2013

Networked Mobile Devices

Android & Arduino

Raúl Marín

Lab 1: Android Studio Introduction

In this Lab session the Android Studio Platform is introduced and a simple application is programmed in order to get an overall description of the tool.

Author: Raúl Marín

Installing Android Studio

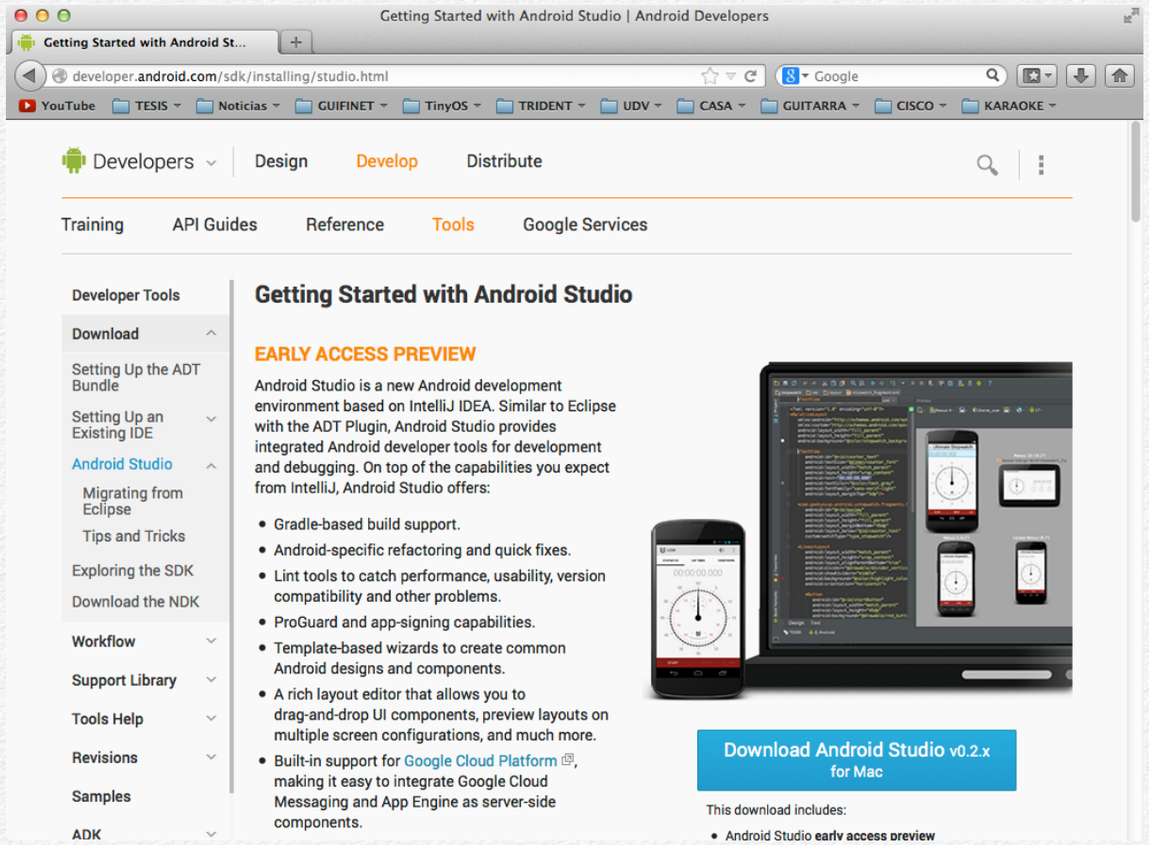
Summary

1. In this section we are going to install the Android Studio platform. This tool can run on Windows, MAC, and Linux, and it is based on the JDK Java Development Kit, which must be previously installed in the system.
2. The Android Studio platform, at the moment of writing, is still in beta version, so it is possible to find some tricks and tips that are needed to get the system properly ready for programming.
3. For example, the use of the JAVA_HOME variable is important to let the Android Studio platform find the correspondent JDK Java machine.

STEP 1

Visit the following Android Studio installation webpage:

http://developer.android.com/sdk/installing/studio.html



STEP 2

Select the operating system you are using and follow the instructions accordingly, as specified in the webpage.

Creating a Basic Application

Summary

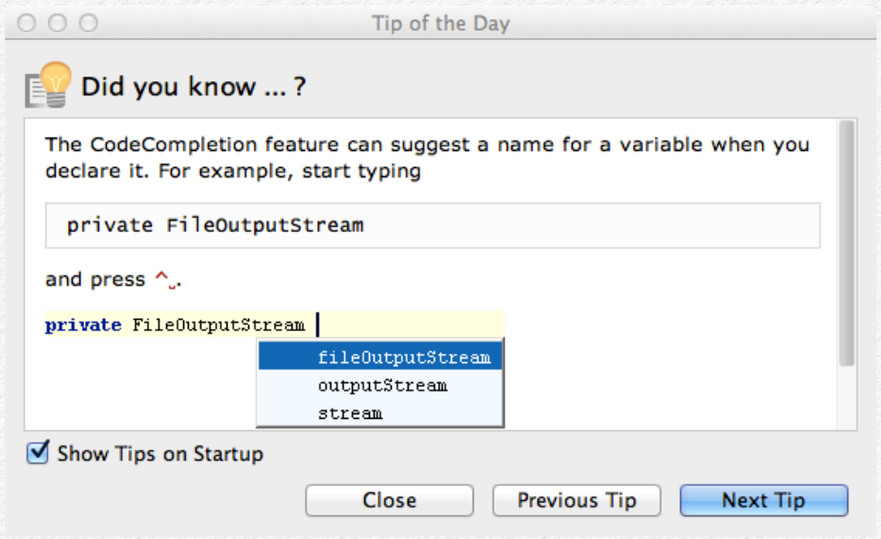
1. **In this section we are going to explain how to create a basic Hello World Application**
2. **The program will be run in both, the Android Simulator and a real Nexus 7 tablet.**

STEP 1

Launch the Android Studio Platform and wait for the sytem to come up.

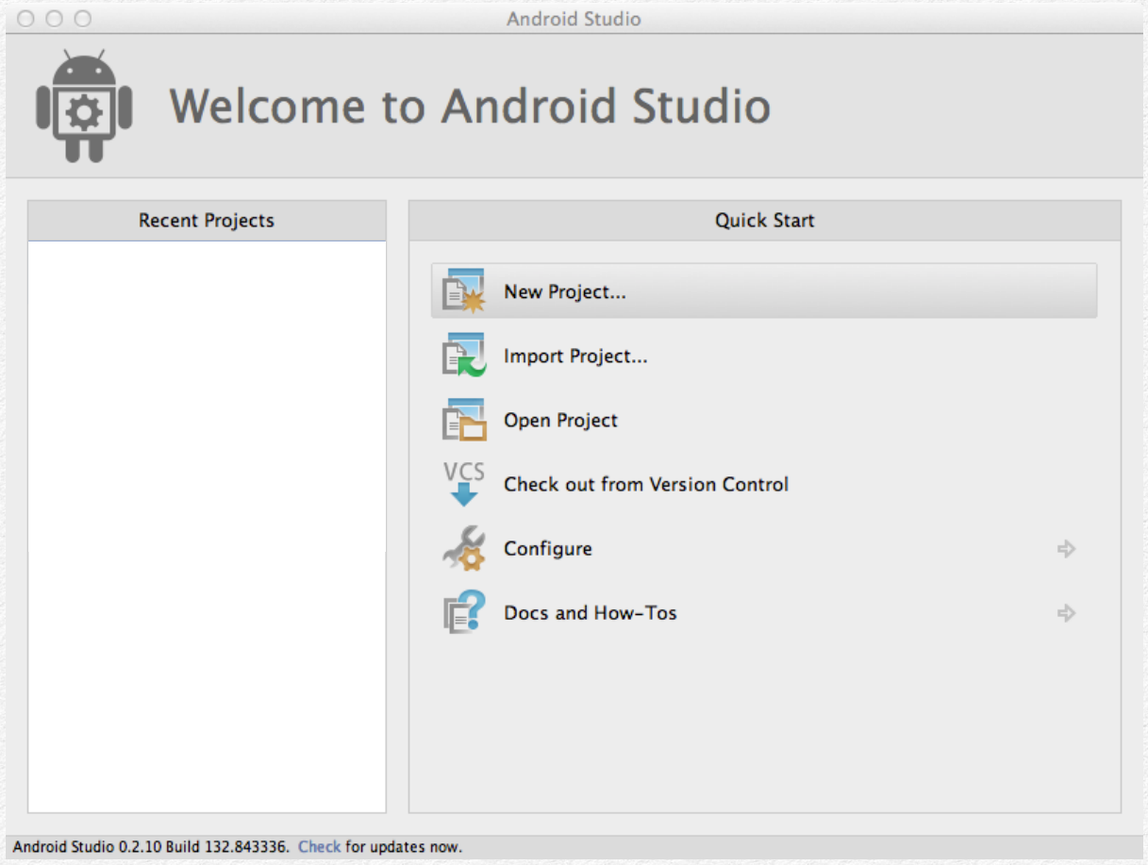
STEP 2

It is interesting to have a look to the tips, to get habituated to the platform more easily.



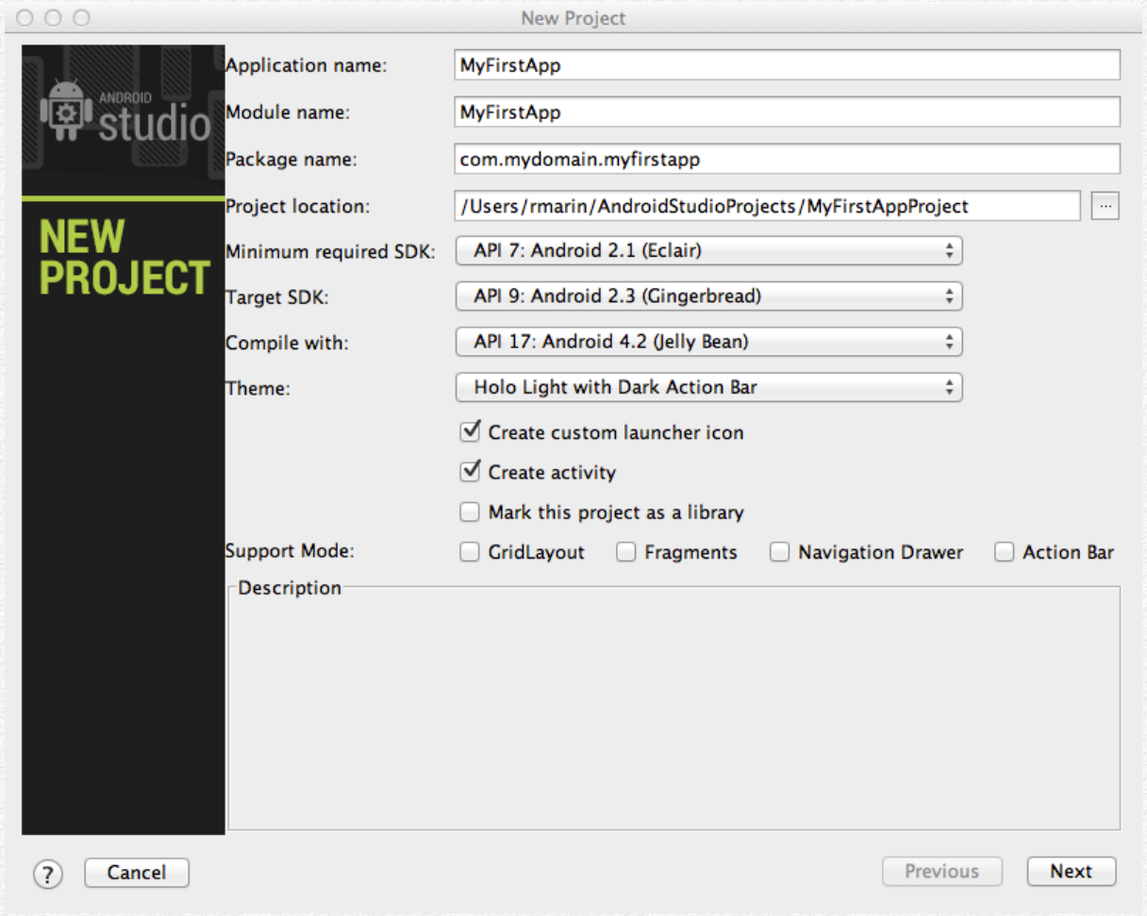
STEP 3

Select the option “New Project”.



