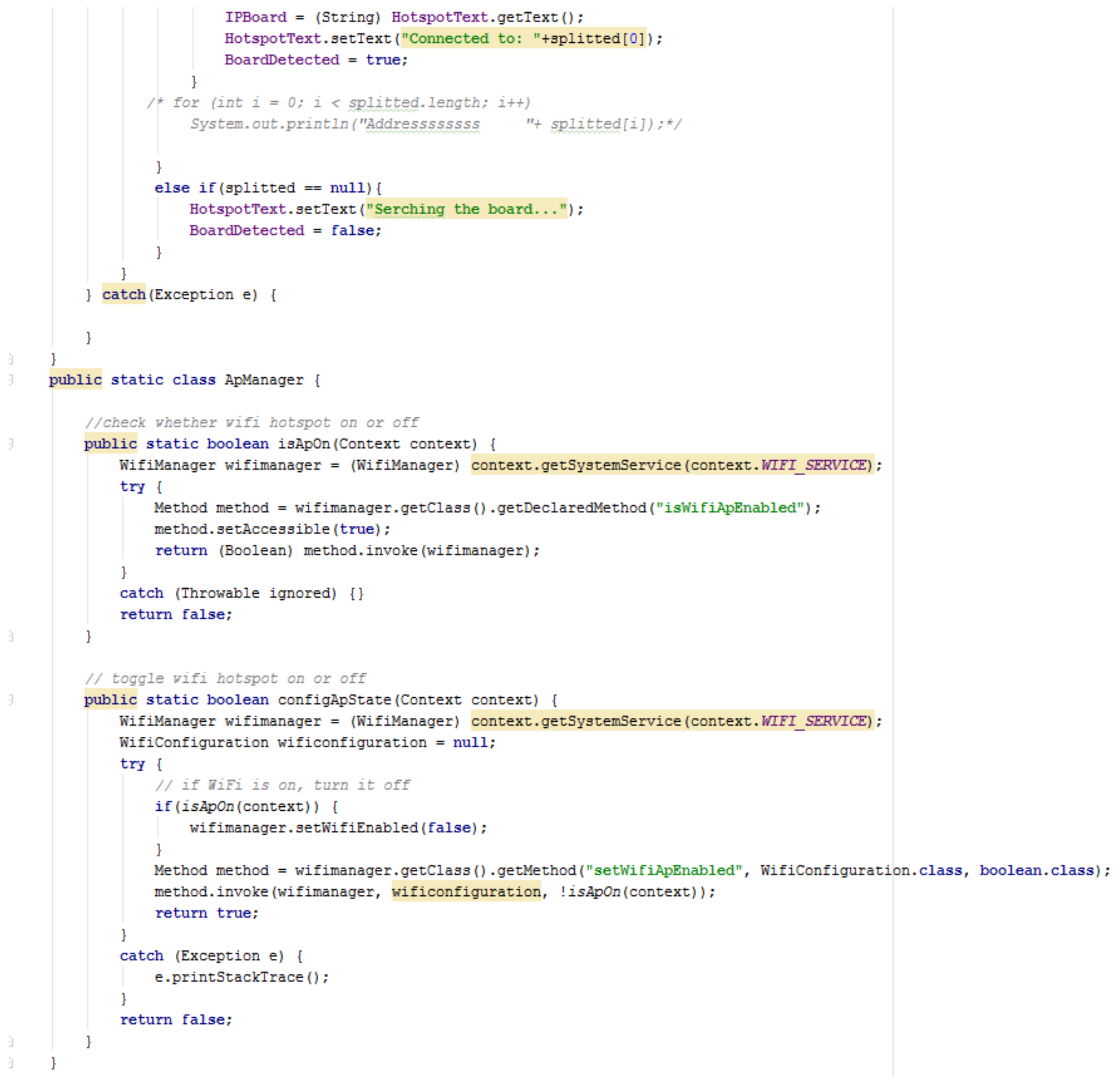
```
private void Hotspot() {

    while(!BoardDetected) {
        //-----------------------------HotSpot---------------------------------
        WifiAccessManager.setWifiApState(this, true); //creates the HotSpot
        //   Configuracio_placa();
        getClientList(); //Print the IP of the client
        //----------------------------------------------------------------------------

    }
    StartServer();
    StartClient();

}

private void WifiIP2() {

    if (wmanager.isWifiEnabled()) {
        wmanager = (WifiManager) getApplicationContext().getSystemService(Context.WIFI_SERVICE);
        IP2 = Formatter.~~formatIpAddress~~(wmanager.getConnectionInfo().getIpAddress());
        FinalIP.setText(IP2);

        StartServer();
    }
    else if(!wmanager.isWifiEnabled()){
        Toast.makeText(getApplicationContext(),"Please, connect to Wi-Fi", Toast.LENGTH_LONG).show();
    }
}

private void Connect() {
    String networkSSID = "Infotronik";

    try {
        WifiManager mWifiManager = (WifiManager) getSystemService(Context.WIFI_SERVICE);
        mWifiManager.setWifiEnabled(true);
```



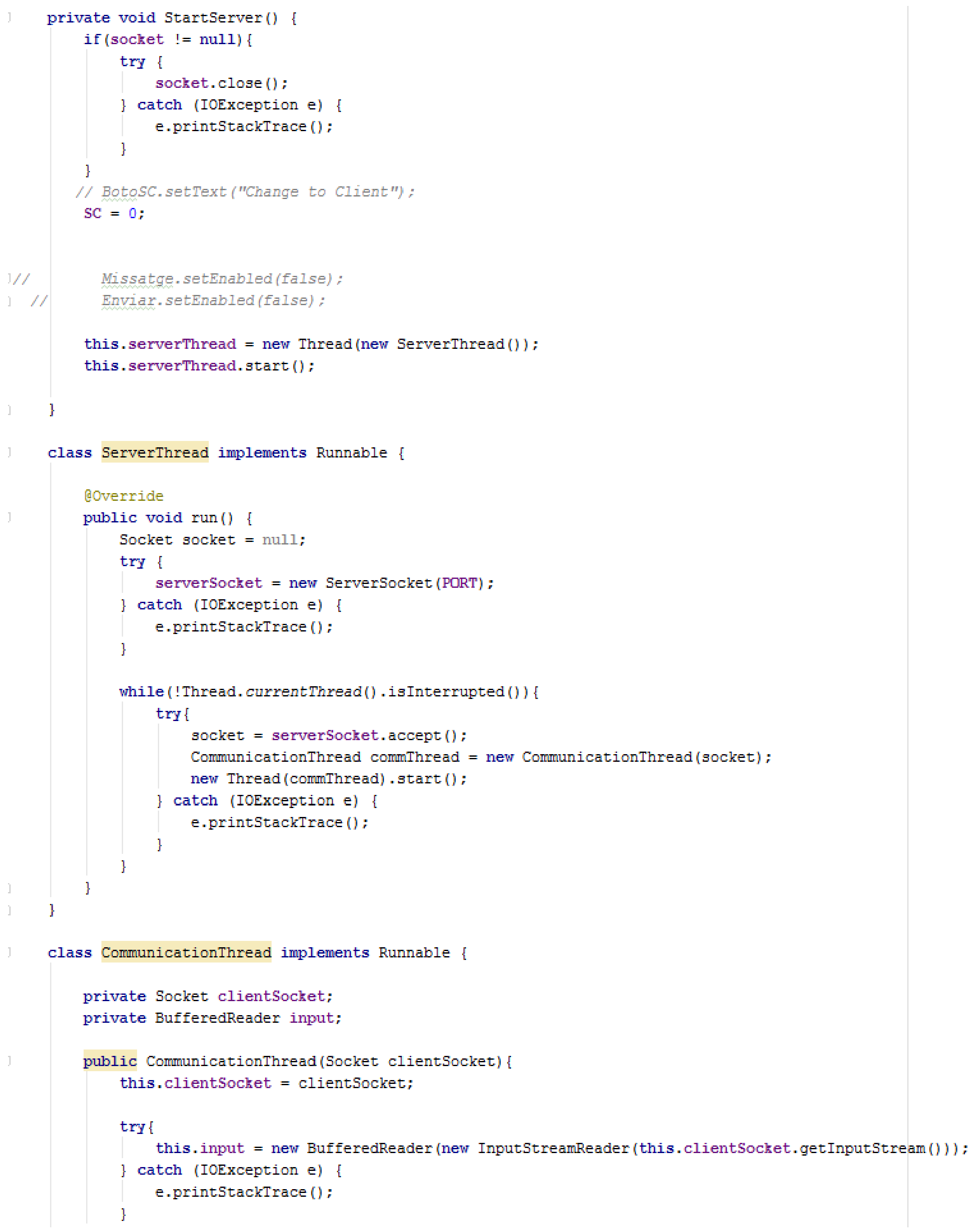
```
    } catch (Exception e) {
    e.printStackTrace();}

}
```



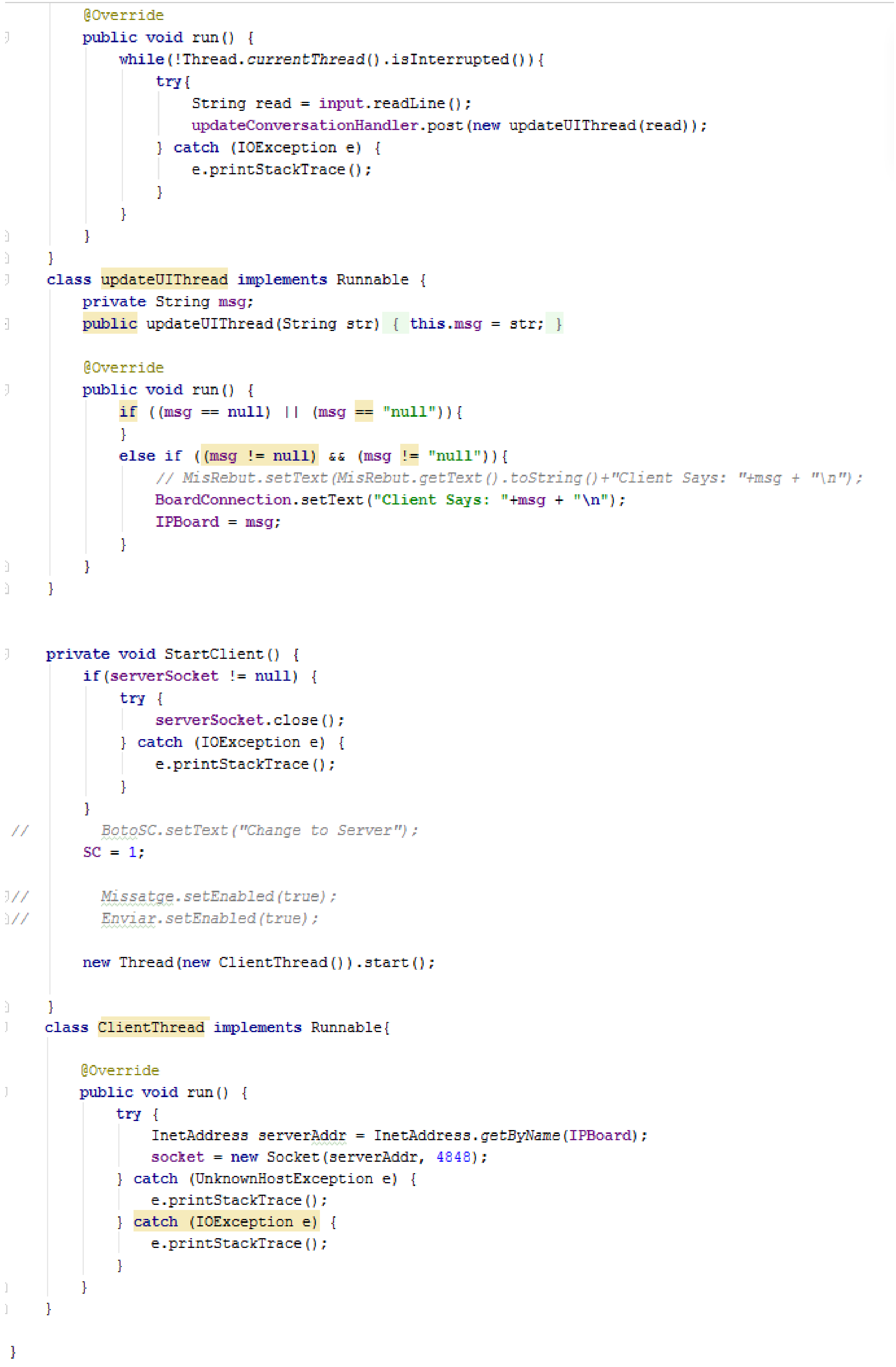
```java
public static class WifiAccessManager {

    private static final String SSID = "1234567890abcdef";
    public static boolean setWifiApState(Context context, boolean enabled) {
        //config = Preconditions.checkNotNull(config);
        try {
            WifiManager mWifiManager = (WifiManager) context.getSystemService(Context.WIFI_SERVICE);
            if (enabled) {
                mWifiManager.setWifiEnabled(false);
            }
            WifiConfiguration conf = getWifiApConfiguration();
            mWifiManager.addNetwork(conf);

            return (Boolean) mWifiManager.getClass().getMethod("setWifiApEnabled",
                    WifiConfiguration.class, boolean.class).invoke(mWifiManager, conf, enabled);
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }

    public static WifiConfiguration getWifiApConfiguration() {
        WifiConfiguration conf = new WifiConfiguration();
        conf.SSID =  SSID;
        conf.allowedAuthAlgorithms.set(WifiConfiguration.AuthAlgorithm.OPEN);
        conf.allowedProtocols.set(WifiConfiguration.Protocol.RSN);
        conf.allowedProtocols.set(WifiConfiguration.Protocol.WPA);
        conf.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.NONE);
        return conf;
    }
}

public void getClientList() {
    int macCount = 0;
    BufferedReader br = null;
    try {
        br = new BufferedReader(new FileReader("/proc/net/arp"));
        String line;
        while ((line = br.readLine()) != null) {
            String[] splitted = line.split(" +");
            if (splitted != null ) {
                // Basic sanity check
                String mac = splitted[3];
                System.out.println("Mac : Outside If "+ mac );
                if (mac.matches("..:..:..:..:..:..")) {
                    macCount++;
               /* ClientList.add("Client(" + macCount + ")");
                IpAddr.add(splitted[0]);
                HWAddr.add(splitted[3]);
                Device.add(splitted[5]);*/
                    System.out.println("Mac : "+ mac + " IP Address : "+splitted[0] );
                    System.out.println("Mac_Count  " + macCount + " MAC_ADDRESS  "+ mac);
                    //Toast.makeText(getApplicationContext(), " IP Address : "+splitted[0] + "   MAC_
                    HotspotText.setText(splitted[0]);
                    IPBoard = (String) HotspotText.getText();
```



```
                    IPBoard = (String) HotspotText.getText();
                    HotspotText.setText("Connected to: "+splitted[0]);
                    BoardDetected = true;
                }
          /* for (int i = 0; i < splitted.length; i++)
                System.out.println("Addressssssss     "+ splitted[i]);*/

            }
            else if(splitted == null){
                HotspotText.setText("Serching the board...");
                BoardDetected = false;
            }
        }
    } catch(Exception e) {

    }
}
public static class ApManager {

    //check whether wifi hotspot on or off
    public static boolean isApOn(Context context) {
        WifiManager wifimanager = (WifiManager) context.getSystemService(context.WIFI_SERVICE);
        try {
            Method method = wifimanager.getClass().getDeclaredMethod("isWifiApEnabled");
            method.setAccessible(true);
            return (Boolean) method.invoke(wifimanager);
        }
        catch (Throwable ignored) {}
        return false;
    }

    // toggle wifi hotspot on or off
    public static boolean configApState(Context context) {
        WifiManager wifimanager = (WifiManager) context.getSystemService(context.WIFI_SERVICE);
        WifiConfiguration wificonfiguration = null;
        try {
            // if WiFi is on, turn it off
            if(isApOn(context)) {
                wifimanager.setWifiEnabled(false);
            }
            Method method = wifimanager.getClass().getMethod("setWifiApEnabled", WifiConfiguration.class, boolean.class);
            method.invoke(wifimanager, wificonfiguration, !isApOn(context));
            return true;
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        return false;
    }
}
```

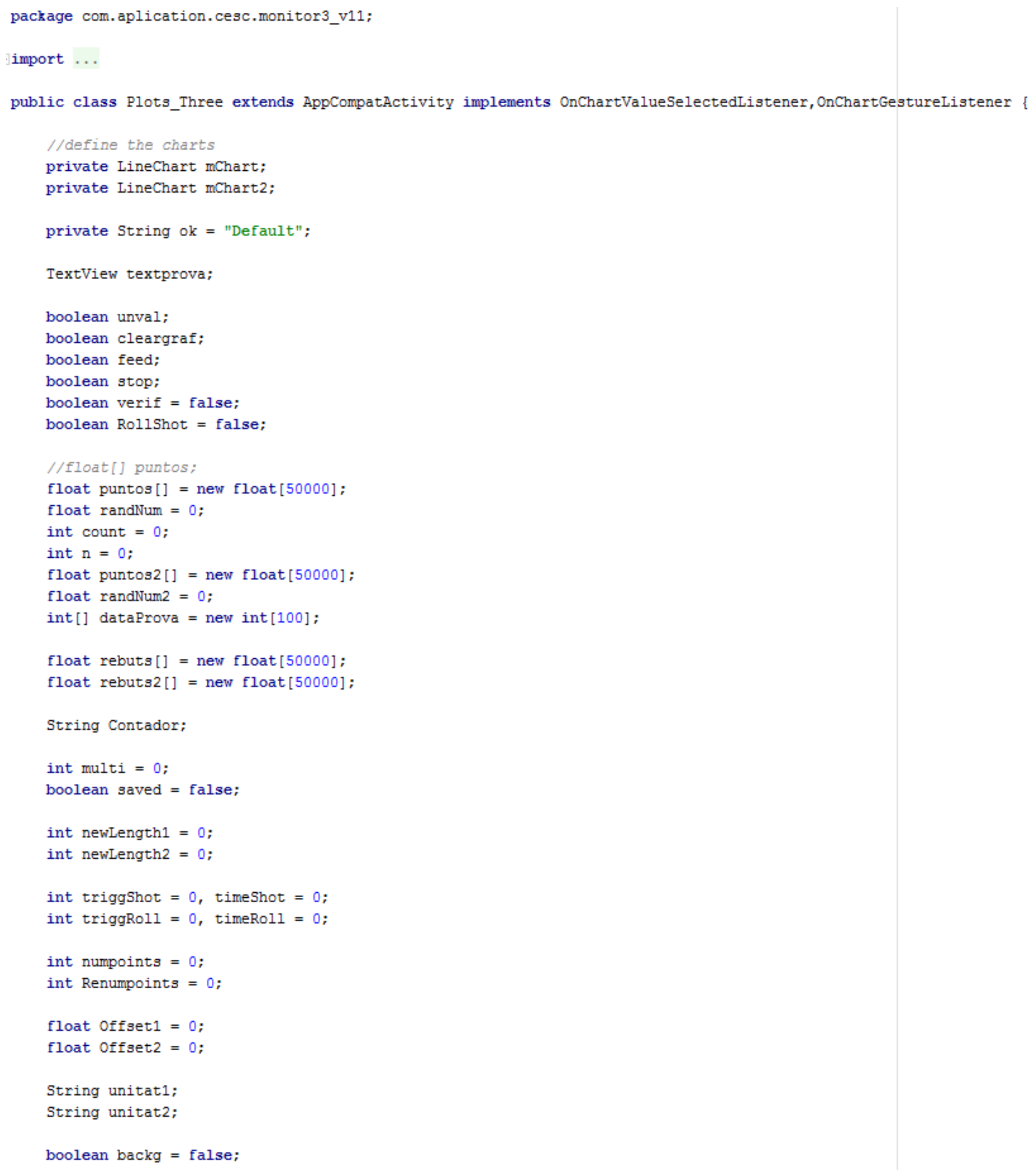


```
    private void StartServer() {
        if(socket != null){
            try {
                socket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
       // BotoSC.setText("Change to Client");
        SC = 0;


//          Missatge.setEnabled(false);
  //        Enviar.setEnabled(false);

        this.serverThread = new Thread(new ServerThread());
        this.serverThread.start();

    }

    class ServerThread implements Runnable {

        @Override
        public void run() {
            Socket socket = null;
            try {
                serverSocket = new ServerSocket(PORT);
            } catch (IOException e) {
                e.printStackTrace();
            }

            while(!Thread.currentThread().isInterrupted()){
                try{
                    socket = serverSocket.accept();
                    CommunicationThread commThread = new CommunicationThread(socket);
                    new Thread(commThread).start();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    class CommunicationThread implements Runnable {

        private Socket clientSocket;
        private BufferedReader input;

        public CommunicationThread(Socket clientSocket){
            this.clientSocket = clientSocket;

            try{
                this.input = new BufferedReader(new InputStreamReader(this.clientSocket.getInputStream()));
            } catch (IOException e) {
                e.printStackTrace();
            }
```

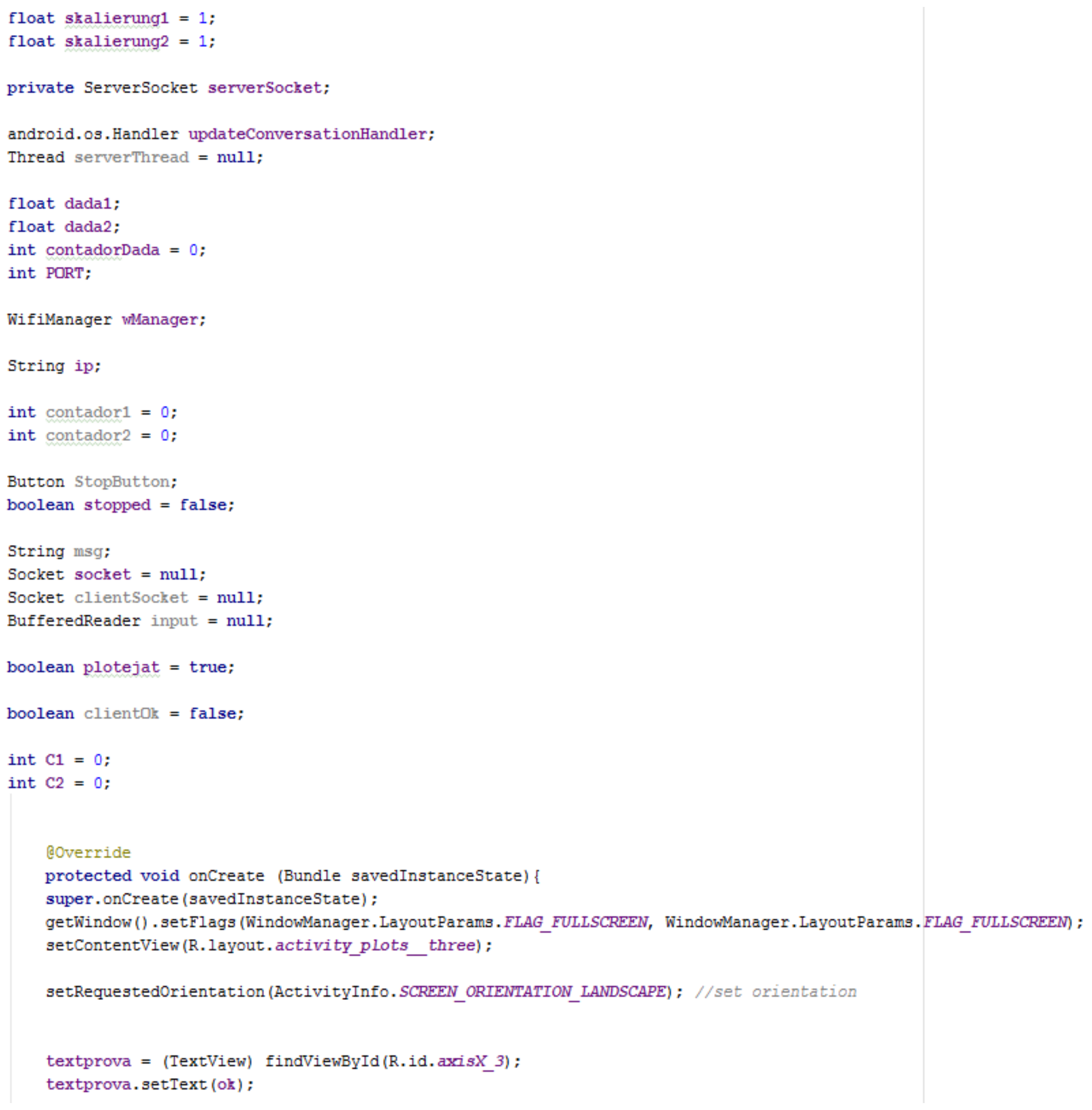


```
        @Override
        public void run() {
            while(!Thread.currentThread().isInterrupted()){
                try{
                    String read = input.readLine();
                    updateConversationHandler.post(new updateUIThread(read));
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
    class updateUIThread implements Runnable {
        private String msg;
        public updateUIThread(String str) { this.msg = str; }

        @Override
        public void run() {
            if ((msg == null) || (msg == "null")){
            }
            else if ((msg != null) && (msg != "null")){
                // MisRebut.setText(MisRebut.getText().toString()+"Client Says: "+msg + "\n");
                BoardConnection.setText("Client Says: "+msg + "\n");
                IPBoard = msg;
            }
        }
    }


    private void StartClient() {
        if(serverSocket != null) {
            try {
                serverSocket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
//        BotoSC.setText("Change to Server");
        SC = 1;

//        Missatge.setEnabled(true);
//        Enviar.setEnabled(true);

        new Thread(new ClientThread()).start();

    }
    class ClientThread implements Runnable{

        @Override
        public void run() {
            try {
                InetAddress serverAddr = InetAddress.getByName(IPBoard);
                socket = new Socket(serverAddr, 4848);
            } catch (UnknownHostException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

}
```

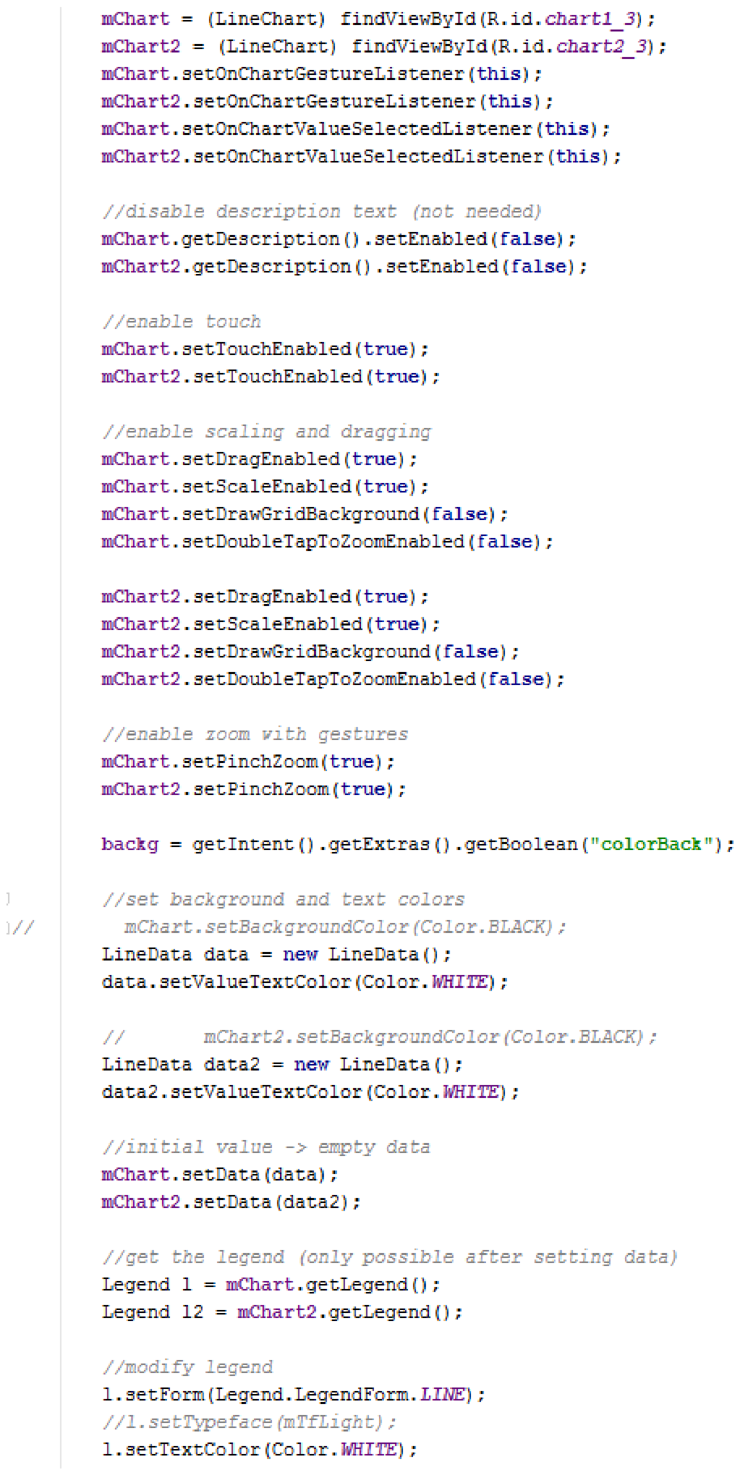
Plots_Three



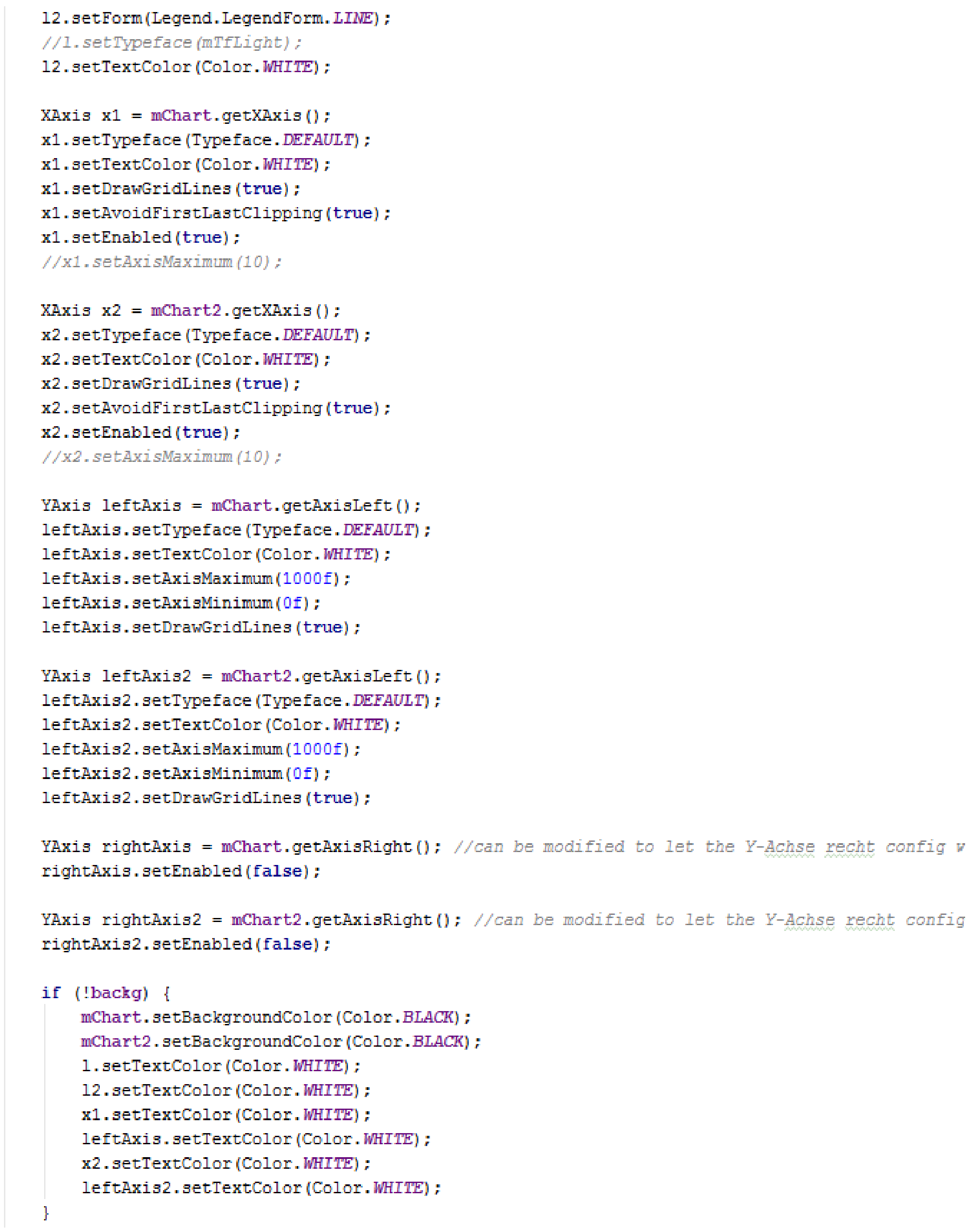
```
package com.aplication.cesc.monitor3_v11;

import ...

public class Plots_Three extends AppCompatActivity implements OnChartValueSelectedListener,OnChartGestureListener {

    //define the charts
    private LineChart mChart;
    private LineChart mChart2;

    private String ok = "Default";

    TextView textprova;

    boolean unval;
    boolean cleargraf;
    boolean feed;
    boolean stop;
    boolean verif = false;
    boolean RollShot = false;

    //float[] puntos;
    float puntos[] = new float[50000];
    float randNum = 0;
    int count = 0;
    int n = 0;
    float puntos2[] = new float[50000];
    float randNum2 = 0;
    int[] dataProva = new int[100];

    float rebuts[] = new float[50000];
    float rebuts2[] = new float[50000];

    String Contador;

    int multi = 0;
    boolean saved = false;

    int newLength1 = 0;
    int newLength2 = 0;

    int triggShot = 0, timeShot = 0;
    int triggRoll = 0, timeRoll = 0;

    int numpoints = 0;
    int Renumpoints = 0;

    float Offset1 = 0;
    float Offset2 = 0;

    String unitat1;
    String unitat2;

    boolean backg = false;
```