STEP 4

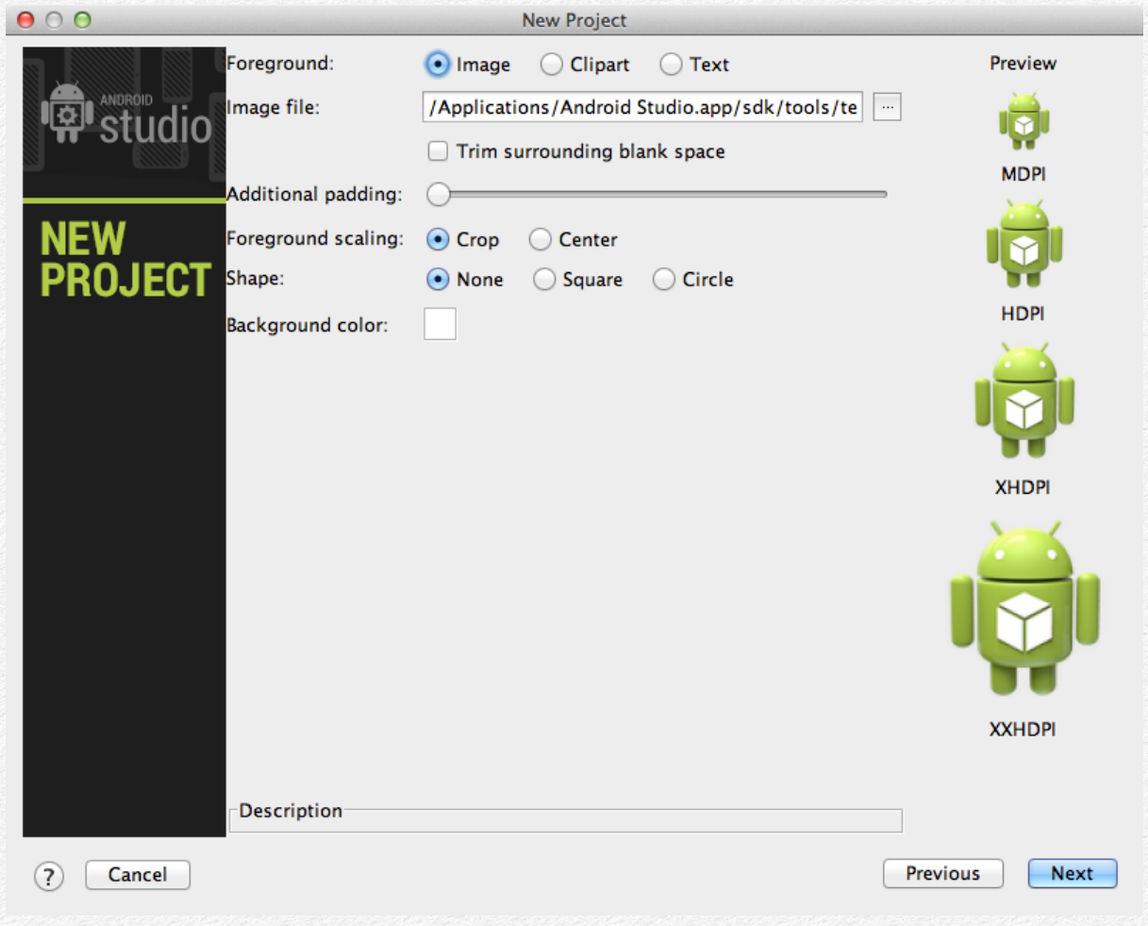
In the following window we are going to provide the “Application Name” as “MyFirstApp”, Select the option “New Project”, the “Package Name” as “com.mydomain.myfirstapp”, the “Minimum required SDK” as “API 10: Android 2.3.3 (Gingerbread)”, and select the project location accordingly.



Finally, click the “Next” button.

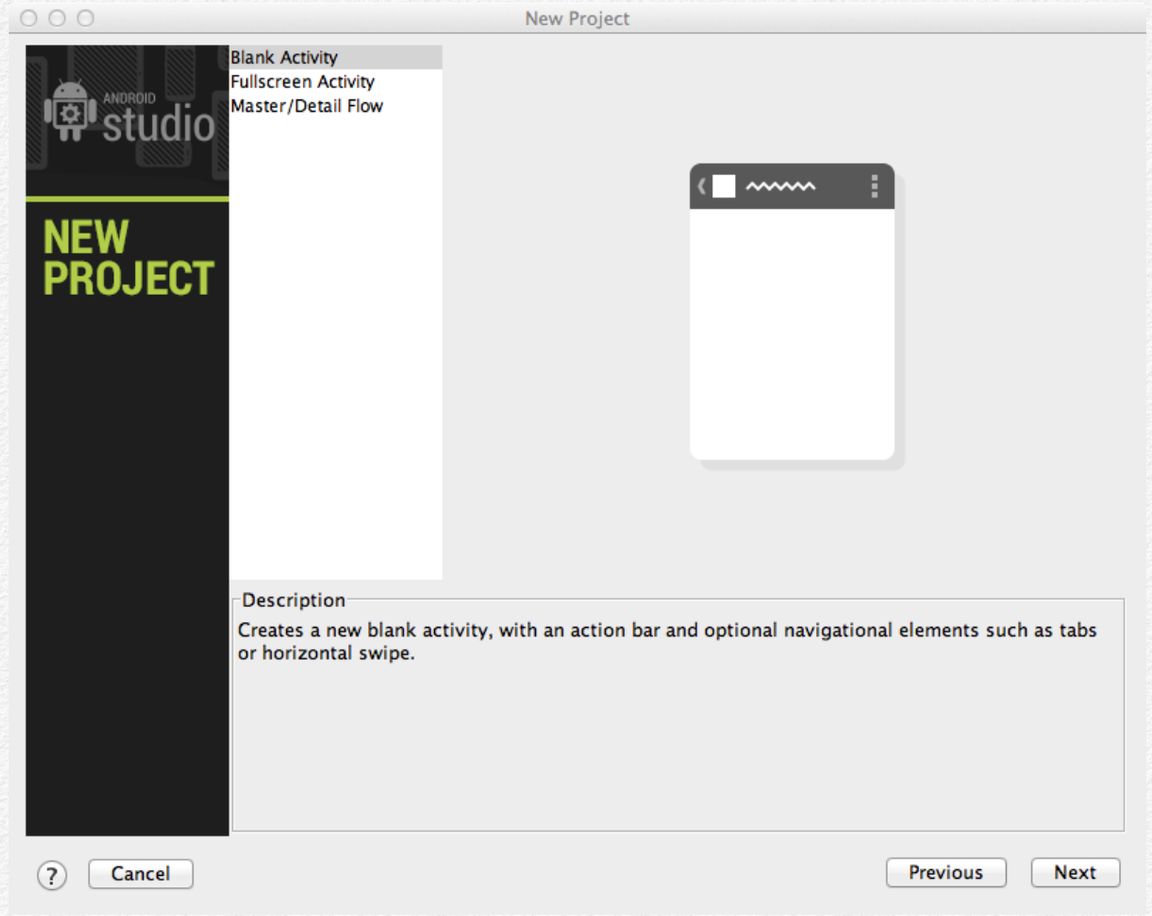
STEP 5

Select the application icon. This can be left as the default.



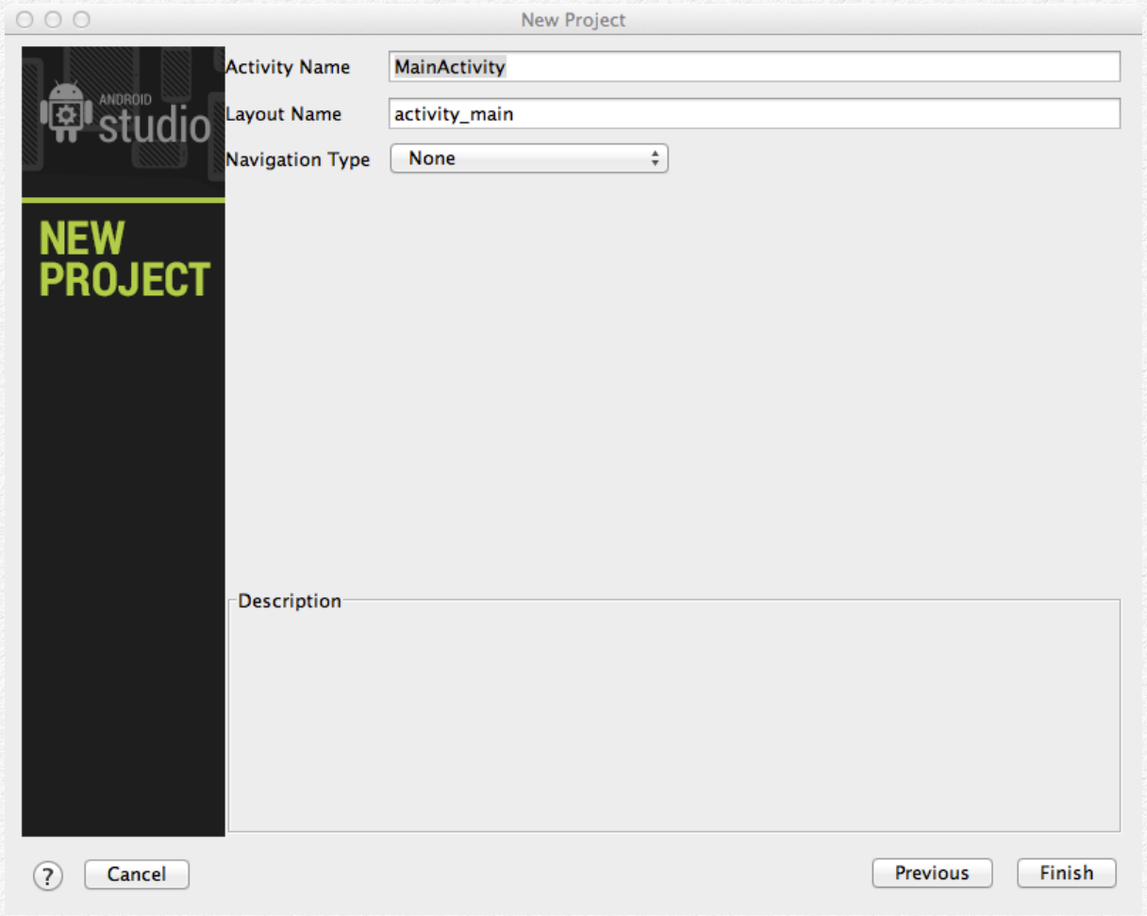
STEP 6

Select the “Blank Activity” option, and then press the “Next” button.