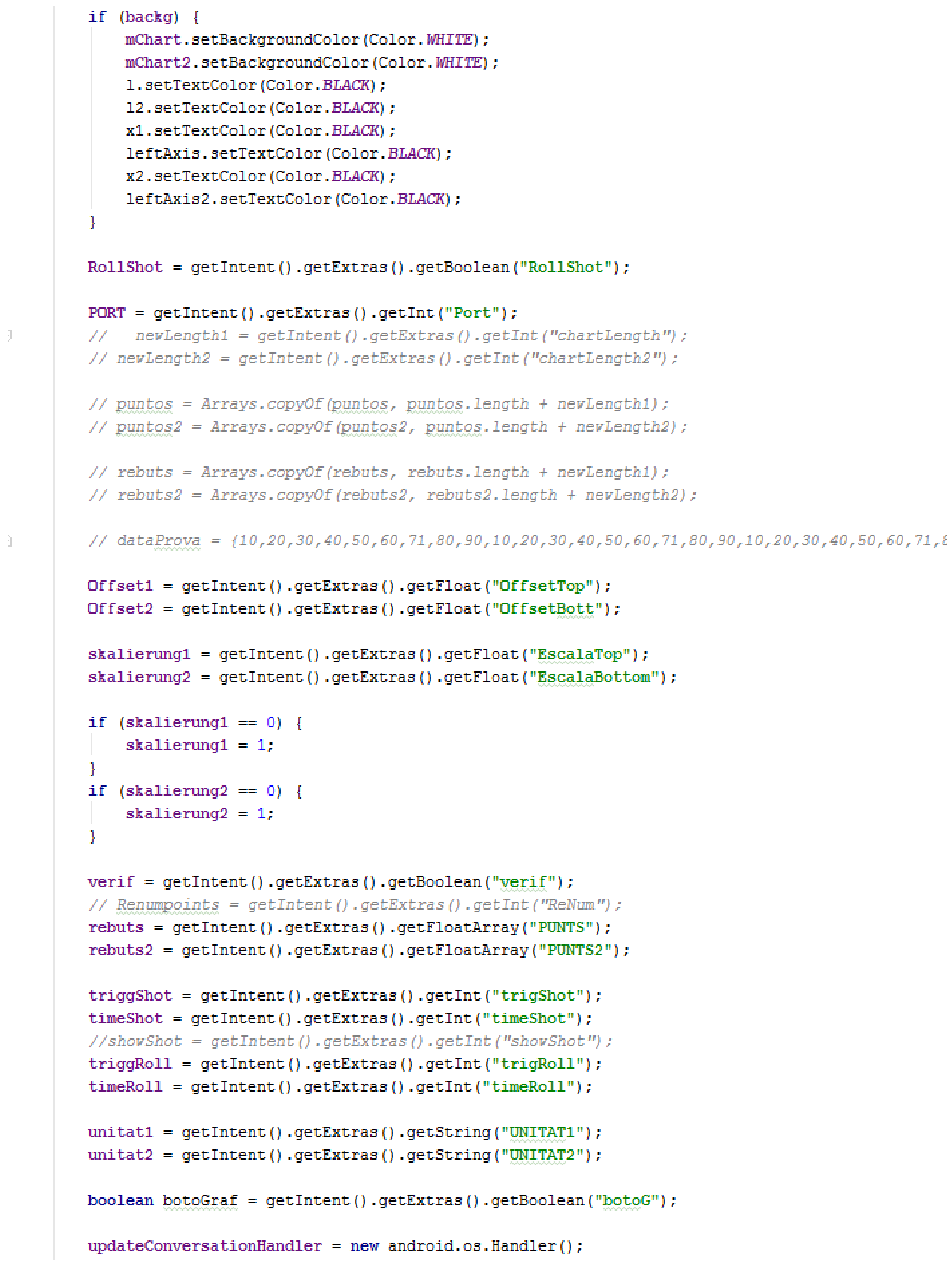
```
float skalierung1 = 1;
float skalierung2 = 1;

private ServerSocket serverSocket;

android.os.Handler updateConversationHandler;
Thread serverThread = null;

float dada1;
float dada2;
int contadorDada = 0;
int PORT;

WifiManager wManager;

String ip;

int contador1 = 0;
int contador2 = 0;

Button StopButton;
boolean stopped = false;

String msg;
Socket socket = null;
Socket clientSocket = null;
BufferedReader input = null;

boolean plotejat = true;

boolean clientOk = false;

int C1 = 0;
int C2 = 0;


    @Override
    protected void onCreate (Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, WindowManager.LayoutParams.FLAG_FULLSCREEN);
    setContentView(R.layout.activity_plots__three);

    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE); //set orientation


    textprova = (TextView) findViewById(R.id.axisX_3);
    textprova.setText(ok);
```



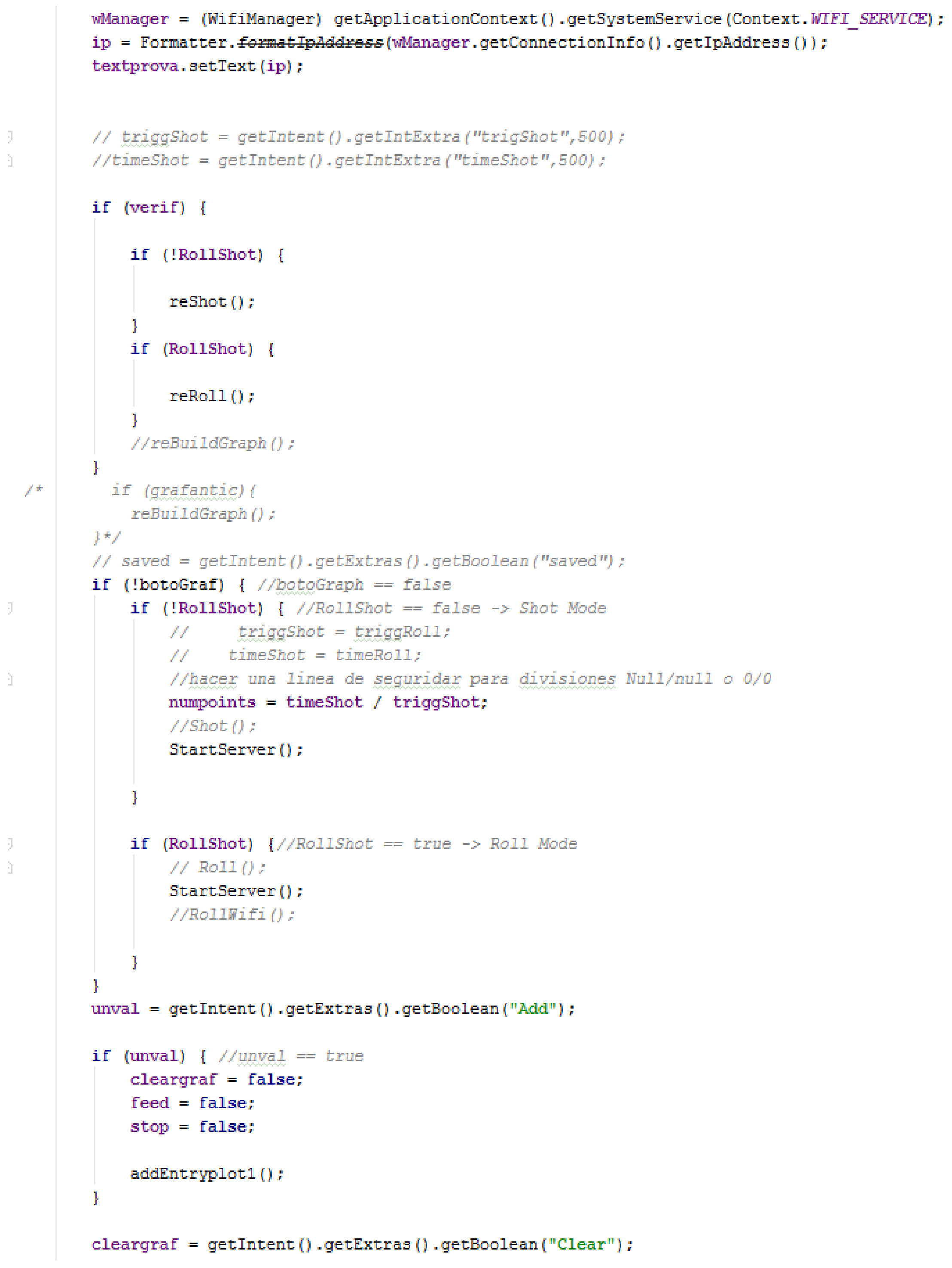
```java
mChart = (LineChart) findViewById(R.id.chart1_3);
mChart2 = (LineChart) findViewById(R.id.chart2_3);
mChart.setOnChartGestureListener(this);
mChart2.setOnChartGestureListener(this);
mChart.setOnChartValueSelectedListener(this);
mChart2.setOnChartValueSelectedListener(this);

//disable description text (not needed)
mChart.getDescription().setEnabled(false);
mChart2.getDescription().setEnabled(false);

//enable touch
mChart.setTouchEnabled(true);
mChart2.setTouchEnabled(true);

//enable scaling and dragging
mChart.setDragEnabled(true);
mChart.setScaleEnabled(true);
mChart.setDrawGridBackground(false);
mChart.setDoubleTapToZoomEnabled(false);

mChart2.setDragEnabled(true);
mChart2.setScaleEnabled(true);
mChart2.setDrawGridBackground(false);
mChart2.setDoubleTapToZoomEnabled(false);

//enable zoom with gestures
mChart.setPinchZoom(true);
mChart2.setPinchZoom(true);

backg = getIntent().getExtras().getBoolean("colorBack");

//set background and text colors
  mChart.setBackgroundColor(Color.BLACK);
LineData data = new LineData();
data.setValueTextColor(Color.WHITE);

//        mChart2.setBackgroundColor(Color.BLACK);
LineData data2 = new LineData();
data2.setValueTextColor(Color.WHITE);

//initial value -> empty data
mChart.setData(data);
mChart2.setData(data2);

//get the legend (only possible after setting data)
Legend l = mChart.getLegend();
Legend l2 = mChart2.getLegend();

//modify legend
l.setForm(Legend.LegendForm.LINE);
//l.setTypeface(mTfLight);
l.setTextColor(Color.WHITE);
```



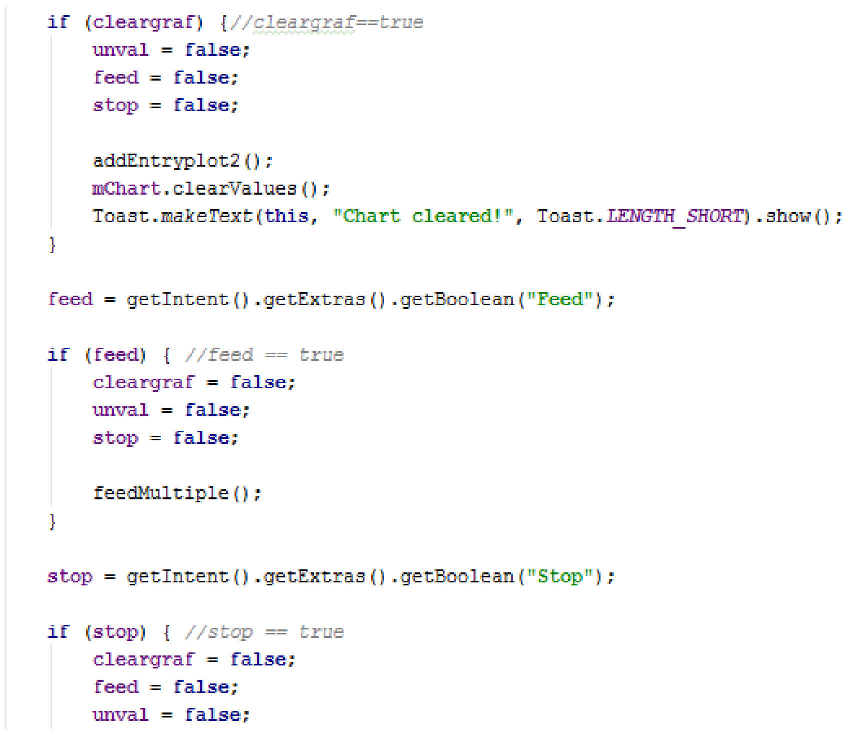
```
l2.setForm(Legend.LegendForm.LINE);
//l.setTypeface(mTfLight);
l2.setTextColor(Color.WHITE);

XAxis x1 = mChart.getXAxis();
x1.setTypeface(Typeface.DEFAULT);
x1.setTextColor(Color.WHITE);
x1.setDrawGridLines(true);
x1.setAvoidFirstLastClipping(true);
x1.setEnabled(true);
//x1.setAxisMaximum(10);

XAxis x2 = mChart2.getXAxis();
x2.setTypeface(Typeface.DEFAULT);
x2.setTextColor(Color.WHITE);
x2.setDrawGridLines(true);
x2.setAvoidFirstLastClipping(true);
x2.setEnabled(true);
//x2.setAxisMaximum(10);

YAxis leftAxis = mChart.getAxisLeft();
leftAxis.setTypeface(Typeface.DEFAULT);
leftAxis.setTextColor(Color.WHITE);
leftAxis.setAxisMaximum(1000f);
leftAxis.setAxisMinimum(0f);
leftAxis.setDrawGridLines(true);

YAxis leftAxis2 = mChart2.getAxisLeft();
leftAxis2.setTypeface(Typeface.DEFAULT);
leftAxis2.setTextColor(Color.WHITE);
leftAxis2.setAxisMaximum(1000f);
leftAxis2.setAxisMinimum(0f);
leftAxis2.setDrawGridLines(true);

YAxis rightAxis = mChart.getAxisRight(); //can be modified to let the Y-Achse recht config w
rightAxis.setEnabled(false);

YAxis rightAxis2 = mChart2.getAxisRight(); //can be modified to let the Y-Achse recht config
rightAxis2.setEnabled(false);

if (!backg) {
    mChart.setBackgroundColor(Color.BLACK);
    mChart2.setBackgroundColor(Color.BLACK);
    l.setTextColor(Color.WHITE);
    l2.setTextColor(Color.WHITE);
    x1.setTextColor(Color.WHITE);
    leftAxis.setTextColor(Color.WHITE);
    x2.setTextColor(Color.WHITE);
    leftAxis2.setTextColor(Color.WHITE);
}
```



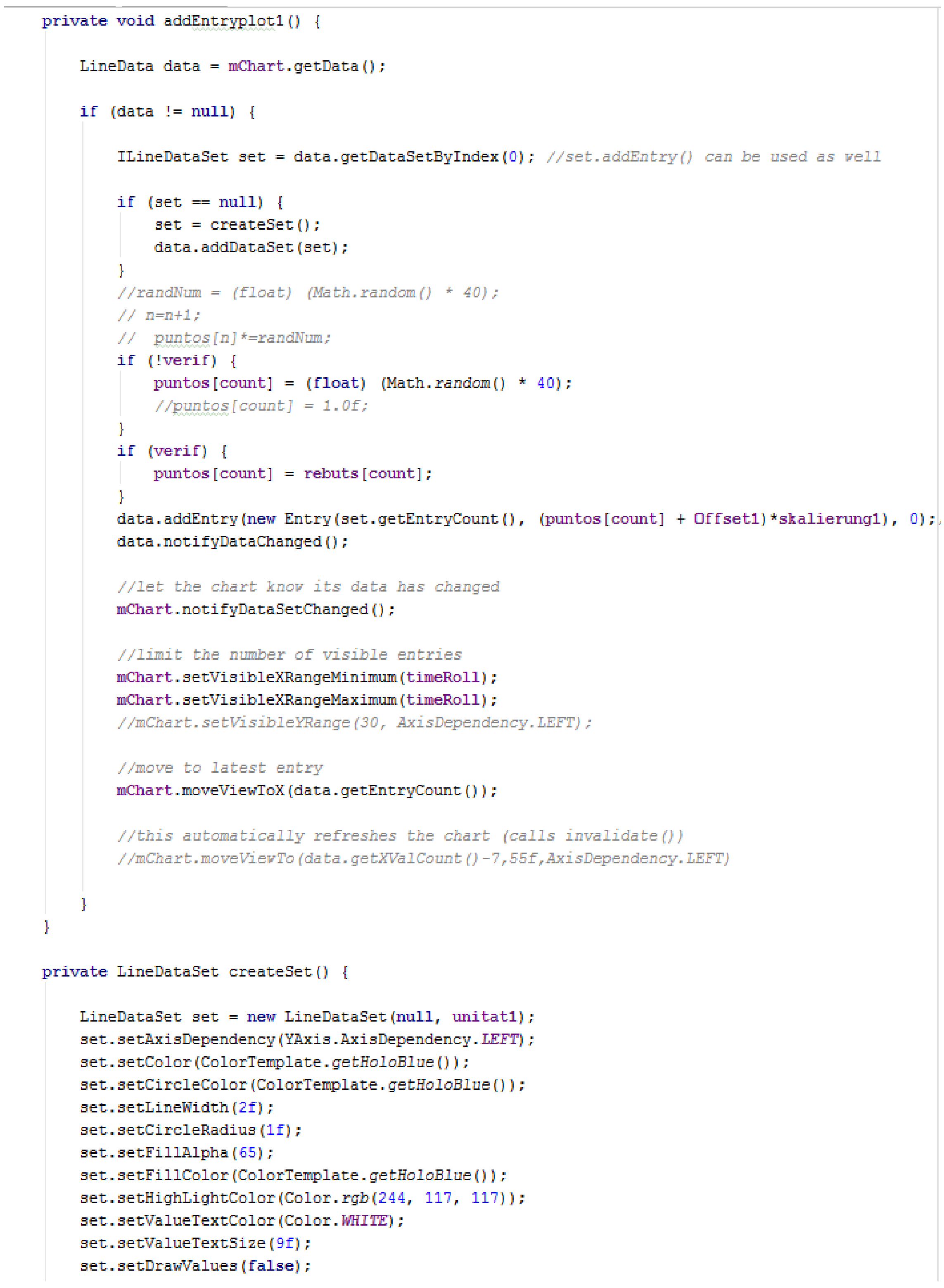
```
if (backg) {
    mChart.setBackgroundColor(Color.WHITE);
    mChart2.setBackgroundColor(Color.WHITE);
    l.setTextColor(Color.BLACK);
    l2.setTextColor(Color.BLACK);
    x1.setTextColor(Color.BLACK);
    leftAxis.setTextColor(Color.BLACK);
    x2.setTextColor(Color.BLACK);
    leftAxis2.setTextColor(Color.BLACK);
}

RollShot = getIntent().getExtras().getBoolean("RollShot");

PORT = getIntent().getExtras().getInt("Port");
//   newLength1 = getIntent().getExtras().getInt("chartLength");
// newLength2 = getIntent().getExtras().getInt("chartLength2");

// puntos = Arrays.copyOf(puntos, puntos.length + newLength1);
// puntos2 = Arrays.copyOf(puntos2, puntos.length + newLength2);

// rebuts = Arrays.copyOf(rebuts, rebuts.length + newLength1);
// rebuts2 = Arrays.copyOf(rebuts2, rebuts2.length + newLength2);

// dataProva = {10,20,30,40,50,60,71,80,90,10,20,30,40,50,60,71,80,90,10,20,30,40,50,60,71,

Offset1 = getIntent().getExtras().getFloat("OffsetTop");
Offset2 = getIntent().getExtras().getFloat("OffsetBott");

skalierung1 = getIntent().getExtras().getFloat("EscalaTop");
skalierung2 = getIntent().getExtras().getFloat("EscalaBottom");

if (skalierung1 == 0) {
    skalierung1 = 1;
}
if (skalierung2 == 0) {
    skalierung2 = 1;
}

verif = getIntent().getExtras().getBoolean("verif");
// Renumpoints = getIntent().getExtras().getInt("ReNum");
rebuts = getIntent().getExtras().getFloatArray("PUNTS");
rebuts2 = getIntent().getExtras().getFloatArray("PUNTS2");

triggShot = getIntent().getExtras().getInt("trigShot");
timeShot = getIntent().getExtras().getInt("timeShot");
//showShot = getIntent().getExtras().getInt("showShot");
triggRoll = getIntent().getExtras().getInt("trigRoll");
timeRoll = getIntent().getExtras().getInt("timeRoll");

unitat1 = getIntent().getExtras().getString("UNITAT1");
unitat2 = getIntent().getExtras().getString("UNITAT2");

boolean botoGraf = getIntent().getExtras().getBoolean("botoG");

updateConversationHandler = new android.os.Handler();
```



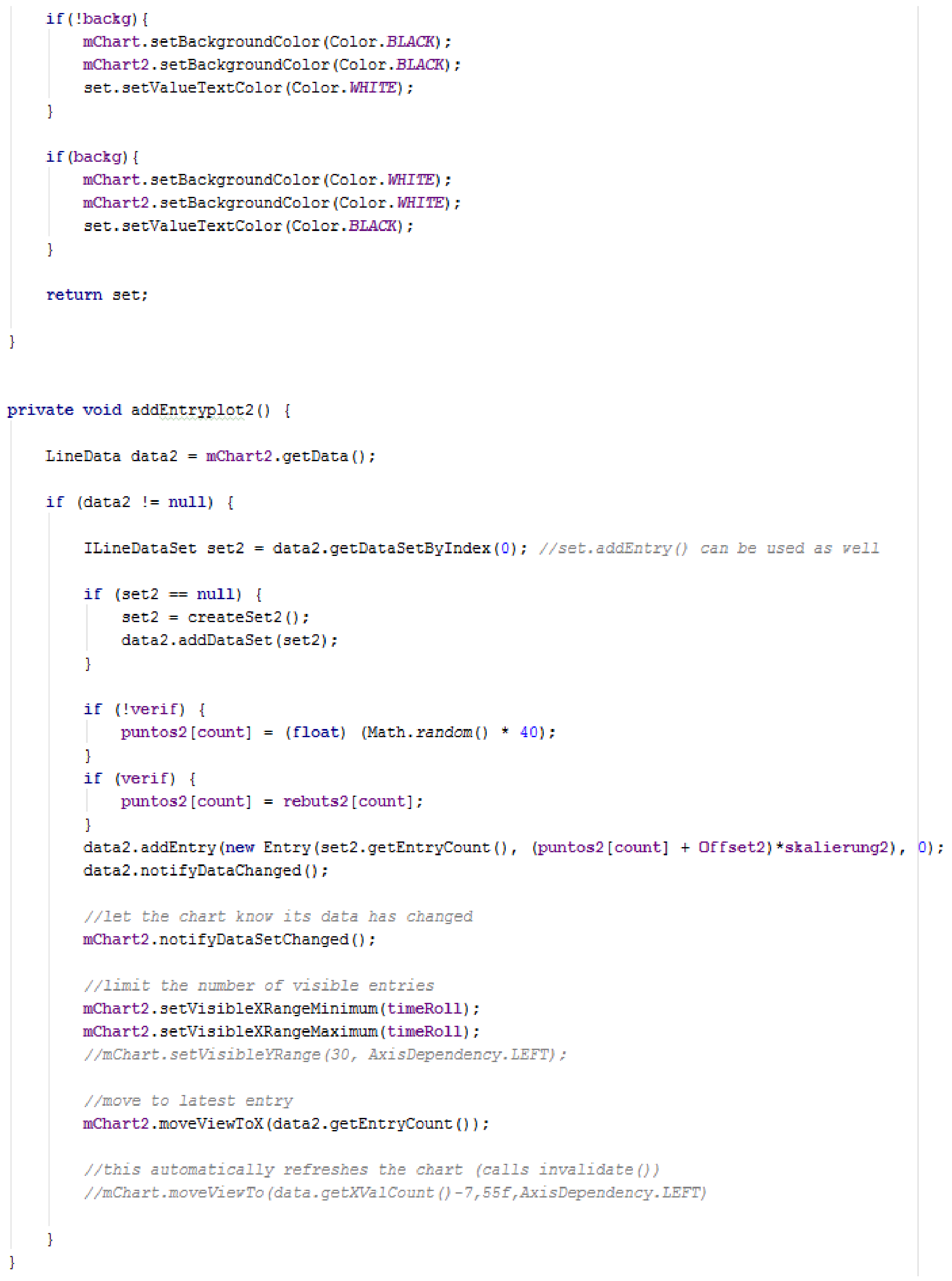
```
wManager = (WifiManager) getApplicationContext().getSystemService(Context.WIFI_SERVICE);
ip = Formatter.~~formatIpAddress~~(wManager.getConnectionInfo().getIpAddress());
textprova.setText(ip);


// triggShot = getIntent().getIntExtra("trigShot",500);
//timeShot = getIntent().getIntExtra("timeShot",500);

if (verif) {

    if (!RollShot) {

        reShot();
    }
    if (RollShot) {

        reRoll();
    }
    //reBuildGraph();
}
/*  if (grafantic){
    reBuildGraph();
}*/
// saved = getIntent().getExtras().getBoolean("saved");
if (!botoGraf) { //botoGraph == false
    if (!RollShot) { //RollShot == false -> Shot Mode
        //      triggShot = triggRoll;
        //     timeShot = timeRoll;
        //hacer una linea de seguridar para divisiones Null/null o 0/0
        numpoints = timeShot / triggShot;
        //Shot();
        StartServer();

    }

    if (RollShot) {//RollShot == true -> Roll Mode
        // Roll();
        StartServer();
        //RollWifi();

    }
}
unval = getIntent().getExtras().getBoolean("Add");

if (unval) { //unval == true
    cleargraf = false;
    feed = false;
    stop = false;

    addEntryplot1();
}

cleargraf = getIntent().getExtras().getBoolean("Clear");
```



```
if (cleargraf) {//cleargraf==true
    unval = false;
    feed = false;
    stop = false;

    addEntryplot2();
    mChart.clearValues();
    Toast.makeText(this, "Chart cleared!", Toast.LENGTH_SHORT).show();
}

feed = getIntent().getExtras().getBoolean("Feed");

if (feed) { //feed == true
    cleargraf = false;
    unval = false;
    stop = false;

    feedMultiple();
}

stop = getIntent().getExtras().getBoolean("Stop");

if (stop) { //stop == true
    cleargraf = false;
    feed = false;
    unval = false;
```

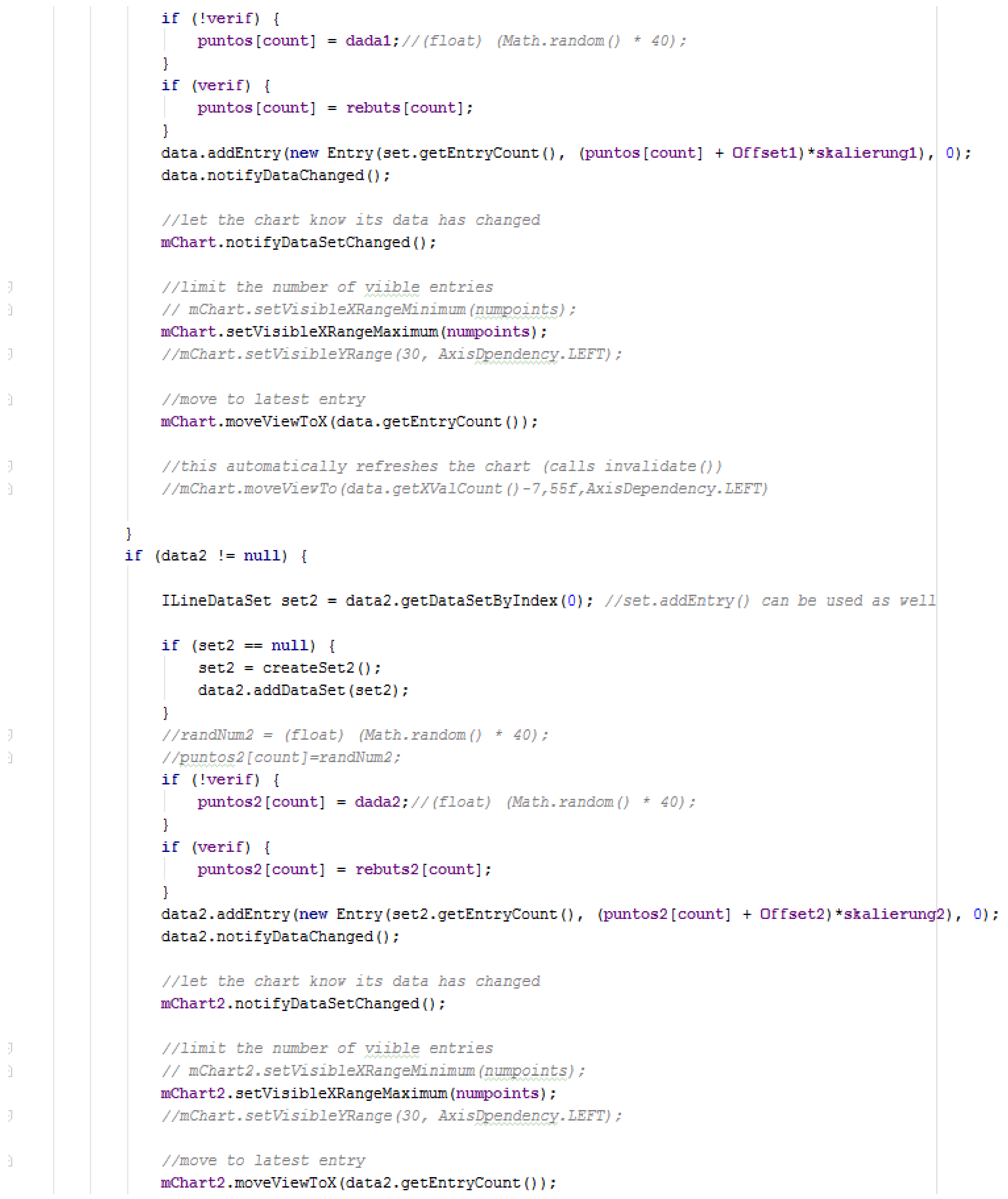


```
private void addEntryplot1() {

    LineData data = mChart.getData();

    if (data != null) {

        ILineDataSet set = data.getDataSetByIndex(0); //set.addEntry() can be used as well

        if (set == null) {
            set = createSet();
            data.addDataSet(set);
        }
        //randNum = (float) (Math.random() * 40);
        // n=n+1;
        //  puntos[n]*=randNum;
        if (!verif) {
            puntos[count] = (float) (Math.random() * 40);
            //puntos[count] = 1.0f;
        }
        if (verif) {
            puntos[count] = rebuts[count];
        }
        data.addEntry(new Entry(set.getEntryCount(), (puntos[count] + Offset1)*skalierung1), 0);
        data.notifyDataChanged();

        //let the chart know its data has changed
        mChart.notifyDataSetChanged();

        //limit the number of visible entries
        mChart.setVisibleXRangeMinimum(timeRoll);
        mChart.setVisibleXRangeMaximum(timeRoll);
        //mChart.setVisibleYRange(30, AxisDependency.LEFT);

        //move to latest entry
        mChart.moveViewToX(data.getEntryCount());

        //this automatically refreshes the chart (calls invalidate())
        //mChart.moveViewTo(data.getXValCount()-7,55f,AxisDependency.LEFT)

    }
}

private LineDataSet createSet() {

    LineDataSet set = new LineDataSet(null, unitat1);
    set.setAxisDependency(YAxis.AxisDependency.LEFT);
    set.setColor(ColorTemplate.getHoloBlue());
    set.setCircleColor(ColorTemplate.getHoloBlue());
    set.setLineWidth(2f);
    set.setCircleRadius(1f);
    set.setFillAlpha(65);
    set.setFillColor(ColorTemplate.getHoloBlue());
    set.setHighLightColor(Color.rgb(244, 117, 117));
    set.setValueTextColor(Color.WHITE);
    set.setValueTextSize(9f);
    set.setDrawValues(false);
```