STEP 7

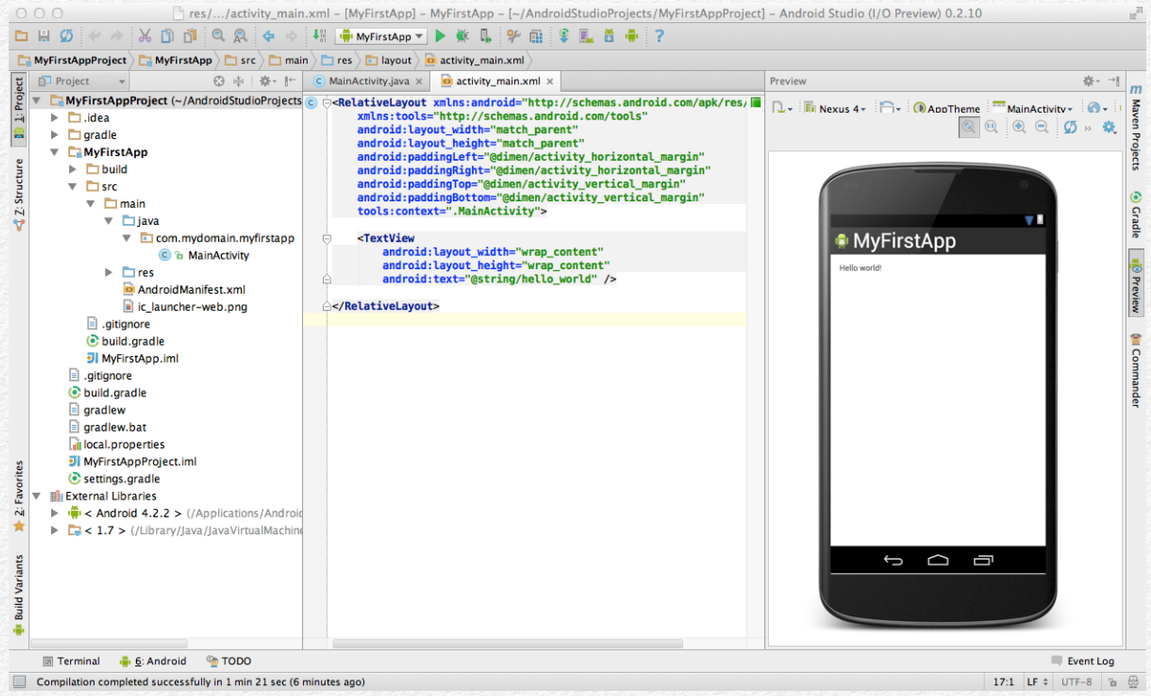
Now the system asks for the “Activity Name”, which is the main class of the project related to the window, and the “Layout Name”. The default values will be used for this window.



Now the system will take a while to prepared the layout and the project file structure.

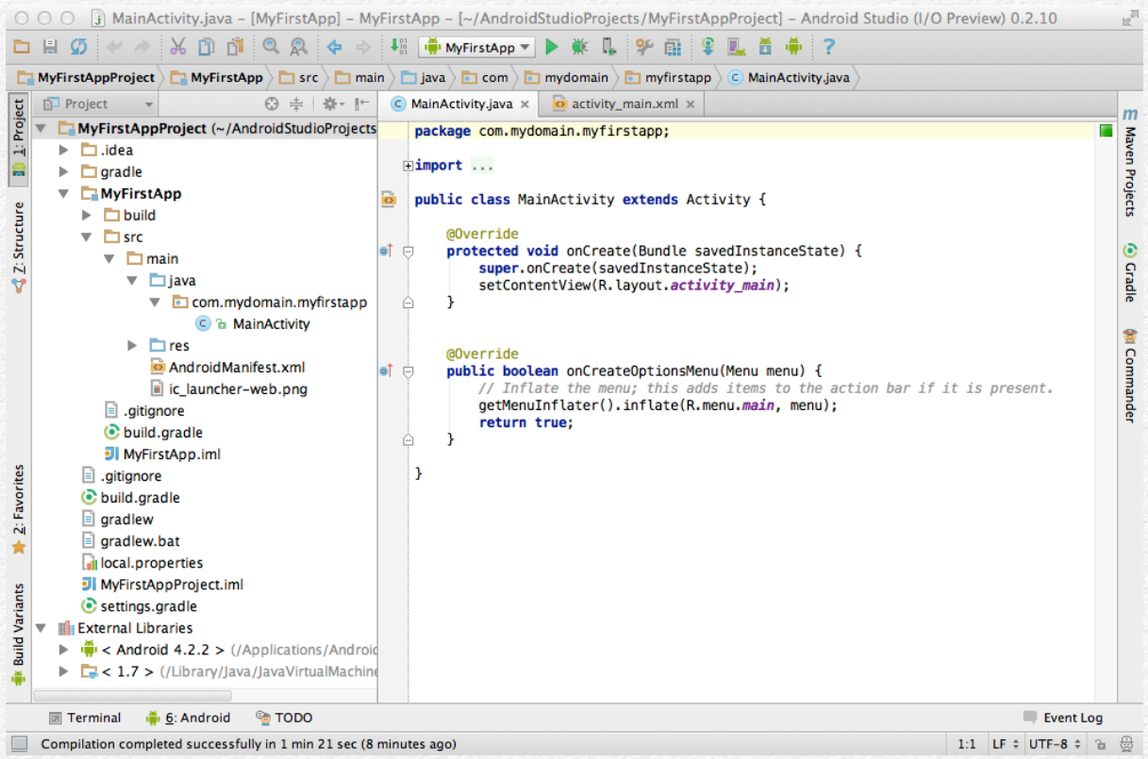
STEP 8

Click the project button or select the menu “View>Tool Windows>Project”. Expand the project files tree.



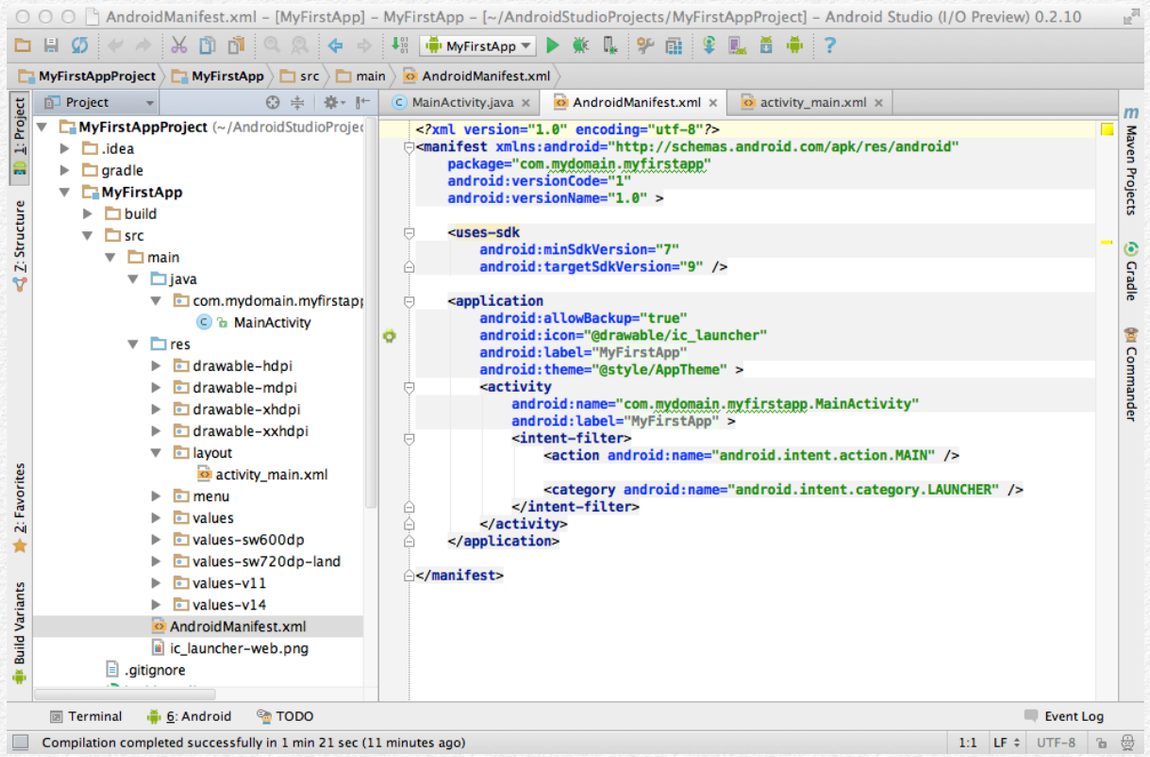
STEP 9

Have a look to the activity_main.xml layout, and the MainActivity.java file, by selecting the corresponding tab. This files refer to the main window of the application, which shows a string called “Hello World”



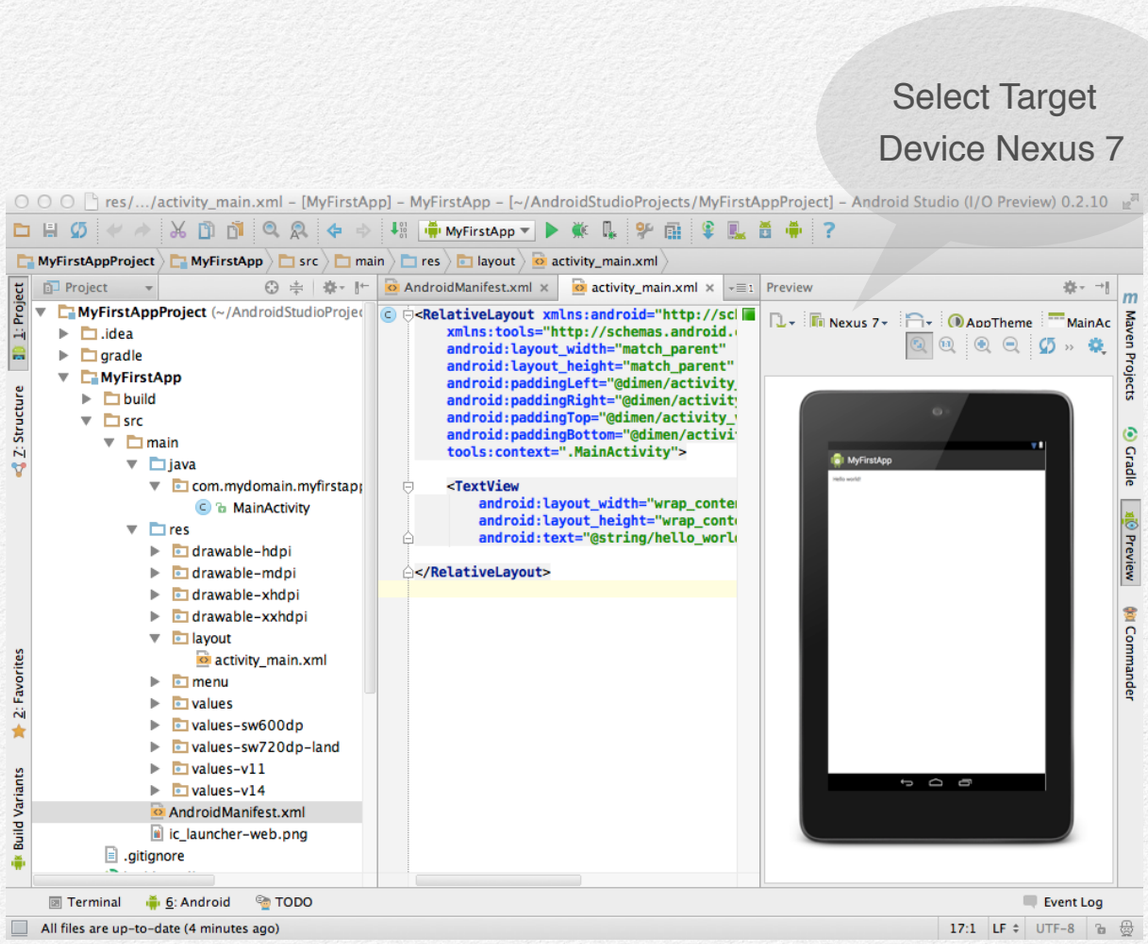
STEP 10

Have a look to the file “MyFirstApp>src>main>res>AndroidManifest.xml”, where the details and configuration parameters of the application are stored. Depending on the resources the application is going to use from the device (e.g. Internet, Bluetooth, etc.), this file must be updated accordingly to get permission from the operating system.