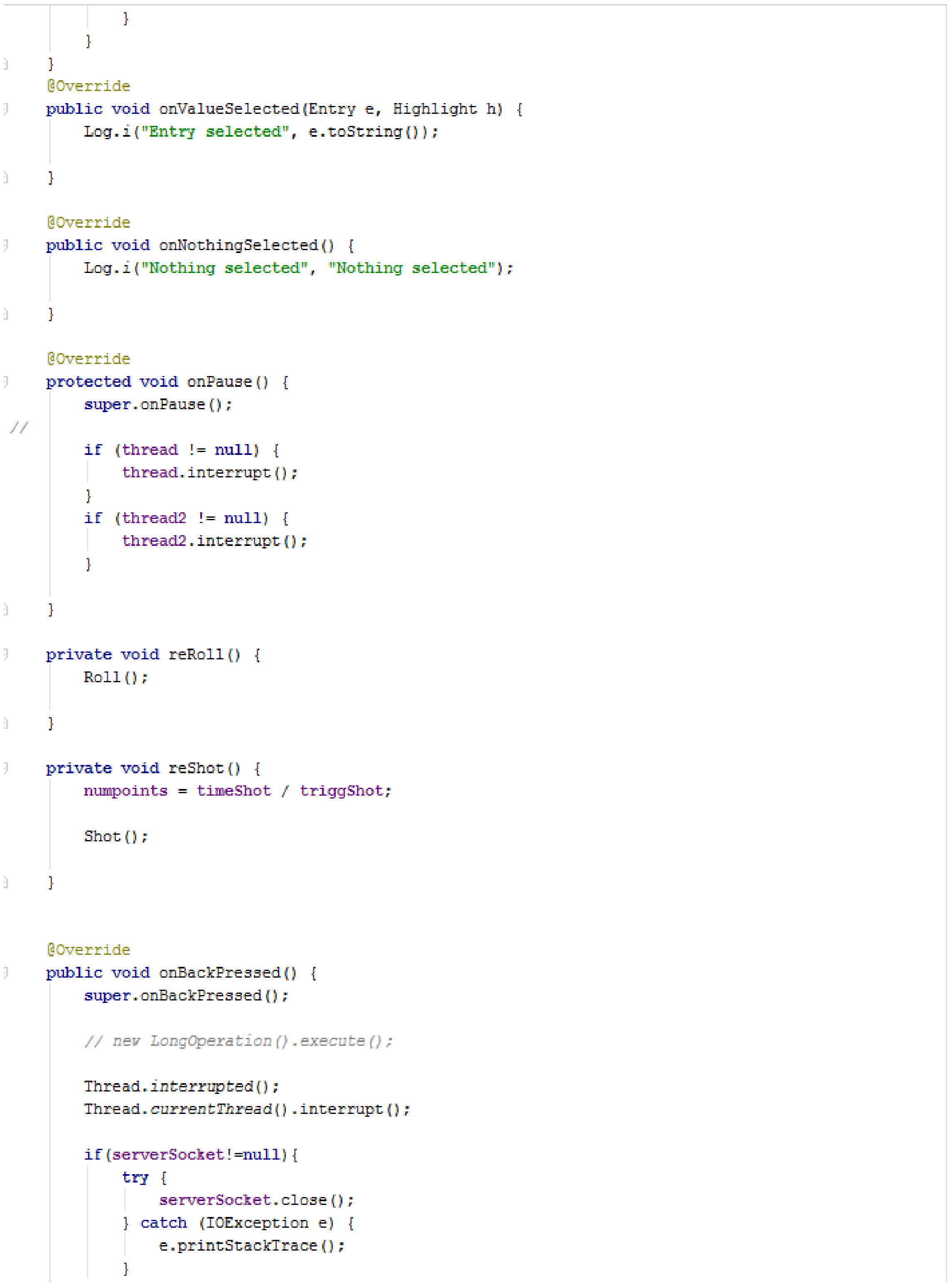


```
    if(!backg){
        mChart.setBackgroundColor(Color.BLACK);
        mChart2.setBackgroundColor(Color.BLACK);
        set.setValueTextColor(Color.WHITE);
    }

    if(backg){
        mChart.setBackgroundColor(Color.WHITE);
        mChart2.setBackgroundColor(Color.WHITE);
        set.setValueTextColor(Color.BLACK);
    }

    return set;

}


private void addEntryplot2() {

    LineData data2 = mChart2.getData();

    if (data2 != null) {

        ILineDataSet set2 = data2.getDataSetByIndex(0); //set.addEntry() can be used as well

        if (set2 == null) {
            set2 = createSet2();
            data2.addDataSet(set2);
        }

        if (!verif) {
            puntos2[count] = (float) (Math.random() * 40);
        }
        if (verif) {
            puntos2[count] = rebuts2[count];
        }
        data2.addEntry(new Entry(set2.getEntryCount(), (puntos2[count] + Offset2)*skalierung2), 0);
        data2.notifyDataChanged();

        //let the chart know its data has changed
        mChart2.notifyDataSetChanged();

        //limit the number of visible entries
        mChart2.setVisibleXRangeMinimum(timeRoll);
        mChart2.setVisibleXRangeMaximum(timeRoll);
        //mChart.setVisibleYRange(30, AxisDependency.LEFT);

        //move to latest entry
        mChart2.moveViewToX(data2.getEntryCount());

        //this automatically refreshes the chart (calls invalidate())
        //mChart.moveViewTo(data.getXValCount()-7,55f,AxisDependency.LEFT)

    }
}
```

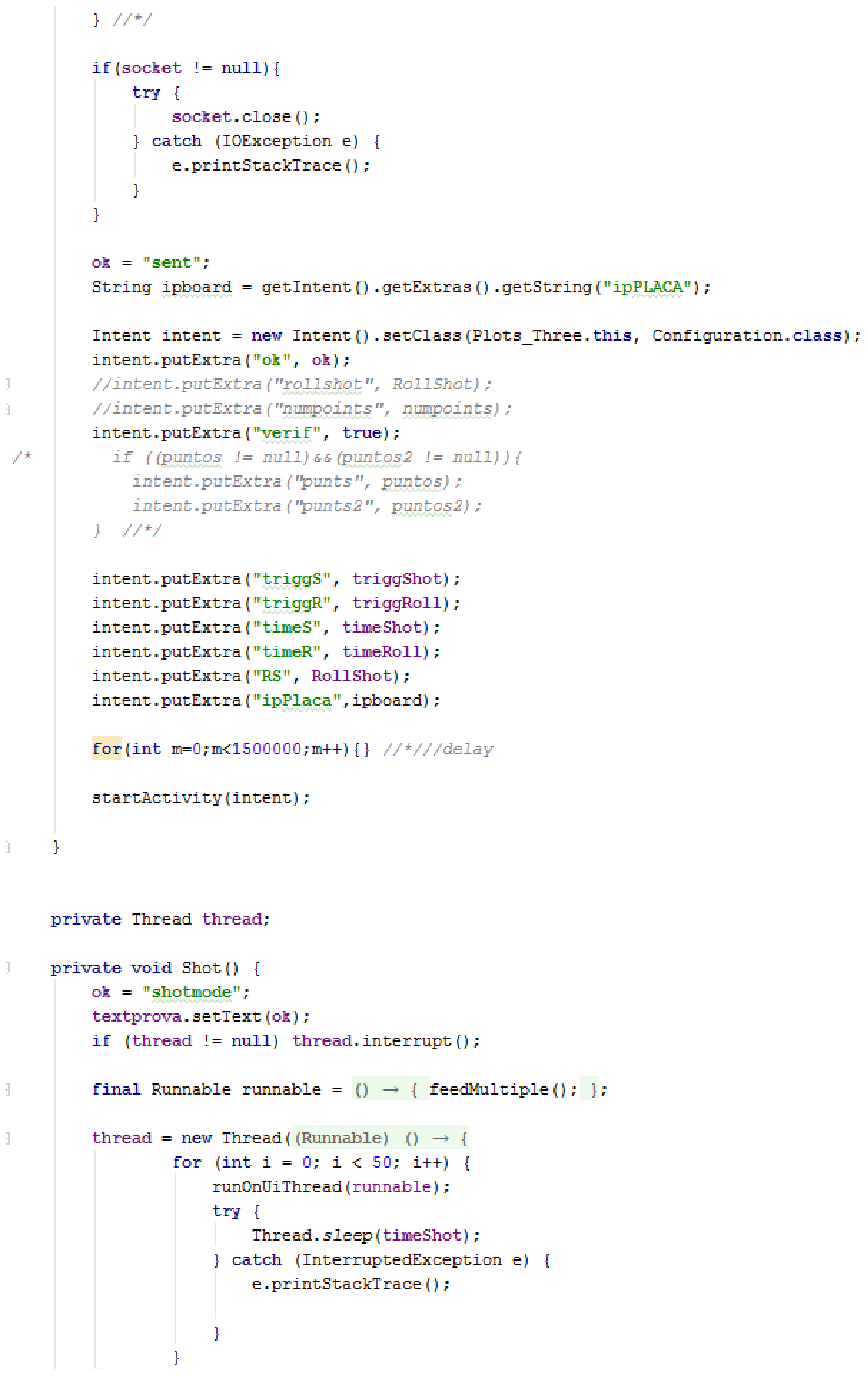
```
private LineDataSet createSet2() {

    LineDataSet set = new LineDataSet(null, unitat2);
    set.setAxisDependency(YAxis.AxisDependency.LEFT);
    set.setColor(Color.RED);
    set.setCircleColor(Color.RED);
    set.setLineWidth(2f);
    set.setCircleRadius(1f);
    set.setFillAlpha(65);
    set.setFillColor(Color.RED);
    set.setHighLightColor(Color.rgb(244, 117, 117));
    set.setValueTextColor(Color.WHITE);
    set.setValueTextSize(9f);
    set.setDrawValues(false);

    if(!backg){
        mChart.setBackgroundColor(Color.BLACK);
        mChart2.setBackgroundColor(Color.BLACK);
        set.setValueTextColor(Color.WHITE);
    }

    if(backg){
        mChart.setBackgroundColor(Color.WHITE);
        mChart2.setBackgroundColor(Color.WHITE);
        set.setValueTextColor(Color.BLACK);
    }

    return set;

}


private void feedMultiple() {
    //newLength1 = getIntent().getExtras().getInt("SIZE2");
    //ok = newLength1;
    //newLength1 = 30;
    textprova.setText("");
    for (count = 0; count < 100; count++) { //timeShot
        //randNum = puntos[count];

        LineData data = mChart.getData();
        LineData data2 = mChart2.getData();

        if (data != null) {

            ILineDataSet set = data.getDataSetByIndex(0); //set.addEntry() can be used as well

            if (set == null) {
                set = createSet();
                data.addDataSet(set);
            }
            //randNum2 = (float) (Math.random() * 40);
            // n=n+1;
            //  puntos[n]*=randNum;
```



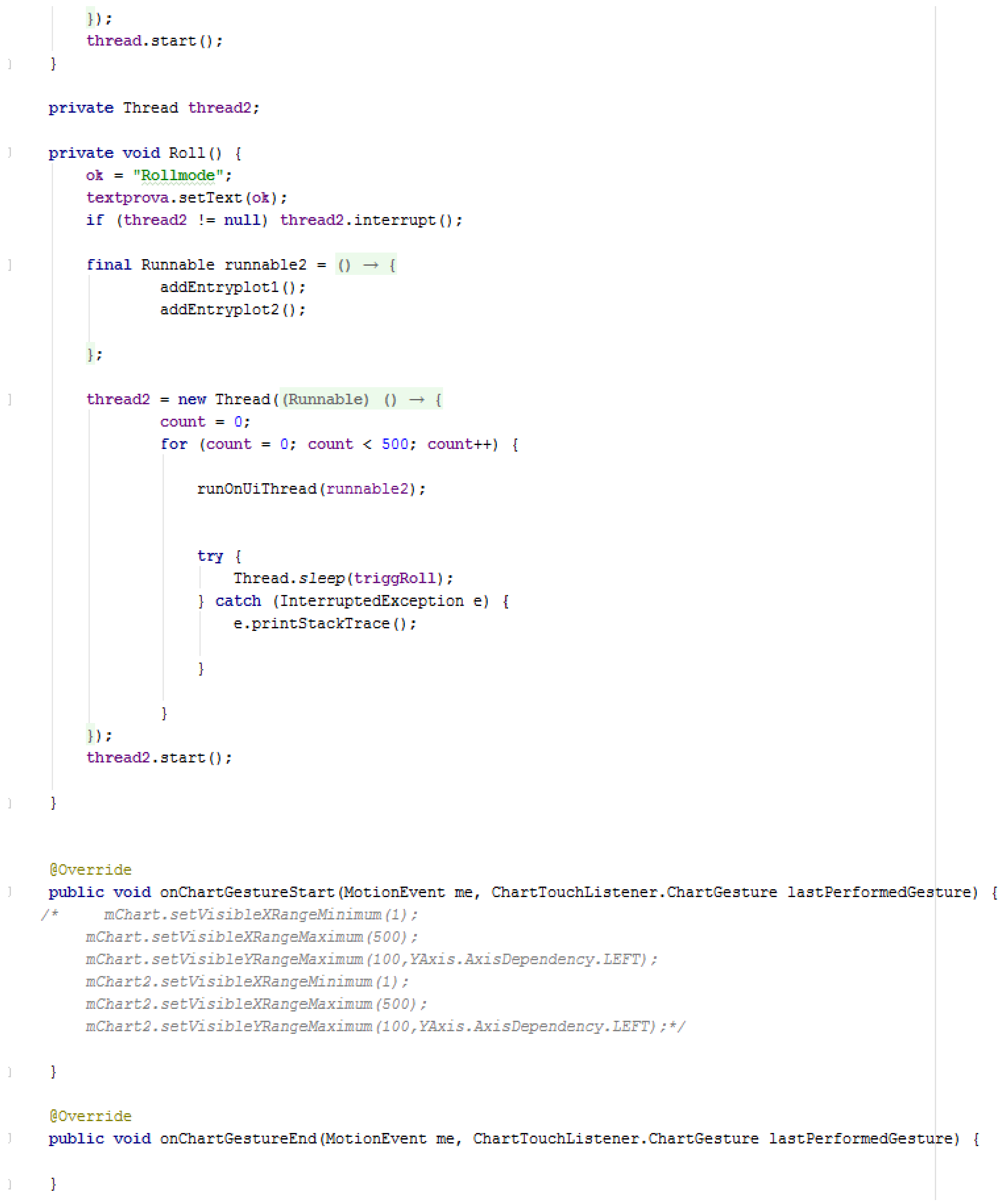
```
                if (!verif) {
                    puntos[count] = dada1;//(float) (Math.random() * 40);
                }
                if (verif) {
                    puntos[count] = rebuts[count];
                }
                data.addEntry(new Entry(set.getEntryCount(), (puntos[count] + Offset1)*skalierung1), 0);
                data.notifyDataChanged();

                //let the chart know its data has changed
                mChart.notifyDataSetChanged();

                //limit the number of viible entries
                // mChart.setVisibleXRangeMinimum(numpoints);
                mChart.setVisibleXRangeMaximum(numpoints);
                //mChart.setVisibleYRange(30, AxisDpendency.LEFT);

                //move to latest entry
                mChart.moveViewToX(data.getEntryCount());

                //this automatically refreshes the chart (calls invalidate())
                //mChart.moveViewTo(data.getXValCount()-7,55f,AxisDependency.LEFT)

            }
            if (data2 != null) {

                ILineDataSet set2 = data2.getDataSetByIndex(0); //set.addEntry() can be used as well

                if (set2 == null) {
                    set2 = createSet2();
                    data2.addDataSet(set2);
                }
                //randNum2 = (float) (Math.random() * 40);
                //puntos2[count]=randNum2;
                if (!verif) {
                    puntos2[count] = dada2;//(float) (Math.random() * 40);
                }
                if (verif) {
                    puntos2[count] = rebuts2[count];
                }
                data2.addEntry(new Entry(set2.getEntryCount(), (puntos2[count] + Offset2)*skalierung2), 0);
                data2.notifyDataChanged();

                //let the chart know its data has changed
                mChart2.notifyDataSetChanged();

                //limit the number of viible entries
                // mChart2.setVisibleXRangeMinimum(numpoints);
                mChart2.setVisibleXRangeMaximum(numpoints);
                //mChart.setVisibleYRange(30, AxisDpendency.LEFT);

                //move to latest entry
                mChart2.moveViewToX(data2.getEntryCount());
```