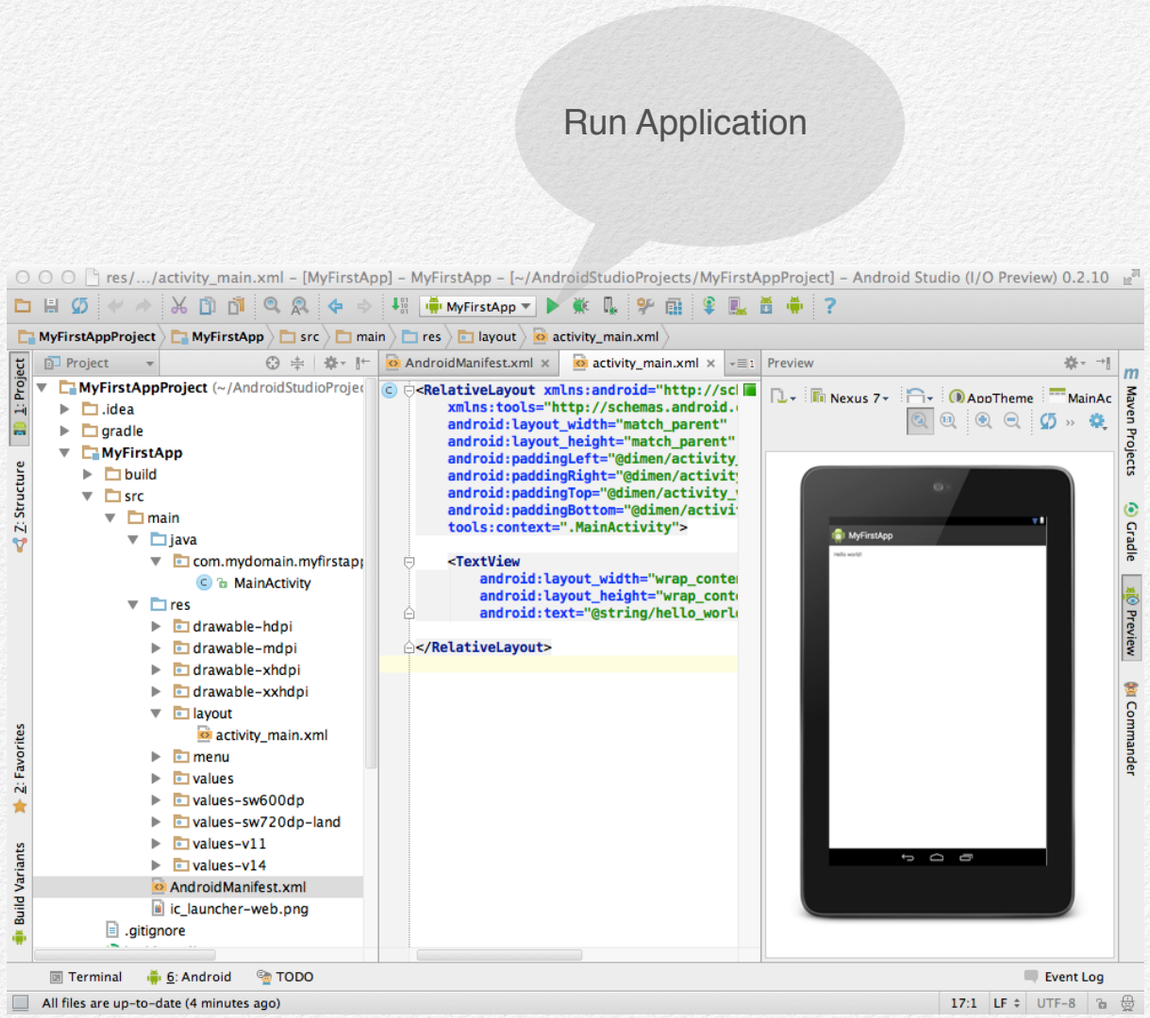
STEP 11

Select the “main_activity.xml” file to visualize the actual layout. In this step we are going to associate the layout to the Nexus 7 tablet. For this, we can select the device button on top of the preview window, as shown in the figure.



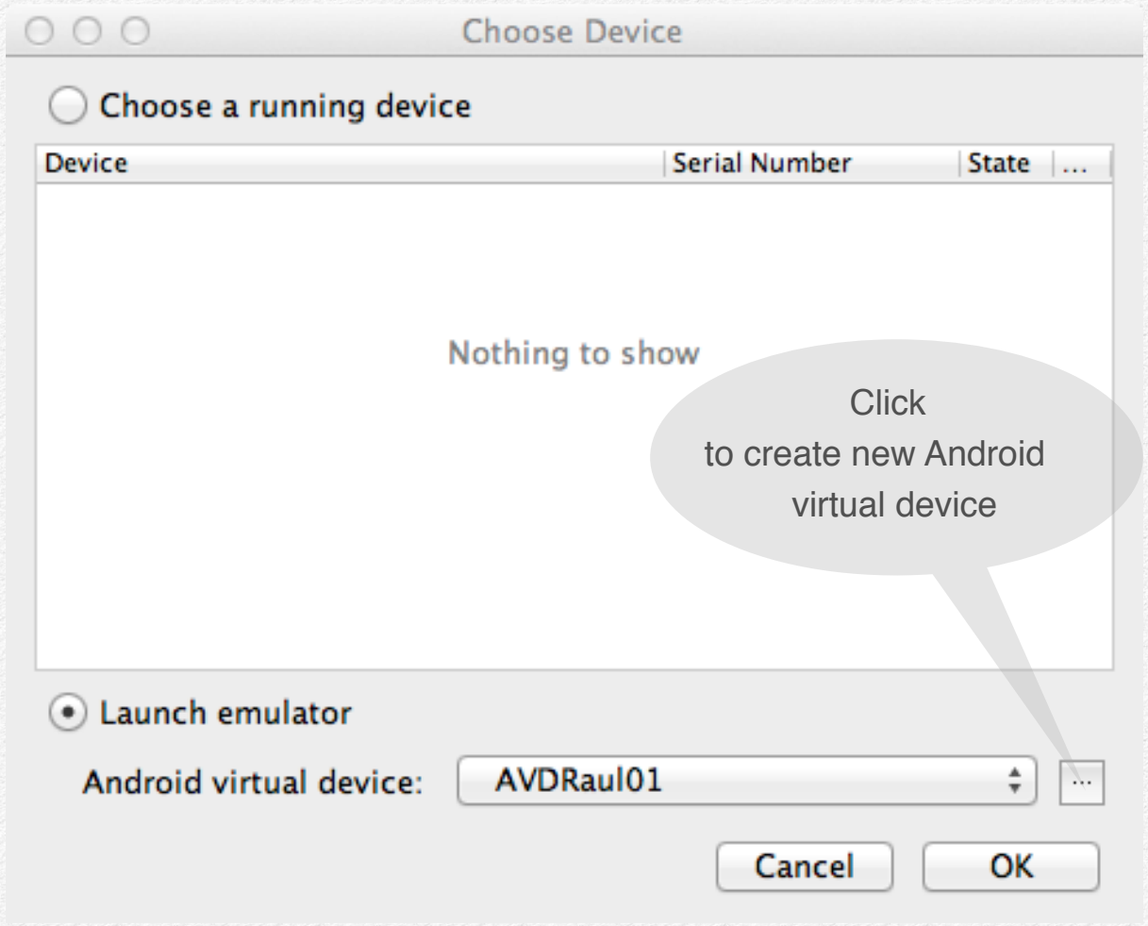
STEP 12

Select the “Run ‘MyFirstApp’” button, on the tool bar.



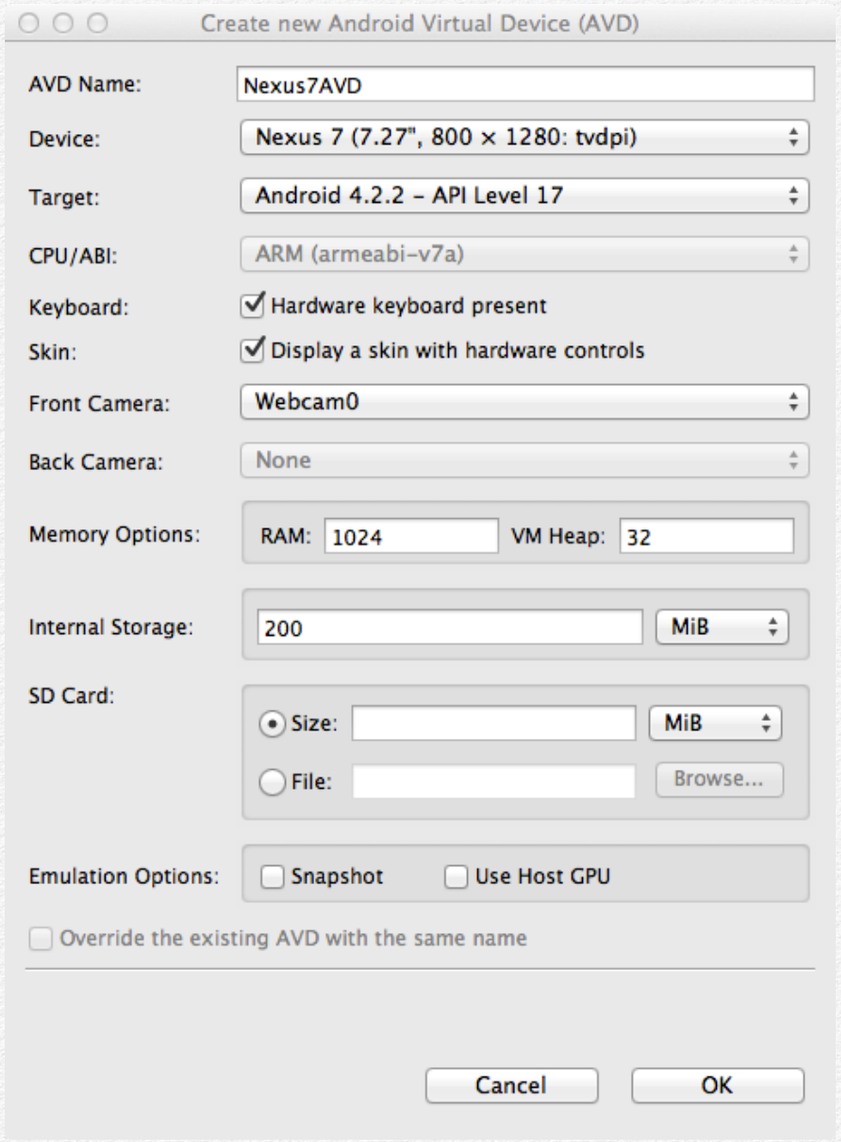
STEP 13

After a while, the select target window will come up, as shown in the figure. For the moment, as we have not connected the Nexus 7 tablet to the computer yet, the only option available is the "Launch emulator". If the computer does not have an Android virtual device yet, click on the button shown in the figure and follow the next step. Otherwise, click the OK button.