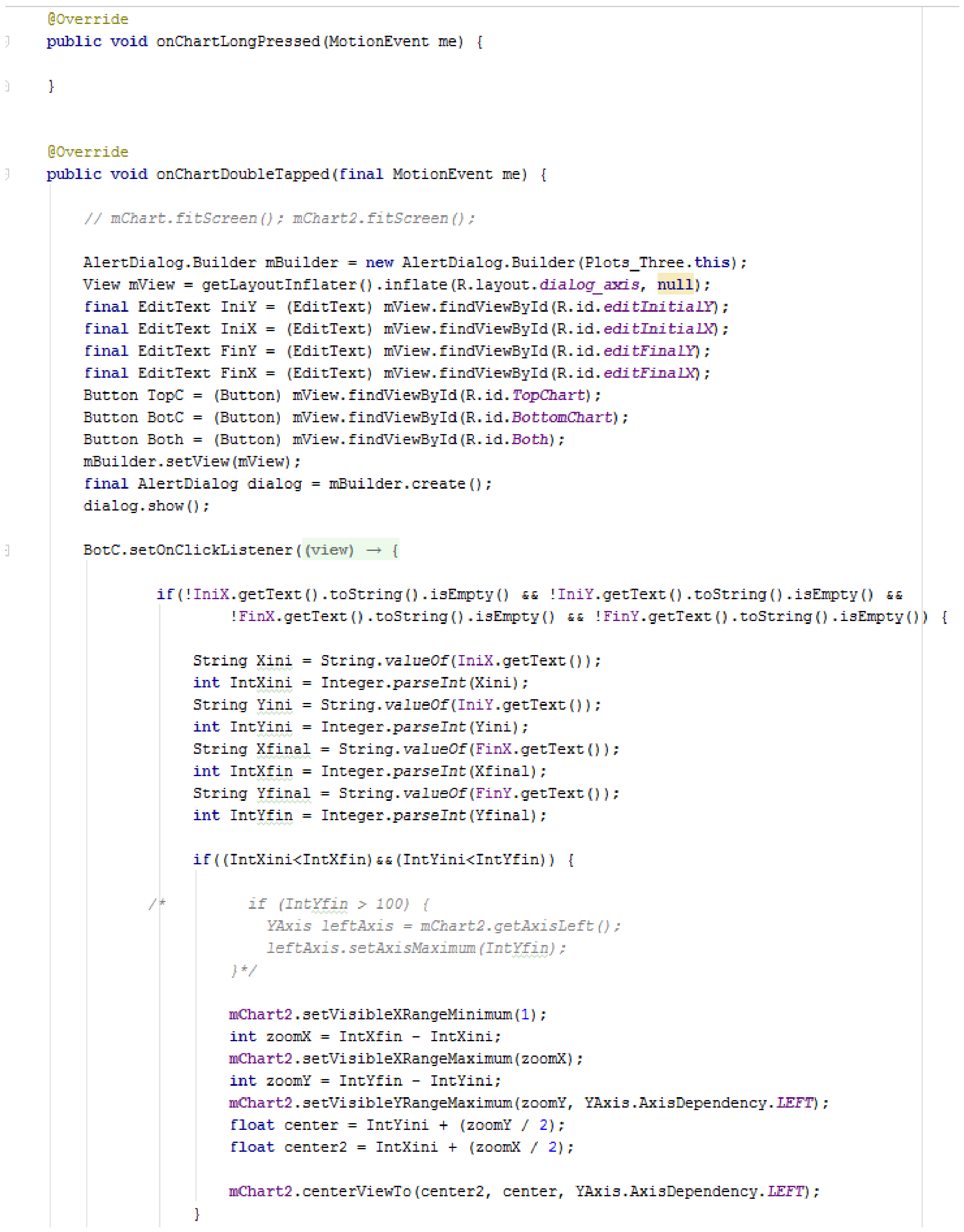
```
            }
        }
    }
    @Override
    public void onValueSelected(Entry e, Highlight h) {
        Log.i("Entry selected", e.toString());

    }

    @Override
    public void onNothingSelected() {
        Log.i("Nothing selected", "Nothing selected");

    }

    @Override
    protected void onPause() {
        super.onPause();
//
        if (thread != null) {
            thread.interrupt();
        }
        if (thread2 != null) {
            thread2.interrupt();
        }

    }

    private void reRoll() {
        Roll();

    }

    private void reShot() {
        numpoints = timeShot / triggShot;

        Shot();

    }


    @Override
    public void onBackPressed() {
        super.onBackPressed();

        // new LongOperation().execute();

        Thread.interrupted();
        Thread.currentThread().interrupt();

        if(serverSocket!=null){
            try {
                serverSocket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
```



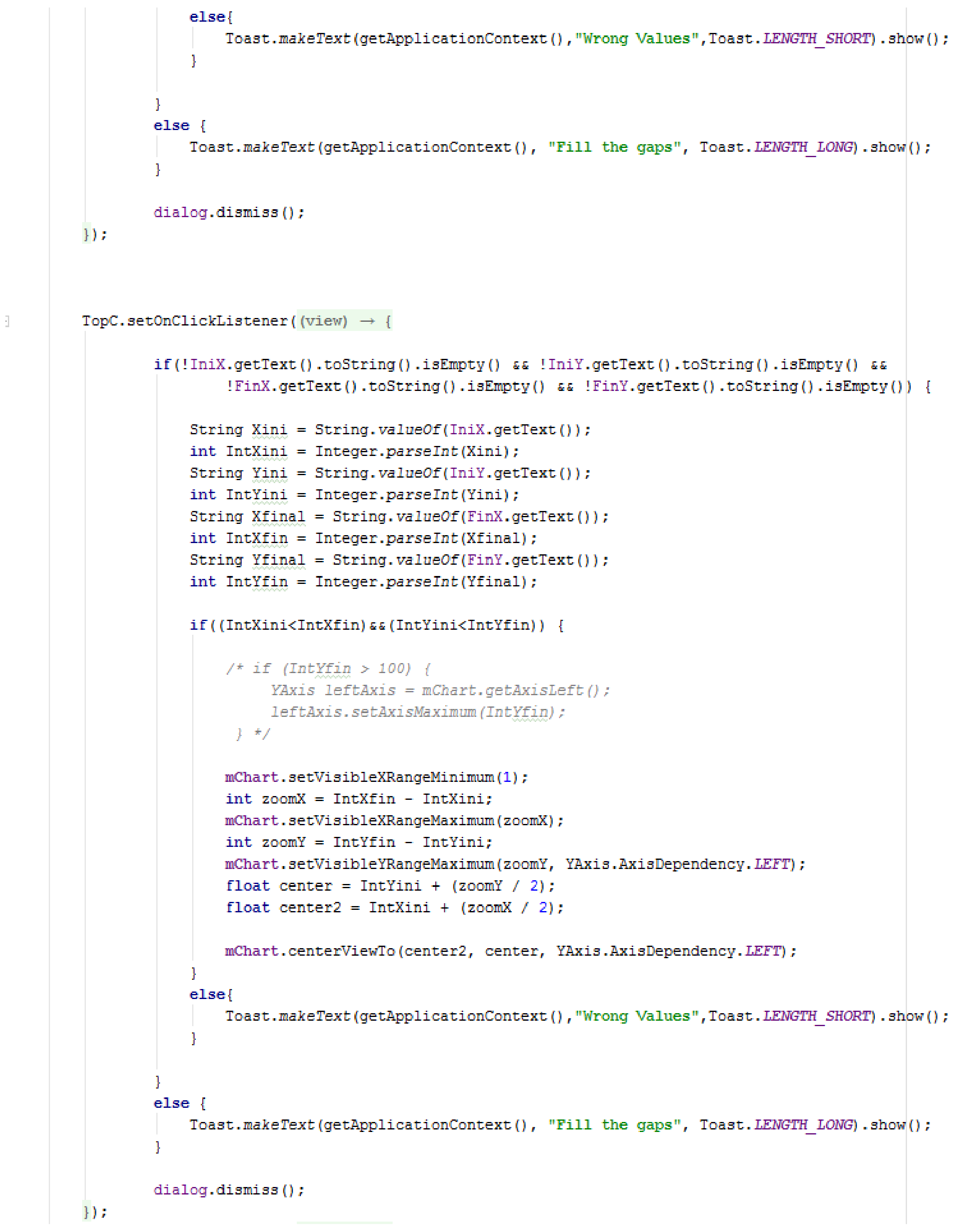
```
        } //*/

        if(socket != null){
            try {
                socket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        ok = "sent";
        String ipboard = getIntent().getExtras().getString("ipPLACA");

        Intent intent = new Intent().setClass(Plots_Three.this, Configuration.class);
        intent.putExtra("ok", ok);
        //intent.putExtra("rollshot", RollShot);
        //intent.putExtra("numpoints", numpoints);
        intent.putExtra("verif", true);
/*        if ((puntos != null)&&(puntos2 != null)){
            intent.putExtra("punts", puntos);
            intent.putExtra("punts2", puntos2);
        }  //*/

        intent.putExtra("triggS", triggShot);
        intent.putExtra("triggR", triggRoll);
        intent.putExtra("timeS", timeShot);
        intent.putExtra("timeR", timeRoll);
        intent.putExtra("RS", RollShot);
        intent.putExtra("ipPlaca",ipboard);

        for(int m=0;m<1500000;m++){} //*///delay

        startActivity(intent);

    }


    private Thread thread;

    private void Shot() {
        ok = "shotmode";
        textprova.setText(ok);
        if (thread != null) thread.interrupt();

        final Runnable runnable = () → { feedMultiple(); };

        thread = new Thread((Runnable) () → {
                for (int i = 0; i < 50; i++) {
                    runOnUiThread(runnable);
                    try {
                        Thread.sleep(timeShot);
                    } catch (InterruptedException e) {
                        e.printStackTrace();

                    }
                }
```



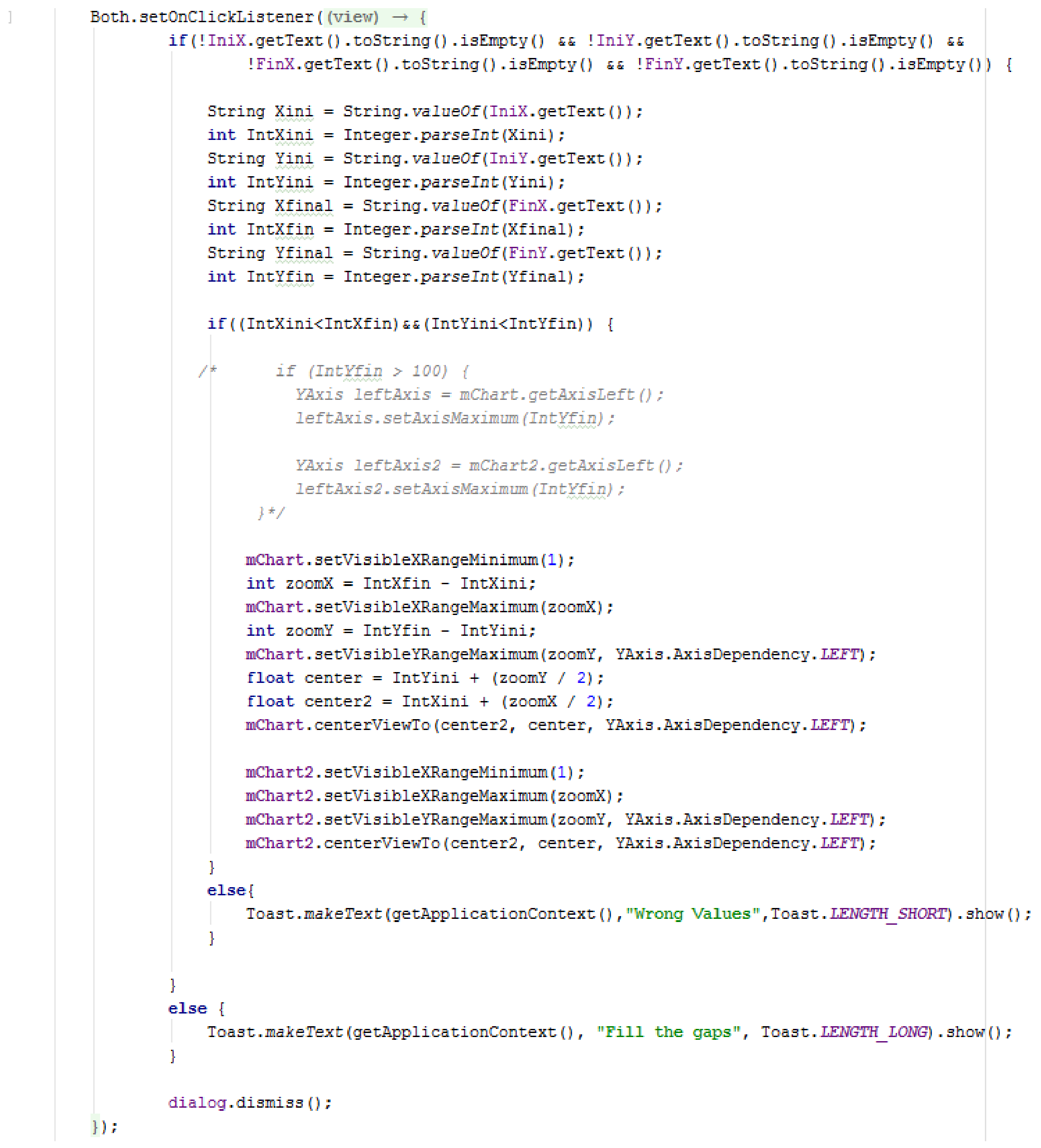
```
        });
        thread.start();
    }

    private Thread thread2;

    private void Roll() {
        ok = "Rollmode";
        textprova.setText(ok);
        if (thread2 != null) thread2.interrupt();

        final Runnable runnable2 = () → {
                addEntryplot1();
                addEntryplot2();

        };

        thread2 = new Thread((Runnable) () → {
                count = 0;
                for (count = 0; count < 500; count++) {

                    runOnUiThread(runnable2);


                    try {
                        Thread.sleep(triggRoll);
                    } catch (InterruptedException e) {
                        e.printStackTrace();

                    }

                }
        });
        thread2.start();

    }


    @Override
    public void onChartGestureStart(MotionEvent me, ChartTouchListener.ChartGesture lastPerformedGesture) {
   /*      mChart.setVisibleXRangeMinimum(1);
        mChart.setVisibleXRangeMaximum(500);
        mChart.setVisibleYRangeMaximum(100,YAxis.AxisDependency.LEFT);
        mChart2.setVisibleXRangeMinimum(1);
        mChart2.setVisibleXRangeMaximum(500);
        mChart2.setVisibleYRangeMaximum(100,YAxis.AxisDependency.LEFT);*/

    }

    @Override
    public void onChartGestureEnd(MotionEvent me, ChartTouchListener.ChartGesture lastPerformedGesture) {

    }
```