STEP 14

To create a new Android Virtual Device, on the "Android Virtual Device Manager" window select the option "New", and use the values shown in the next figure. Press ok.



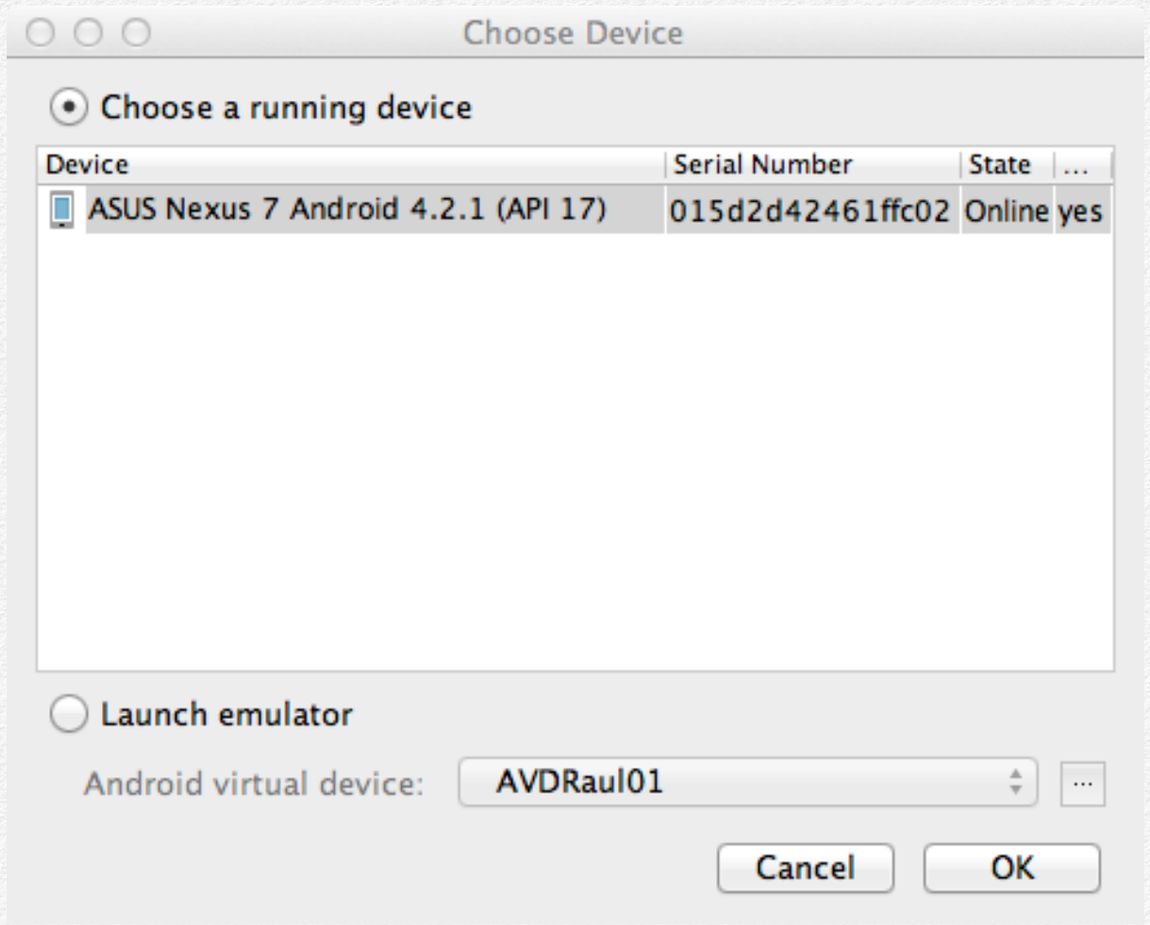
STEP 15

Since Android 4.2 the developer settings have been hidden. In order to allow the program to be executed on the real Nexus 7 Android device, it is necessary to perform the following steps.

1. Select the “Settings” options.
2. Scrool down and select the option “About Tablet”.
3. Tap seven times the “Build number” option. A dialog will come up telling “Now you are a developer”. Go to the “Developer Options” and activate the “USB Debugging”.

STEP 16

Connect the Nexus 7 device to the USB and run the application. Once the “Choose Device” dialog comes up select the tablet and press ok. The application should come up in the tablet.



Designing a Simple User Interface

Summary

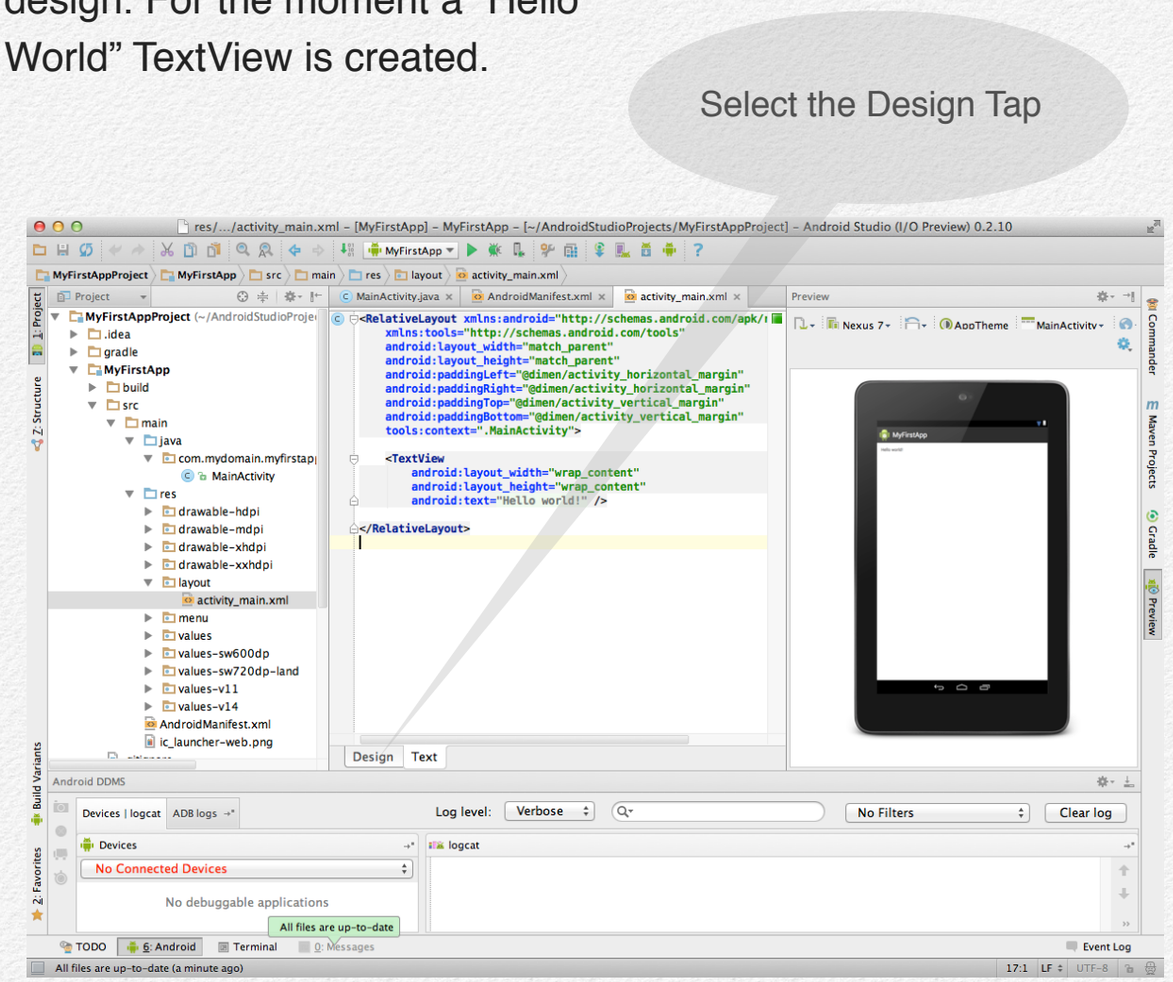
1. **In this section we are going to design a simple Android user interface that gets the user input from buttons.**
2. **The user interface will provide a simple input for an already existing Arduino-based car, which is able to move forward, backward, left and right, accordingly.**
3. **For the moment this section will focus on the user interface design, and later on the required bluetooth commands to move the robot from the Android device will be explained.**

STEP 1

Launch the Android Studio Platform and wait for the sytem to come up. Open the “MyFirstApp” project developed in the previous section.

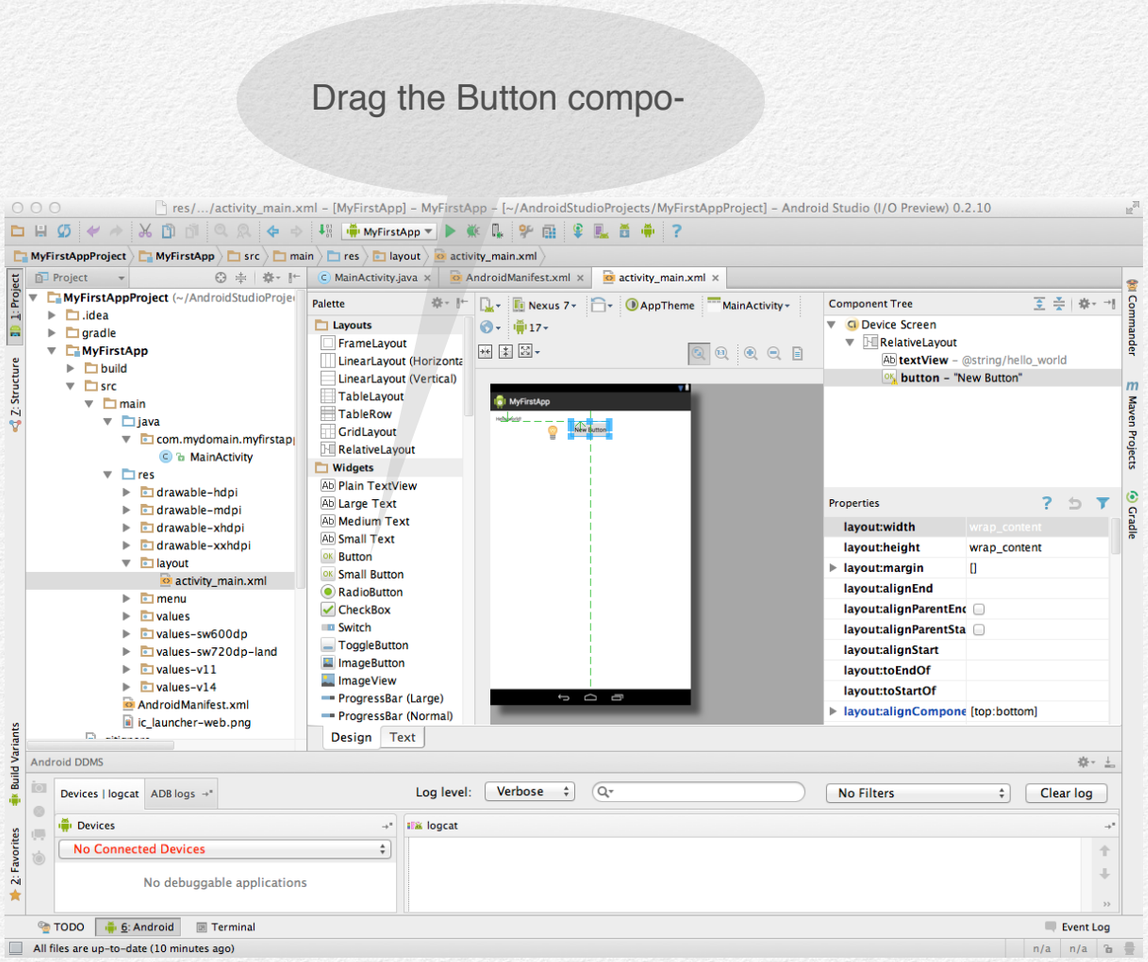
STEP 2

Open the activity_main.xml file, which holds the user interface design. For the moment a “Hello World” TextView is created.



STEP 3

Select the “Design” tab to enable the user interface edition. After that, click and drag the palette button component to the layout. This button will represent moving the robot forward, so it is convenient to place it at the top-center of the window.



STEP 4

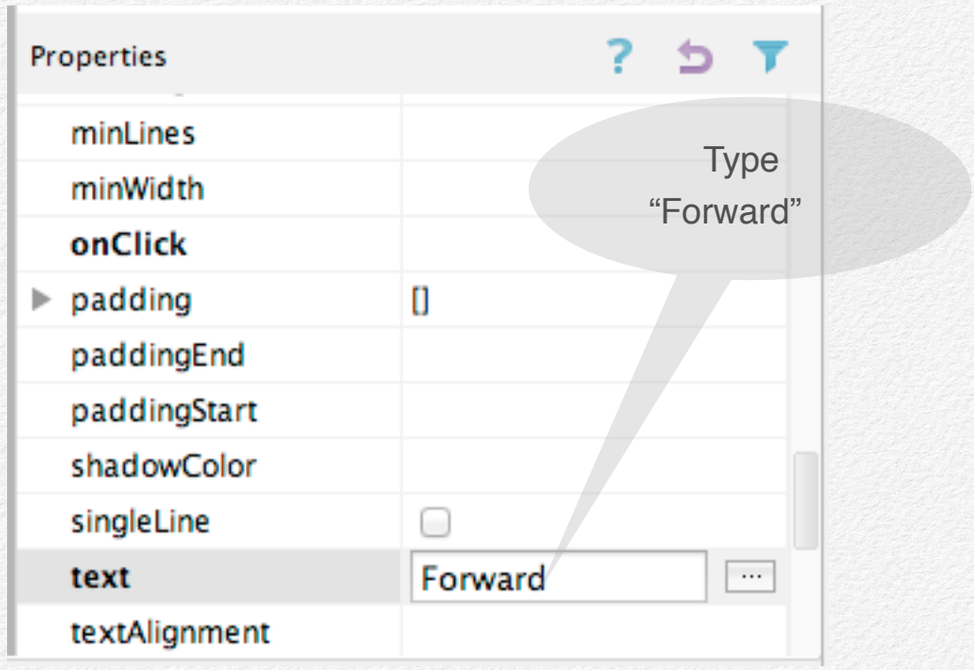
Select the “Design” tab to enable the user interface edition. After that, click and drag the palette button component to the layout. This button will represent moving the robot forward, so it is convenient to place it at the top-center of the window.

STEP 5

Run the application on the simulator or your device to see the button in action.

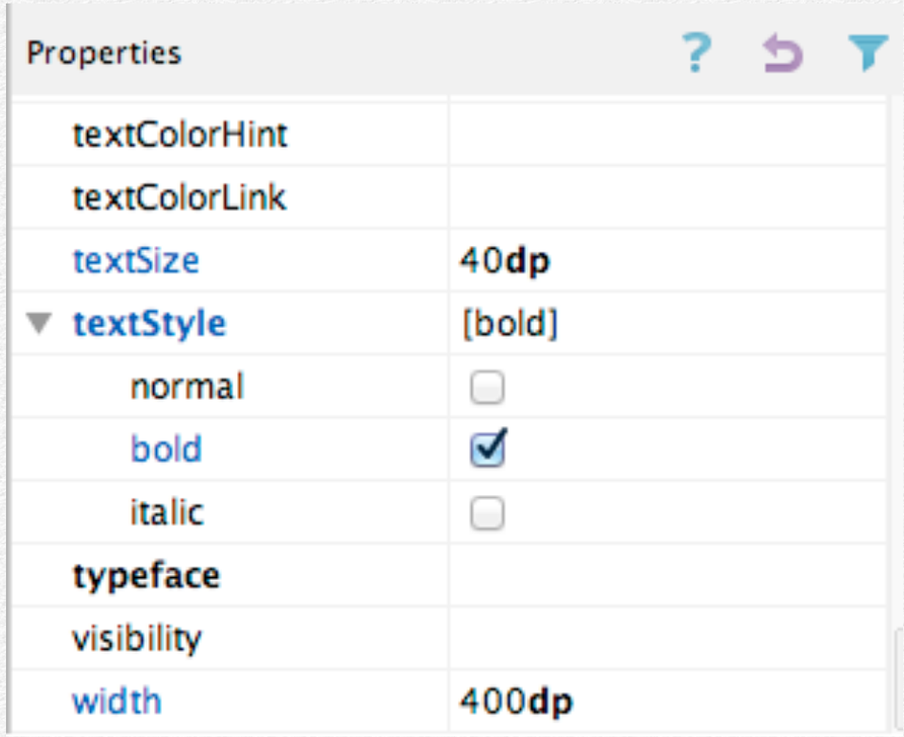
STEP 6

In the “Properties” window look for the “text” attribute and type the “Forward” string.



STEP 7

Update more properties of the button, such as “height”, “width”, “textStyle”, and “textSize”, to get a more friendly input.

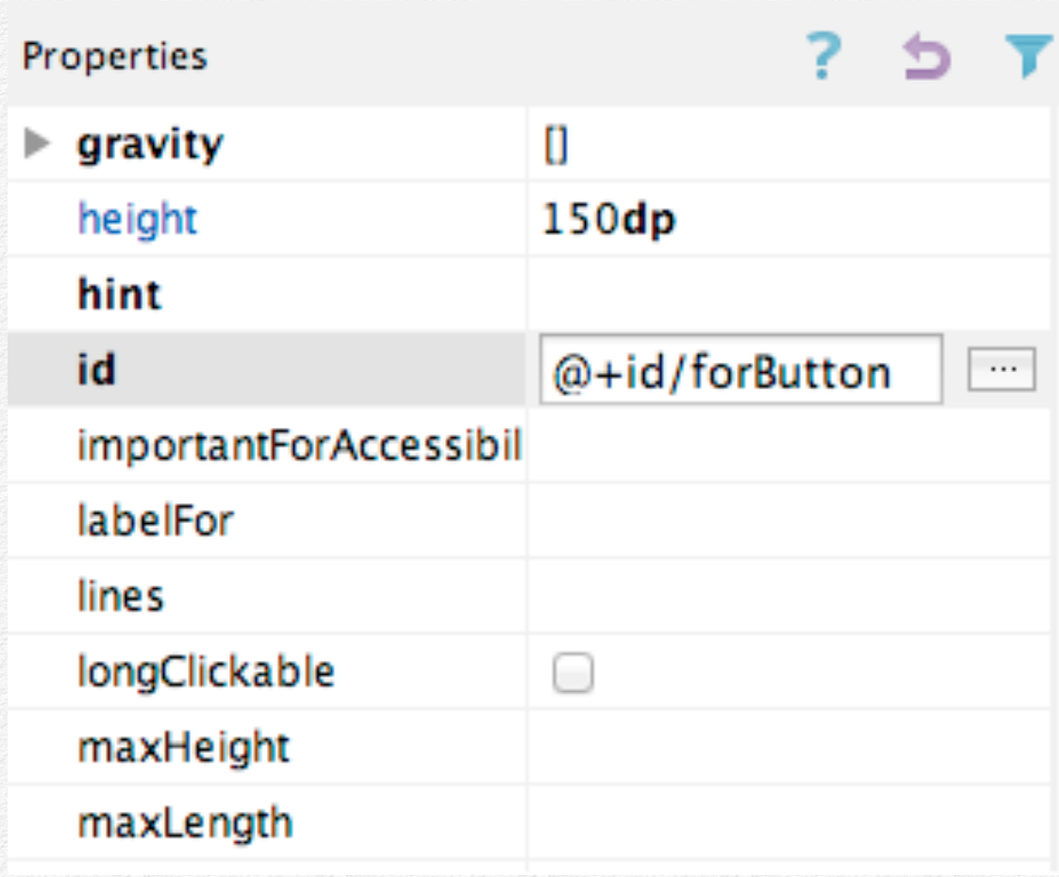


STEP 8

Update more properties of the button, such as “height”, “width”, “textStyle”, and “textSize”, to get a more friendly input.

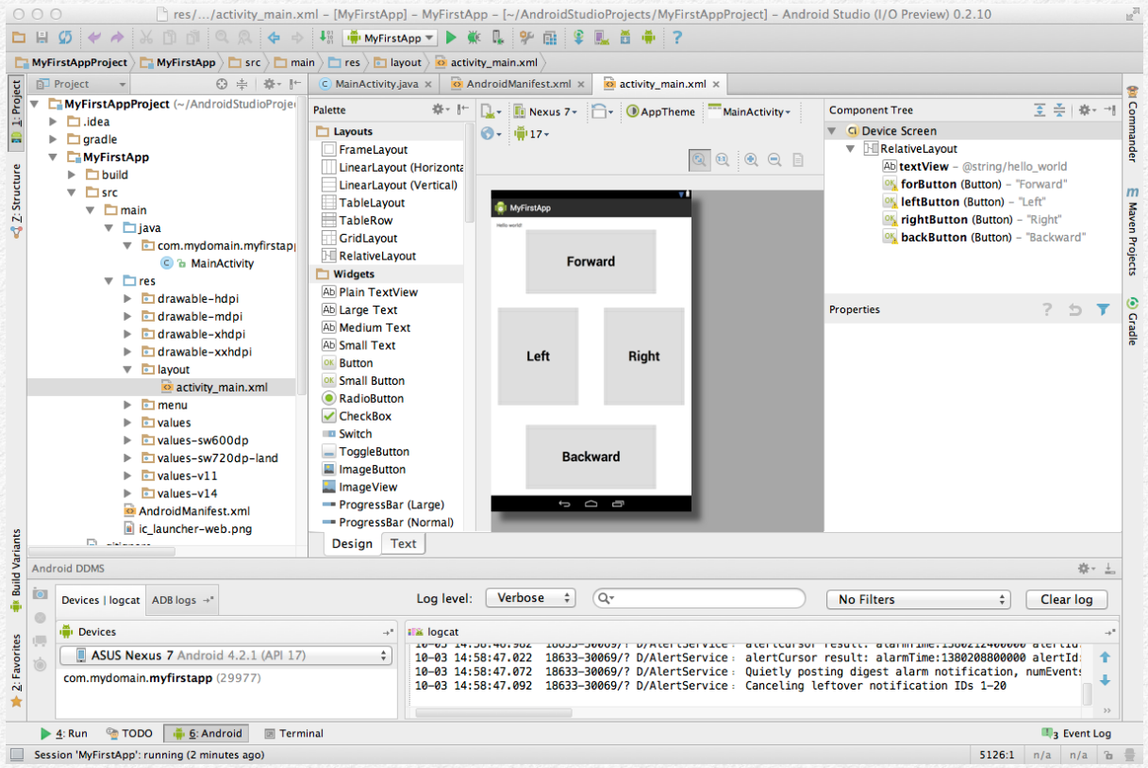
STEP 9

One of the most important properties is the “id”, which gives a name to the variable in order to manage it from the Java code. Update the “id” properties of the button to the name “forButton”, as shown in the next figure.