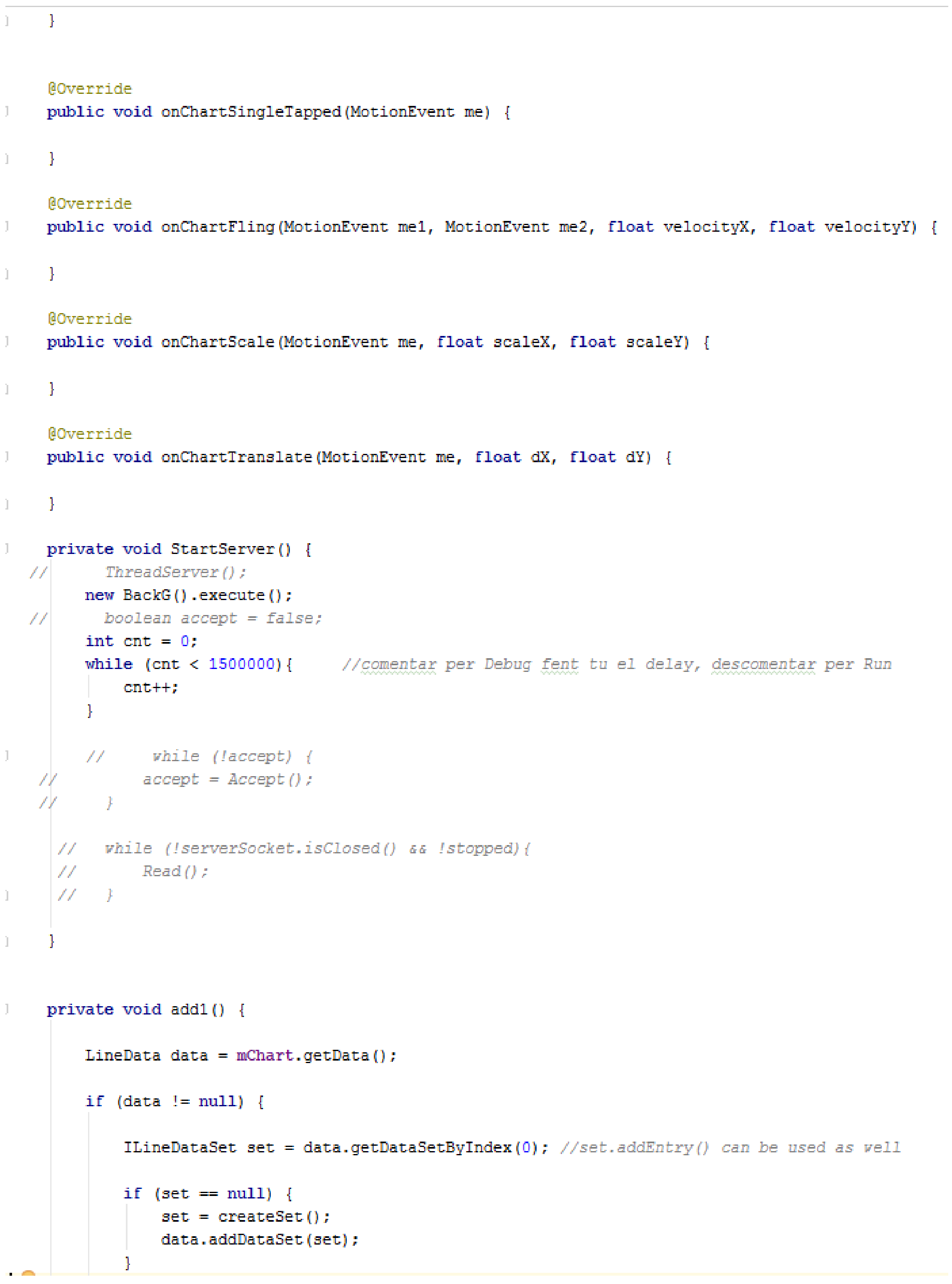
```
@Override
public void onChartLongPressed(MotionEvent me) {

}


@Override
public void onChartDoubleTapped(final MotionEvent me) {

    // mChart.fitScreen(); mChart2.fitScreen();

    AlertDialog.Builder mBuilder = new AlertDialog.Builder(Plots_Three.this);
    View mView = getLayoutInflater().inflate(R.layout.dialog_axis, null);
    final EditText IniY = (EditText) mView.findViewById(R.id.editInitialY);
    final EditText IniX = (EditText) mView.findViewById(R.id.editInitialX);
    final EditText FinY = (EditText) mView.findViewById(R.id.editFinalY);
    final EditText FinX = (EditText) mView.findViewById(R.id.editFinalX);
    Button TopC = (Button) mView.findViewById(R.id.TopChart);
    Button BotC = (Button) mView.findViewById(R.id.BottomChart);
    Button Both = (Button) mView.findViewById(R.id.Both);
    mBuilder.setView(mView);
    final AlertDialog dialog = mBuilder.create();
    dialog.show();

    BotC.setOnClickListener((view) → {

            if(!IniX.getText().toString().isEmpty() && !IniY.getText().toString().isEmpty() &&
                    !FinX.getText().toString().isEmpty() && !FinY.getText().toString().isEmpty()) {

                String Xini = String.valueOf(IniX.getText());
                int IntXini = Integer.parseInt(Xini);
                String Yini = String.valueOf(IniY.getText());
                int IntYini = Integer.parseInt(Yini);
                String Xfinal = String.valueOf(FinX.getText());
                int IntXfin = Integer.parseInt(Xfinal);
                String Yfinal = String.valueOf(FinY.getText());
                int IntYfin = Integer.parseInt(Yfinal);

                if((IntXini<IntXfin)&&(IntYini<IntYfin)) {

           /*          if (IntYfin > 100) {
                        YAxis leftAxis = mChart2.getAxisLeft();
                        leftAxis.setAxisMaximum(IntYfin);
                    }*/

                    mChart2.setVisibleXRangeMinimum(1);
                    int zoomX = IntXfin - IntXini;
                    mChart2.setVisibleXRangeMaximum(zoomX);
                    int zoomY = IntYfin - IntYini;
                    mChart2.setVisibleYRangeMaximum(zoomY, YAxis.AxisDependency.LEFT);
                    float center = IntYini + (zoomY / 2);
                    float center2 = IntXini + (zoomX / 2);

                    mChart2.centerViewTo(center2, center, YAxis.AxisDependency.LEFT);
                }
```



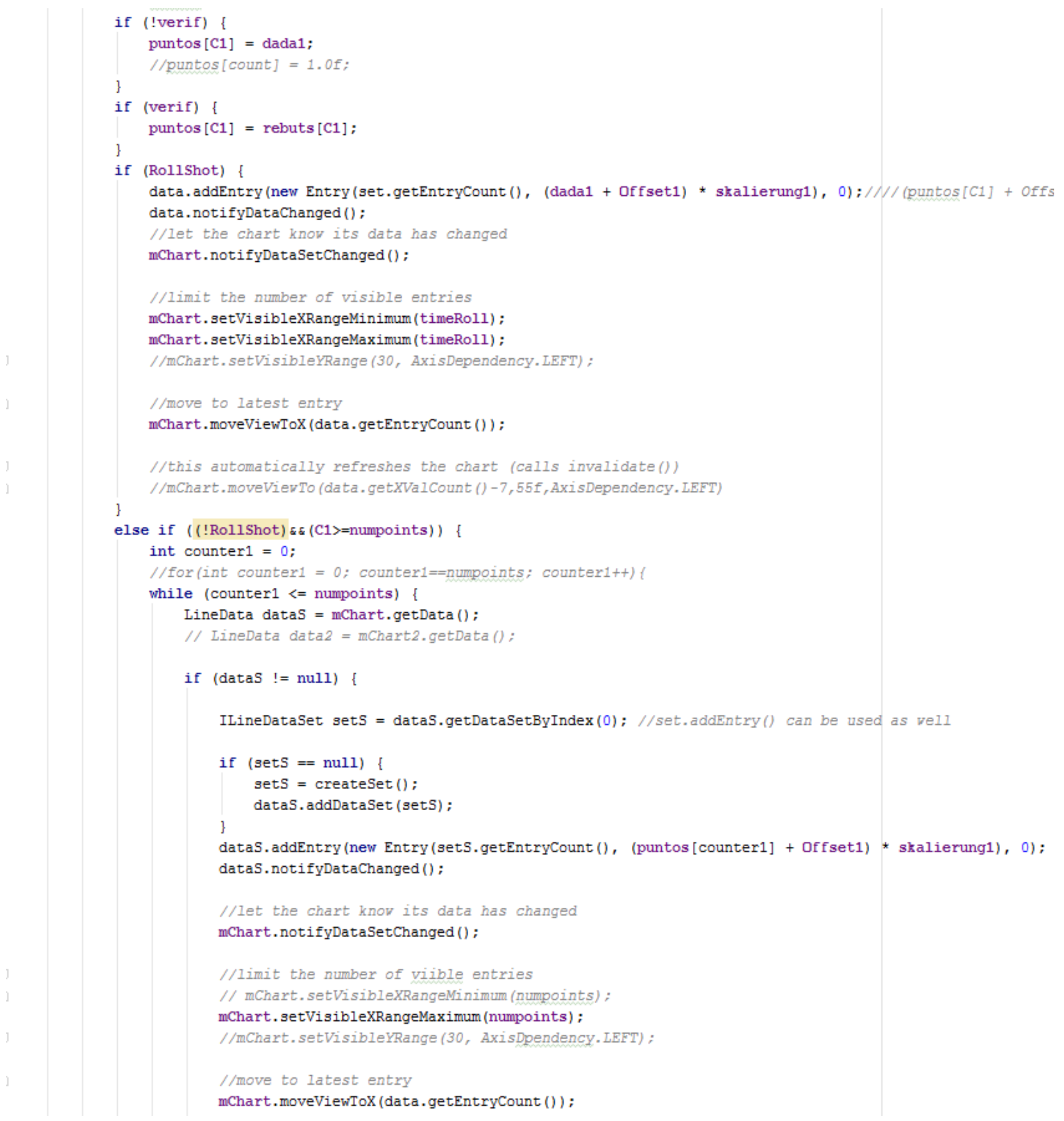
```
                else{
                    Toast.makeText(getApplicationContext(),"Wrong Values",Toast.LENGTH_SHORT).show();
                }

            }
            else {
                Toast.makeText(getApplicationContext(), "Fill the gaps", Toast.LENGTH_LONG).show();
            }

            dialog.dismiss();
    });



    TopC.setOnClickListener((view) → {

            if(!IniX.getText().toString().isEmpty() && !IniY.getText().toString().isEmpty() &&
                    !FinX.getText().toString().isEmpty() && !FinY.getText().toString().isEmpty()) {

                String Xini = String.valueOf(IniX.getText());
                int IntXini = Integer.parseInt(Xini);
                String Yini = String.valueOf(IniY.getText());
                int IntYini = Integer.parseInt(Yini);
                String Xfinal = String.valueOf(FinX.getText());
                int IntXfin = Integer.parseInt(Xfinal);
                String Yfinal = String.valueOf(FinY.getText());
                int IntYfin = Integer.parseInt(Yfinal);

                if((IntXini<IntXfin)&&(IntYini<IntYfin)) {

                    /* if (IntYfin > 100) {
                         YAxis leftAxis = mChart.getAxisLeft();
                         leftAxis.setAxisMaximum(IntYfin);
                     } */

                    mChart.setVisibleXRangeMinimum(1);
                    int zoomX = IntXfin - IntXini;
                    mChart.setVisibleXRangeMaximum(zoomX);
                    int zoomY = IntYfin - IntYini;
                    mChart.setVisibleYRangeMaximum(zoomY, YAxis.AxisDependency.LEFT);
                    float center = IntYini + (zoomY / 2);
                    float center2 = IntXini + (zoomX / 2);

                    mChart.centerViewTo(center2, center, YAxis.AxisDependency.LEFT);
                }
                else{
                    Toast.makeText(getApplicationContext(),"Wrong Values",Toast.LENGTH_SHORT).show();
                }

            }
            else {
                Toast.makeText(getApplicationContext(), "Fill the gaps", Toast.LENGTH_LONG).show();
            }

            dialog.dismiss();
    });
```



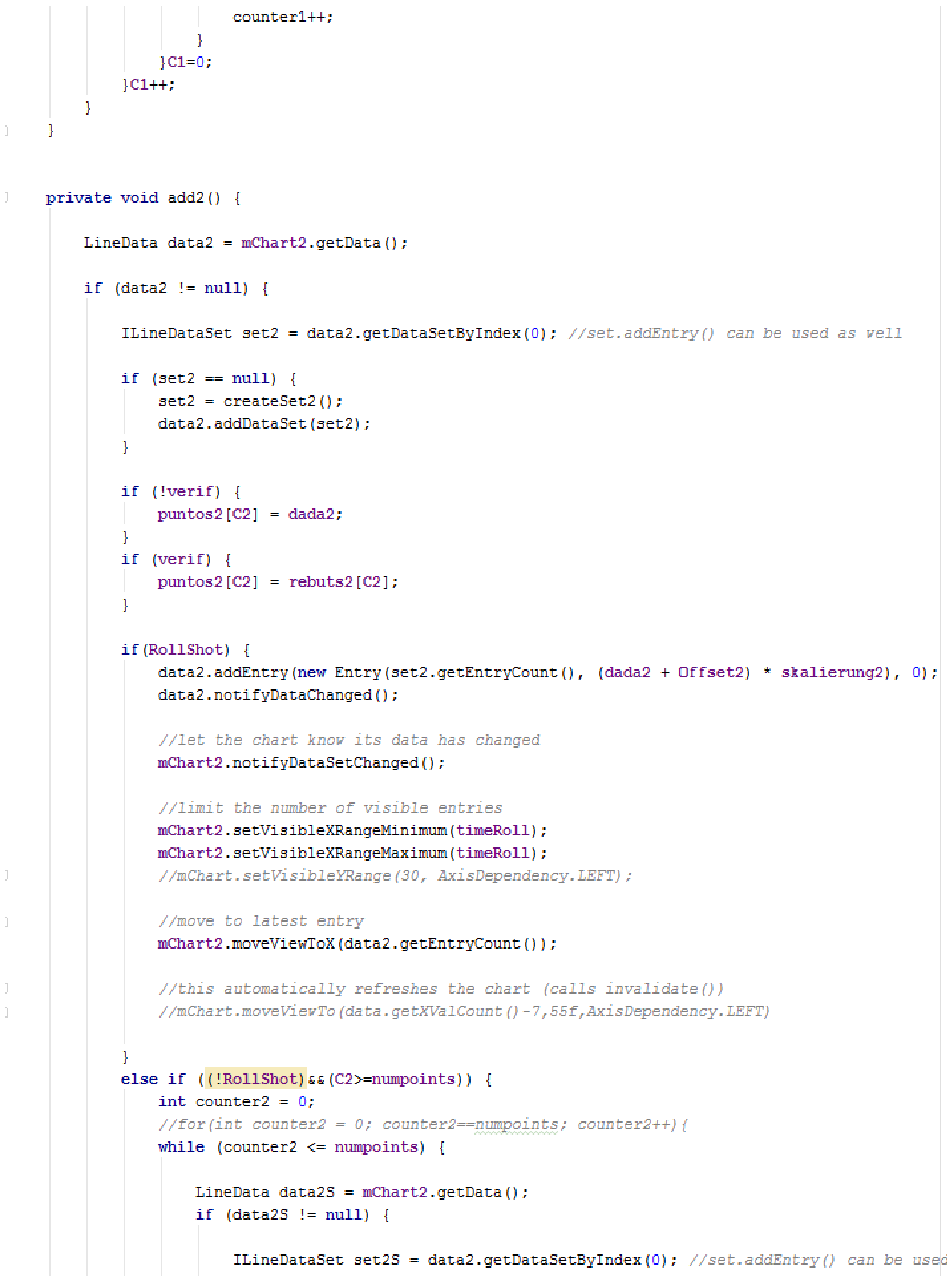
```
Both.setOnClickListener((view) → {
        if(!IniX.getText().toString().isEmpty() && !IniY.getText().toString().isEmpty() &&
                !FinX.getText().toString().isEmpty() && !FinY.getText().toString().isEmpty()) {

            String Xini = String.valueOf(IniX.getText());
            int IntXini = Integer.parseInt(Xini);
            String Yini = String.valueOf(IniY.getText());
            int IntYini = Integer.parseInt(Yini);
            String Xfinal = String.valueOf(FinX.getText());
            int IntXfin = Integer.parseInt(Xfinal);
            String Yfinal = String.valueOf(FinY.getText());
            int IntYfin = Integer.parseInt(Yfinal);

            if((IntXini<IntXfin)&&(IntYini<IntYfin)) {

         /*      if (IntYfin > 100) {
                    YAxis leftAxis = mChart.getAxisLeft();
                    leftAxis.setAxisMaximum(IntYfin);

                    YAxis leftAxis2 = mChart2.getAxisLeft();
                    leftAxis2.setAxisMaximum(IntYfin);
                }*/

                mChart.setVisibleXRangeMinimum(1);
                int zoomX = IntXfin - IntXini;
                mChart.setVisibleXRangeMaximum(zoomX);
                int zoomY = IntYfin - IntYini;
                mChart.setVisibleYRangeMaximum(zoomY, YAxis.AxisDependency.LEFT);
                float center = IntYini + (zoomY / 2);
                float center2 = IntXini + (zoomX / 2);
                mChart.centerViewTo(center2, center, YAxis.AxisDependency.LEFT);

                mChart2.setVisibleXRangeMinimum(1);
                mChart2.setVisibleXRangeMaximum(zoomX);
                mChart2.setVisibleYRangeMaximum(zoomY, YAxis.AxisDependency.LEFT);
                mChart2.centerViewTo(center2, center, YAxis.AxisDependency.LEFT);
            }
            else{
                Toast.makeText(getApplicationContext(),"Wrong Values",Toast.LENGTH_SHORT).show();
            }

        }
        else {
            Toast.makeText(getApplicationContext(), "Fill the gaps", Toast.LENGTH_LONG).show();
        }

        dialog.dismiss();
});
```