STEP 10

Create the corresponding “backward”, “left”, and “right” buttons, following the instructions above.

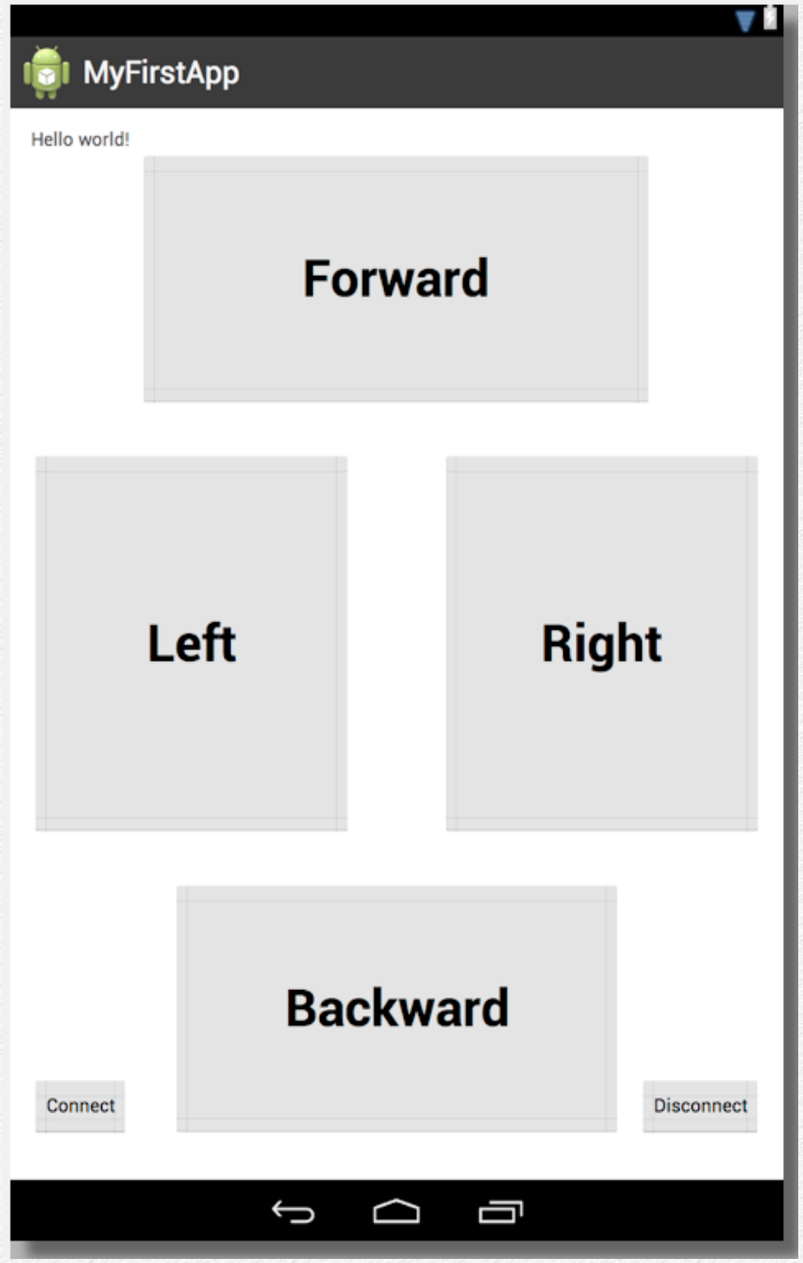


STEP 11

Run the application on the simulator or the tablet, and make sure the buttons function properly.

STEP 12

Add two small buttons for connecting and disconnecting from the bluetooth car.



STEP 13

Now we are going to study how to manage the user interface components from the Java code. Within the “MainActivity.java” file define a Button variable for every component, as shown in the figure. Remember that code variables and components in a graphical layout are decoupled.

```
public class MainActivity extends Activity {


    //User Interface Variables

    Button connectButton;

    Button disconnectButton;

    Button forwardButton;

    Button backwardButton;

    Button turnLeftForwardButton;

    Button turnRightForwardButton;
```

STEP 14

Add the following import sentence to the “MainActivity.java” file, to let the compiler know where the “Button” component is local-ted

```
    import android.widget.*;
```

STEP 15

Add the following sentences to the “onCreate” method of the “MainActivity.java” file, to link the user interface components to the corresponding Java variables. The variable “R” links to the actual project resources.

```
    connectButton = (Button) findViewById(R.id.connectButton);

    disconnectButton = (Button) findViewById(R.id.disconnectButton);

    forwardButton = (Button) findViewById(R.id.forButton);

    backwardButton= (Button) findViewById(R.id.backButton);

    turnLeftForwardButton= (Button) findViewById(R.id.leftButton);

    turnRightForwardButton= (Button) findViewById(R.id.rightButton);
```

STEP 16

Now we are goind to add an action event to every button. For this, we are goind to use the previous “TextView” object, where the “Hello World” string was presented. Once a button will be pressed, the “TextView” object will show this action to the user. For example, once the “Connect” button is pressed, the text associated to the label will be upated to the “Connect button pressed” value. For this, we need to associate a variable to the TextView graphical object. Add the following line to the the “onCreate” method.

```
statusLabel = (TextView) findViewById(R.id.textView);
```

STEP 17

Add a new method to the “MainActivity” class to respond to the “Connect” button interaction. If necessary, add the “import android.view.View” sentence.

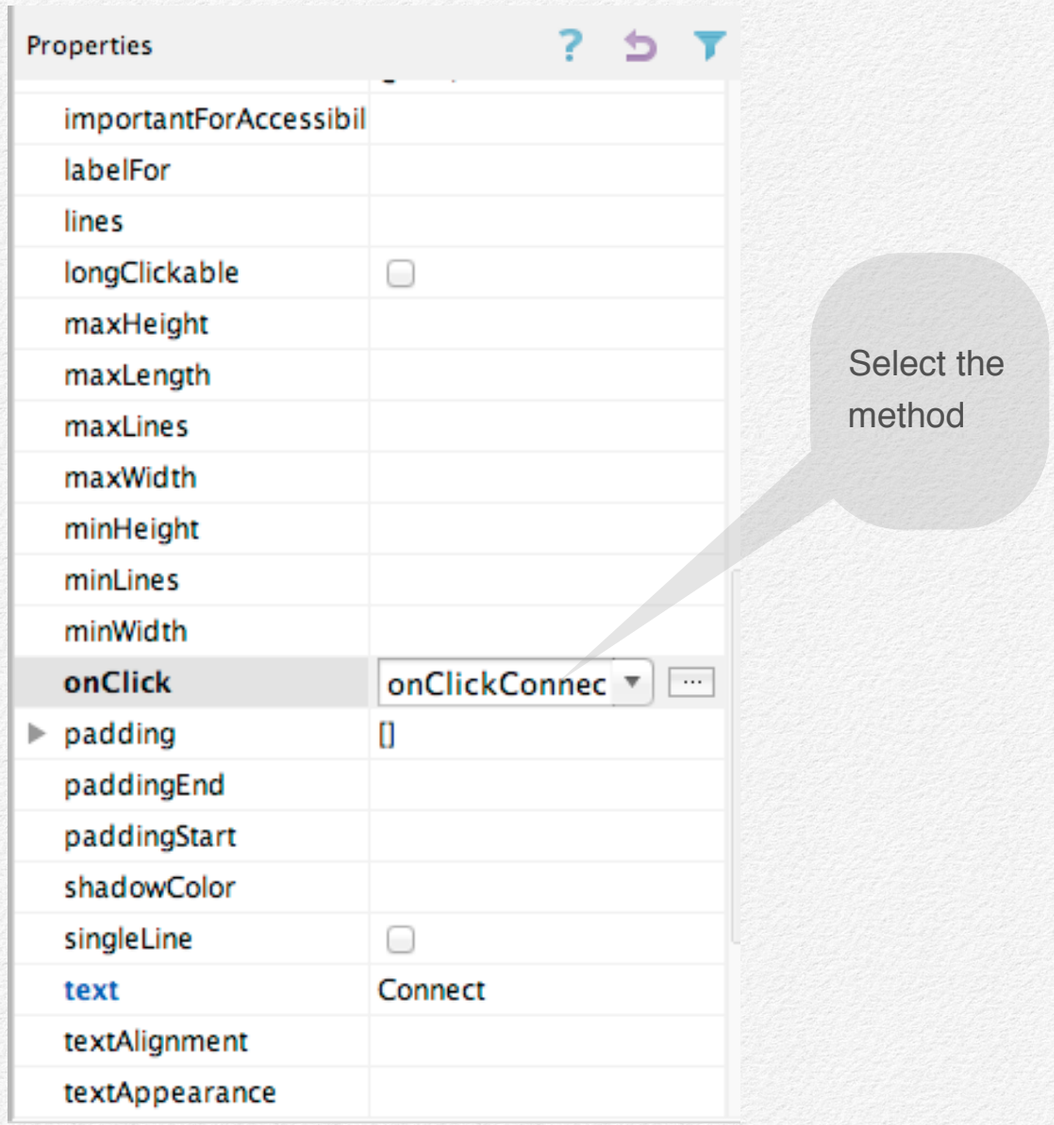
```
public void onClickConnectButton(View view){

    statusLabel.setText("Connect pressed");

}
```

STEP 17

Assign the “onClickConnectButton” method to the “onclick” property. This option should be available automatically.



Lab 2: Bluetooth Interconnection

2

In this Lab session the Bluetooth interconnection is presented, first connecting to an Arduino board provided with 4 motors and their corresponding wheels, and secondly creating a master slave bluetooh application between two Nexus 7 Android platforms.

Author: Raúl Marín

Bluetooth Enabling and Discovering

Summary

1. **In this section we are going to explain how to enable the use of Bluetooth communications in our application.**
2. **Moreover, the basic Bluetooth API to discover remote devices is presented.**
3. **At the moment of writing, to connect two bluetooth devices is necessary to get them paired through a connection, which will be shown in the next section.**

STEP 1

Study the BluetoothAdapter class:

http://developer.android.com/reference/android/bluetooth/BluetoothAdapter.html

As explained in the URL below, to get a BluetoothAdapter representing the local Bluetooth adapter, when running on JELLY_BEAN_MR1 (Android 4.2) and below, call the static getDefaultAdapter() method; when running on JELLY_BEAN_MR2 (Android 4.3) and higher, retrieve it through getSystemService(String) with BLUETOOTH_SERVICE. Fundamentally, this is your starting point for all Bluetooth actions. Once you have the local adapter, you can get a set of BluetoothDevice objects representing all paired devices with getBondedDevices(); start device discovery with startDiscovery(); or create a BluetoothServerSocket to listen for incoming connection requests with listenUsingRfcommWithServiceRecord(String, UUID); or start a scan for Bluetooth LE devices with startLeScan(LeScanCallback).

STEP 2

Following the Android code example of the chapter 1, we are going to modify the method action associated to the “Connect” button. Access the default Bluetooth adapter by adding the following code:

```
BluetoothAdapter bluetooth = BluetoothAdapter.getDefaultAdapter();
```

NOTE: if the “bluetooth” variable returns “null” it means the Android device does not have a Bluetooth adapter.