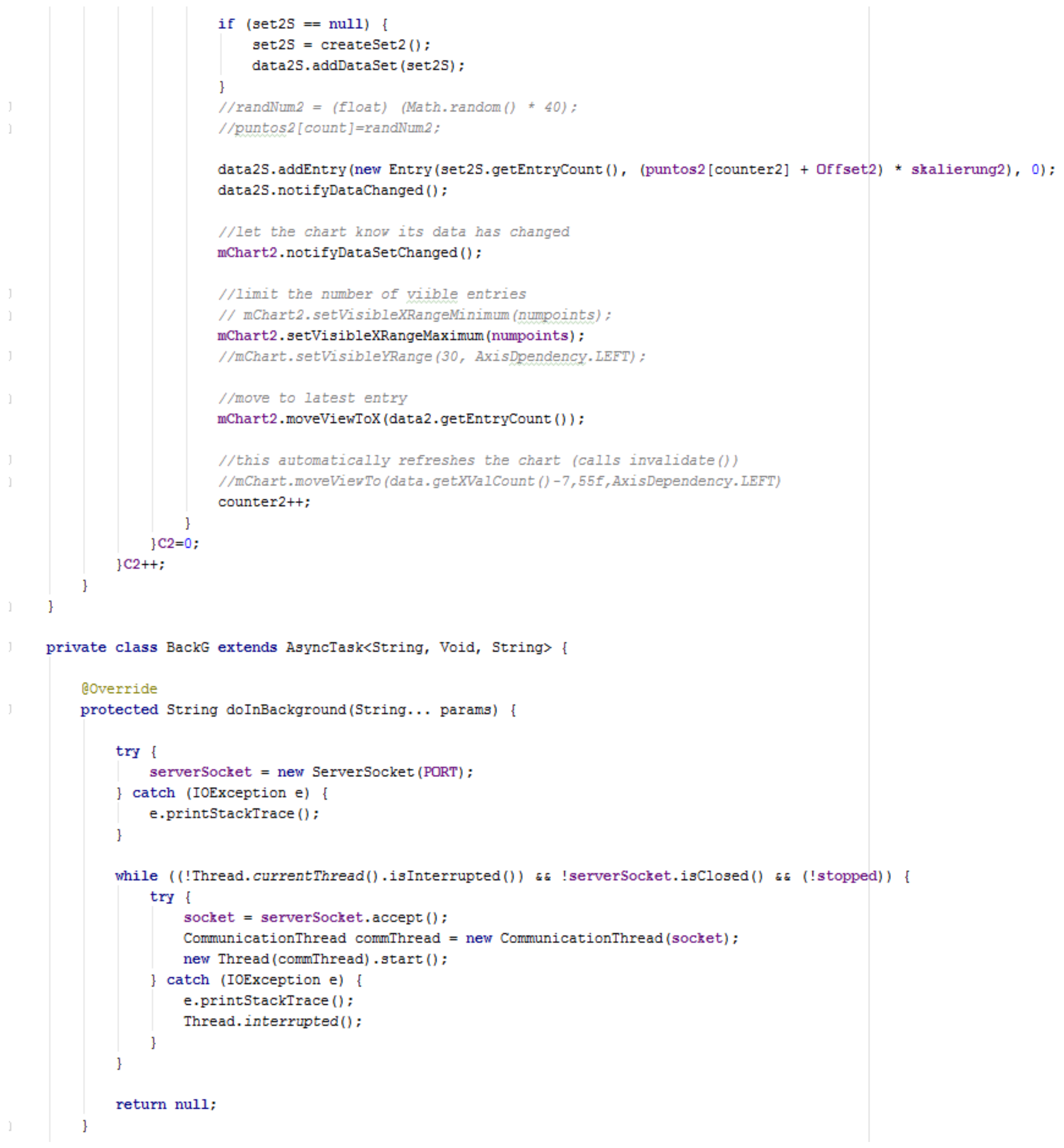
```
    }


    @Override
    public void onChartSingleTapped(MotionEvent me) {

    }

    @Override
    public void onChartFling(MotionEvent me1, MotionEvent me2, float velocityX, float velocityY) {

    }

    @Override
    public void onChartScale(MotionEvent me, float scaleX, float scaleY) {

    }

    @Override
    public void onChartTranslate(MotionEvent me, float dX, float dY) {

    }

    private void StartServer() {
//        ThreadServer();
        new BackG().execute();
//        boolean accept = false;
        int cnt = 0;
        while (cnt < 1500000){      //comentar per Debug fent tu el delay, descomentar per Run
            cnt++;
        }

        //      while (!accept) {
//            accept = Accept();
//        }

    //   while (!serverSocket.isClosed() && !stopped){
    //        Read();
    //    }

    }


    private void add1() {

        LineData data = mChart.getData();

        if (data != null) {

            ILineDataSet set = data.getDataSetByIndex(0); //set.addEntry() can be used as well

            if (set == null) {
                set = createSet();
                data.addDataSet(set);
            }
```



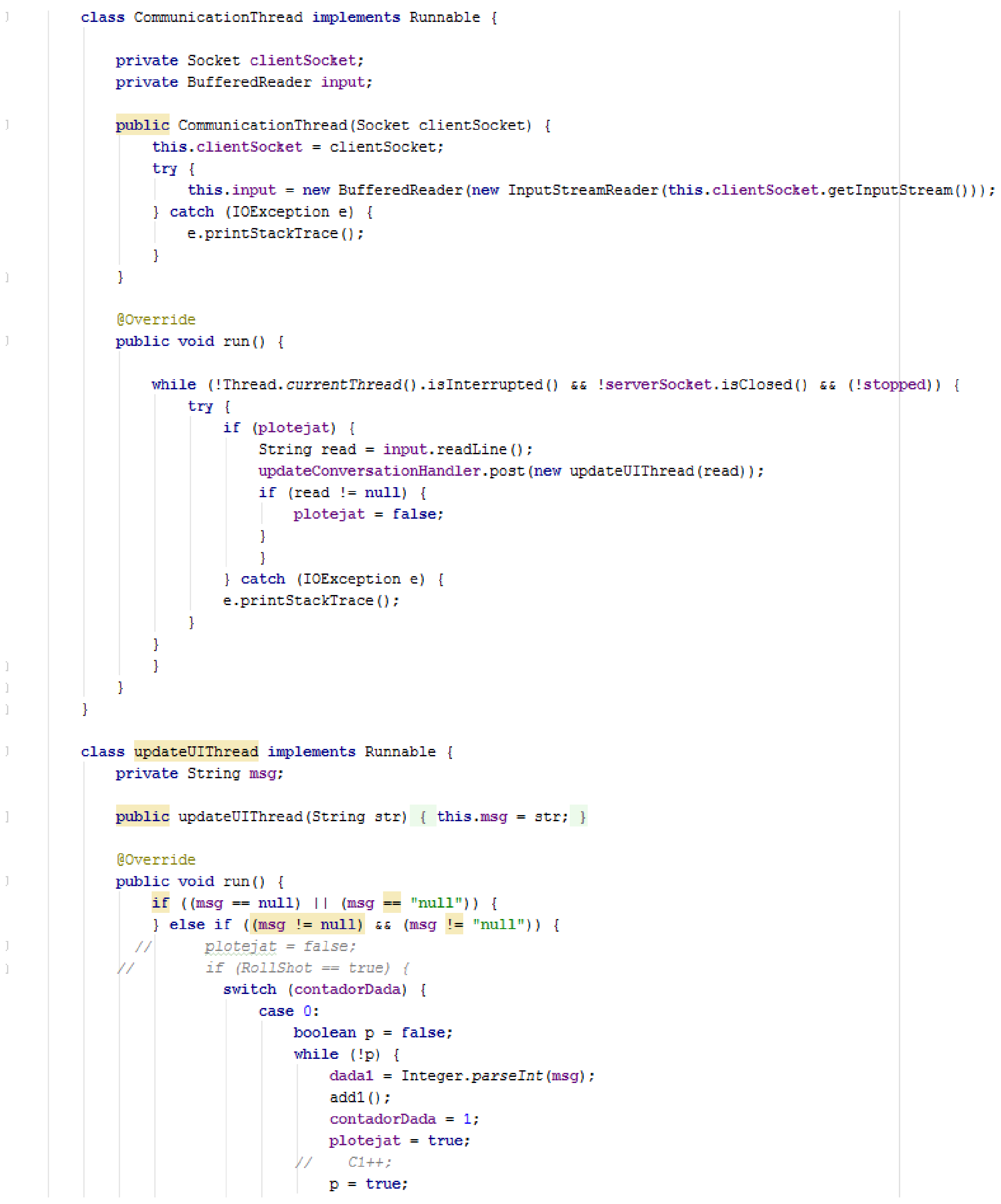
```
if (!verif) {
    puntos[C1] = dada1;
    //puntos[count] = 1.0f;
}
if (verif) {
    puntos[C1] = rebuts[C1];
}
if (RollShot) {
    data.addEntry(new Entry(set.getEntryCount(), (dada1 + Offset1) * skalierung1), 0);////(puntos[C1] + Offs
    data.notifyDataChanged();
    //let the chart know its data has changed
    mChart.notifyDataSetChanged();

    //limit the number of visible entries
    mChart.setVisibleXRangeMinimum(timeRoll);
    mChart.setVisibleXRangeMaximum(timeRoll);
    //mChart.setVisibleYRange(30, AxisDependency.LEFT);

    //move to latest entry
    mChart.moveViewToX(data.getEntryCount());

    //this automatically refreshes the chart (calls invalidate())
    //mChart.moveViewTo(data.getXValCount()-7,55f,AxisDependency.LEFT)
}
else if ((!RollShot)&&(C1>=numpoints)) {
    int counter1 = 0;
    //for(int counter1 = 0; counter1==numpoints; counter1++){
    while (counter1 <= numpoints) {
        LineData dataS = mChart.getData();
        // LineData data2 = mChart2.getData();

        if (dataS != null) {

            ILineDataSet setS = dataS.getDataSetByIndex(0); //set.addEntry() can be used as well

            if (setS == null) {
                setS = createSet();
                dataS.addDataSet(setS);
            }
            dataS.addEntry(new Entry(setS.getEntryCount(), (puntos[counter1] + Offset1) * skalierung1), 0);
            dataS.notifyDataChanged();

            //let the chart know its data has changed
            mChart.notifyDataSetChanged();

            //limit the number of viible entries
            // mChart.setVisibleXRangeMinimum(numpoints);
            mChart.setVisibleXRangeMaximum(numpoints);
            //mChart.setVisibleYRange(30, AxisDpendency.LEFT);

            //move to latest entry
            mChart.moveViewToX(data.getEntryCount());
```



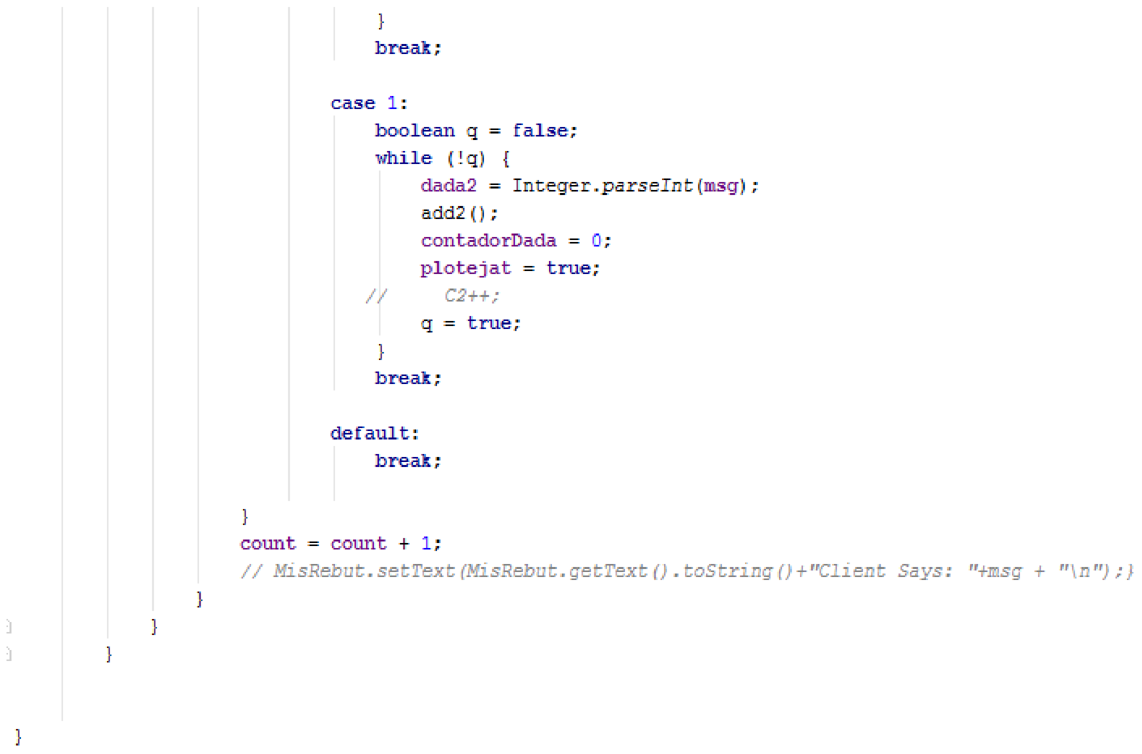
```
                    counter1++;
                }
            }C1=0;
        }C1++;
    }
}


private void add2() {

    LineData data2 = mChart2.getData();

    if (data2 != null) {

        ILineDataSet set2 = data2.getDataSetByIndex(0); //set.addEntry() can be used as well

        if (set2 == null) {
            set2 = createSet2();
            data2.addDataSet(set2);
        }

        if (!verif) {
            puntos2[C2] = dada2;
        }
        if (verif) {
            puntos2[C2] = rebuts2[C2];
        }

        if(RollShot) {
            data2.addEntry(new Entry(set2.getEntryCount(), (dada2 + Offset2) * skalierung2), 0);
            data2.notifyDataChanged();

            //let the chart know its data has changed
            mChart2.notifyDataSetChanged();

            //limit the number of visible entries
            mChart2.setVisibleXRangeMinimum(timeRoll);
            mChart2.setVisibleXRangeMaximum(timeRoll);
            //mChart.setVisibleYRange(30, AxisDependency.LEFT);

            //move to latest entry
            mChart2.moveViewToX(data2.getEntryCount());

            //this automatically refreshes the chart (calls invalidate())
            //mChart.moveViewTo(data.getXValCount()-7,55f,AxisDependency.LEFT)

        }
        else if ((!RollShot)&&(C2>=numpoints)) {
            int counter2 = 0;
            //for(int counter2 = 0; counter2==numpoints; counter2++){
            while (counter2 <= numpoints) {

                LineData data2S = mChart2.getData();
                if (data2S != null) {

                    ILineDataSet set2S = data2.getDataSetByIndex(0); //set.addEntry() can be used
```



```
                    if (set2S == null) {
                        set2S = createSet2();
                        data2S.addDataSet(set2S);
                    }
                    //randNum2 = (float) (Math.random() * 40);
                    //puntos2[count]=randNum2;

                    data2S.addEntry(new Entry(set2S.getEntryCount(), (puntos2[counter2] + Offset2) * skalierung2), 0);
                    data2S.notifyDataChanged();

                    //let the chart know its data has changed
                    mChart2.notifyDataSetChanged();

                    //limit the number of viible entries
                    // mChart2.setVisibleXRangeMinimum(numpoints);
                    mChart2.setVisibleXRangeMaximum(numpoints);
                    //mChart.setVisibleYRange(30, AxisDpendency.LEFT);

                    //move to latest entry
                    mChart2.moveViewToX(data2.getEntryCount());

                    //this automatically refreshes the chart (calls invalidate())
                    //mChart.moveViewTo(data.getXValCount()-7,55f,AxisDependency.LEFT)
                    counter2++;
                }
            }C2=0;
        }C2++;
    }
}

private class BackG extends AsyncTask<String, Void, String> {

    @Override
    protected String doInBackground(String... params) {

        try {
            serverSocket = new ServerSocket(PORT);
        } catch (IOException e) {
            e.printStackTrace();
        }

        while ((!Thread.currentThread().isInterrupted()) && !serverSocket.isClosed() && (!stopped)) {
            try {
                socket = serverSocket.accept();
                CommunicationThread commThread = new CommunicationThread(socket);
                new Thread(commThread).start();
            } catch (IOException e) {
                e.printStackTrace();
                Thread.interrupted();
            }
        }

        return null;
    }
```



```java
class CommunicationThread implements Runnable {

    private Socket clientSocket;
    private BufferedReader input;

    public CommunicationThread(Socket clientSocket) {
        this.clientSocket = clientSocket;
        try {
            this.input = new BufferedReader(new InputStreamReader(this.clientSocket.getInputStream()));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void run() {

        while (!Thread.currentThread().isInterrupted() && !serverSocket.isClosed() && (!stopped)) {
            try {
                if (plotejat) {
                    String read = input.readLine();
                    updateConversationHandler.post(new updateUIThread(read));
                    if (read != null) {
                        plotejat = false;
                    }
                }
            } catch (IOException e) {
            e.printStackTrace();
            }
        }
        }
    }
}

class updateUIThread implements Runnable {
    private String msg;

    public updateUIThread(String str) { this.msg = str; }

    @Override
    public void run() {
        if ((msg == null) || (msg == "null")) {
        } else if ((msg != null) && (msg != "null")) {
 //         plotejat = false;
//          if (RollShot == true) {
                switch (contadorDada) {
                    case 0:
                        boolean p = false;
                        while (!p) {
                            dada1 = Integer.parseInt(msg);
                            add1();
                            contadorDada = 1;
                            plotejat = true;
                        //    C1++;
                            p = true;
```



```
                                }
                                break;

                            case 1:
                                boolean q = false;
                                while (!q) {
                                    dada2 = Integer.parseInt(msg);
                                    add2();
                                    contadorDada = 0;
                                    plotejat = true;
                               //      C2++;
                                    q = true;
                                }
                                break;

                            default:
                                break;

                    }
                    count = count + 1;
                    // MisRebut.setText(MisRebut.getText().toString()+"Client Says: "+msg + "\n");}
                }
            }
        }

}
```