STEP 3

Import the following class:

```
import android.bluetooth.BluetoothAdapter;
```

STEP 4

Modify the application manifest to add permission for using the Bluetooth adapter. Add the following lines to the file “AndroidManifest.xml”.

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

STEP 5

Check if the bluetooth adapter is enabled by adding the following code. Extract your bluetooth name and adress.

```
if(bluetooth.isEnabled()){
        String address = bluetooth.getAddress();
        String name = bluetooth.getName();
    }
```

STEP 6

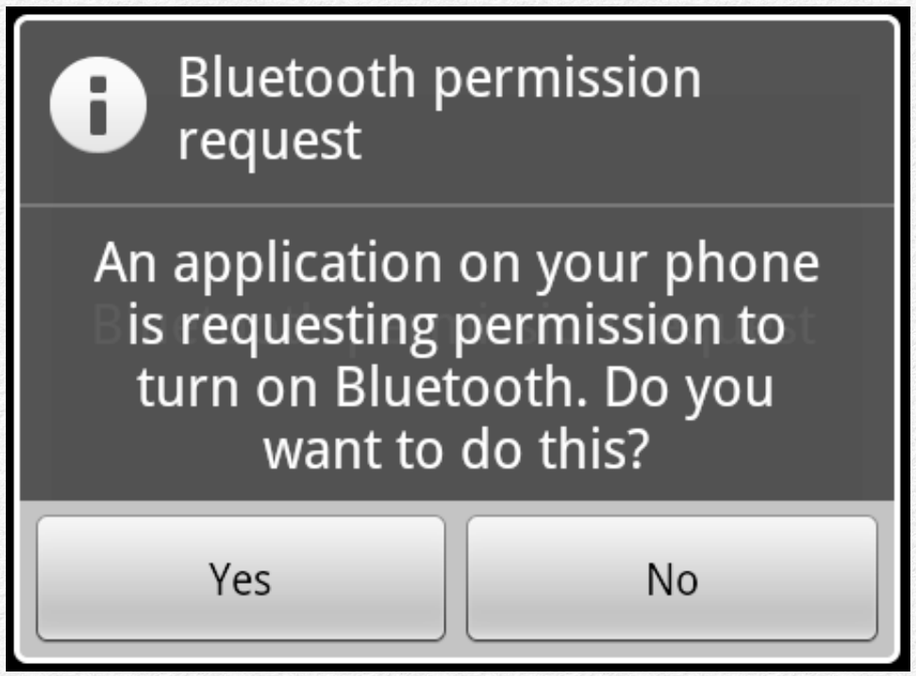
Disable and enable the bluetooth adapter in the Android system by changing the preferences, and check the result of the function “bluetooth.isEnabled()”.

STEP 7

Add the following code to the connect action, as long as the Bluetooth adapter is disabled, to ask the user to enable it.

```
startActivityForResult(new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE),1);
```

As the result, the following window will come up.



STEP 8

In order to capture the user selection it is necessary to add to the “MainActivity.java” file a method called “onActivityResult”, which permits to program the appropiate code depending on the button selected by the operator.

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == 1) //Bluetooth permission request window
        if (resultCode == RESULT_OK){
            statusLabel.setText("User Enabled Bluetooth"+requestCode);
        }else{
            statusLabel.setText("User Did not enable Bluetooth"+requestCode);
        }
}
```

The “requestCode” variable stores the Intent idendifier specified in STEP 7 (i.e. “1”).

STEP 9

Add a global variable to the “MainActivity.java” file called “bluetoothActive”, which stores the availability of the bluetooth adapter in order to proceed with the connection.

Update the “bluetoothActive” variable accordingly in both, the “Connect” button action and the “onActivityResult”.

STEP 10

Declare a global “ArrayList” variable to store the discovered devices.

```
private ArrayList<BluetoothDevice> deviceList =
new ArrayList<BluetoothDevice>();
```

;

STEP 11

Create a new private method called “startDiscovery”. We are going to include in this method the necessary Java code to discover the remote bluetooth devices.

```
private void startDiscovery(){
    if (bluetoothActive){
        deviceList.clear();
        registerReceiver(discoveryResult, new
                IntentFilter(BluetoothDevice.ACTION_FOUND));
        //Device's discovery
        statusLabel.setText("Bluetooth startDiscovery");
        Log.d("MyFirstApp", "startDiscovery");
        bluetooth.startDiscovery();
    }
}
```

As the “startDiscovery” action is asynchronous, it is necessary to create an object (i.e. BroadcastReceiver) that receives the bluetooth discovery events and treats them accordingly. For this, an “IntentFilter” object is created, which is associated to the “BluetoothDevice.ACTION_FOUND” discovery event.

STEP 12

Create the “BroadcastReceiver” object to print the information from the discovered bluetooth devices.

```
BroadcastReceiver discoveryResult = new BroadcastReceiver() {
    @Override
public void onReceive(Context context, Intent intent) {
    String remoteDeviceName =
            intent.getStringExtra(BluetoothDevice.EXTRA_NAME);
    BluetoothDevice remoteDevice =
            intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
    int rssi =
        intent.getShortExtra(BluetoothDevice.EXTRA_RSSI,Short.MIN_VALUE);
    deviceList.add(remoteDevice);
    Log.d("MyFirstApp", "Discovered "+ remoteDeviceName);
    Log.d("MyFirstApp", "RSSI "+ rssi + "dBm");
}
```

Lab 3: Wifi Interconnection

3

In this Lab session the Wifi interconnection is presented. First of all, some resources are presented for managing and monitoring the network. Secondly, a client/server application will be presented.

Author: Raúl Marín

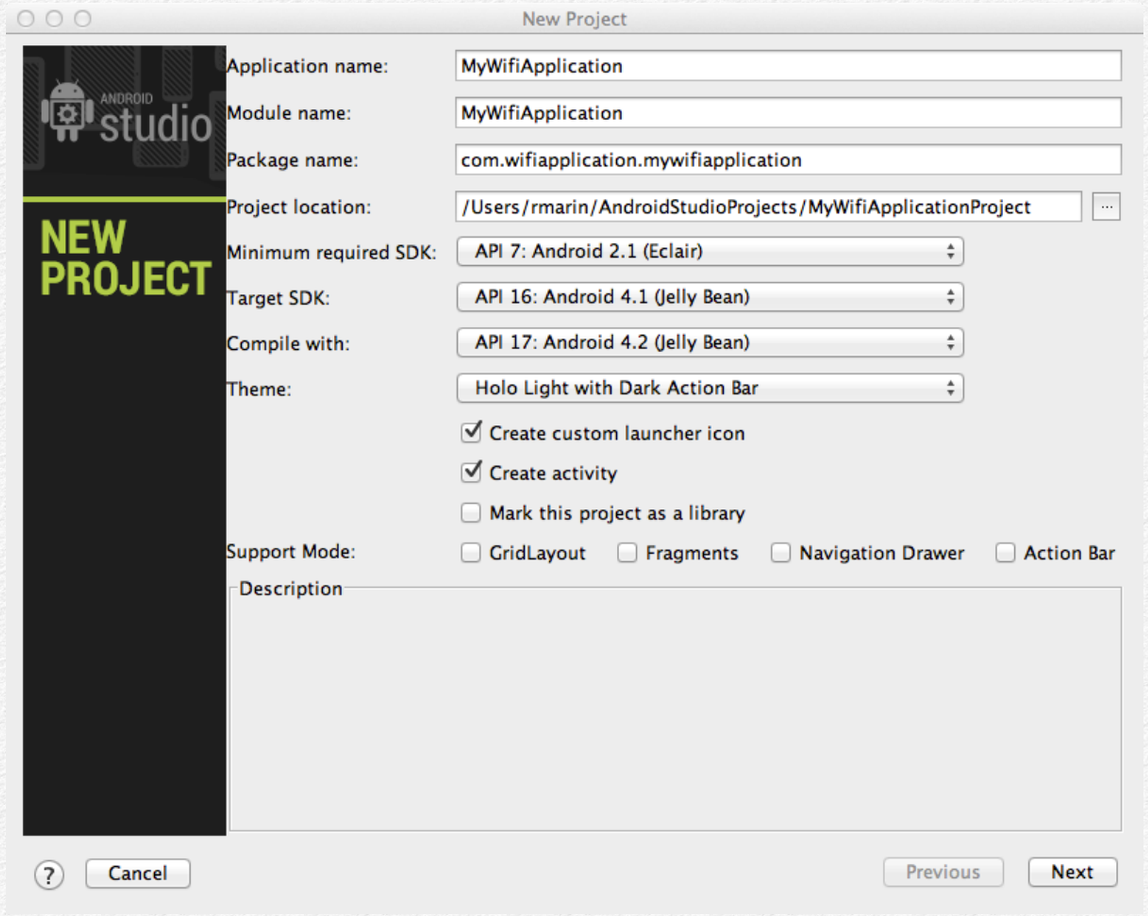
Managing the Wifi connection

Summary

1. First of all, in this section we are going to study the “Connectivity Manager”, which permits to get the information about the way the application is connected to the networks.
2. After that, the section will focus on the “WifiManager” object, which permits the configuration of Wifi connections, managing them, scaning networks and monitoring their state.

STEP 1

Create a new Android project using the Blank Activity view. Call it “MyWifiAplication”.



STEP 2

Open the “AndroidManifest.xml” file and add the following lines to activate the “Connectivity Manager”.

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

STEP 3

The main object to get information and manage the wifi adapter is the “Connectivity Manager”. To access this object add this code to the “onCreate” method in your main activity.

```
String service = Context.CONNECTIVITY_SERVICE;
ConnectivityManager connectivity = (ConnectivityManager) getSystemService(service);
```

As explained in the following reference:

http://developer.android.com/reference/android/net/ConnectivityManager.html

The primary responsibilities of this class are to:

1. Monitor network connections (Wi-Fi, GPRS, UMTS, etc.)
2. Send broadcast intents when network connectivity changes
3. Attempt to "fail over" to another network when connectivity to a network is lost
4. Provide an API that allows applications to query the coarse-grained or fine-grained state of the available networks

STEP 4

To get access to the state of the available network connections use the "ConnectivityManager.getActiveNetworkInfo()" method, which returns a "NetworkInfo" object with the connection values. Add the following code to the "onCreate" method.

```
NetworkInfo activeNetwork = connectivity.getActiveNetworkInfo();
    boolean isConnected = ((activeNetwork != null) &&
            activeNetwork.isConnectedOrConnecting());
    boolean isWifi = (activeNetwork.getType() == ConnectivityManager.TYPE_WIFI);
```

STEP 5 (Exercise)

Create a user interface, containing an “Explore Connection” button, that once pressed shows in the screen if the connection is Wifi, Bluetooth and its properties (name and type).

STEP 6

Now we are going to study the “WifiManager”, which permits managing the Wifi connections. First of all ask permission to the platform for managing the network, by adding the following lines to the manifest:

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

STEP 7

Access the WifiManager object by adding the following code:

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

STEP 8

Once the user presses the “Explore Connection” button in the application, use the WifiManager object to check if the wifi connection is enabled, and if not, enable it by using the following code as basis:

```
if (!wifiManager.isWifiEnabled()){
    if (wifiManager.getWifiState() != WifiManager.WIFI_STATE_ENABLING){
        wifiManager.setWifiEnabled(true);
        Log.v("MyWifiApplicationLOG","Activating wifi");
    }
}
```

STEP 9

Study the WifiManager reference and experiment with some of its methods, such as the “getConnectionInfo()”.

http://developer.android.com/reference/android/net/wifi/WifiManager.html

STEP 10 (Exercise)

Add a new TextView to the user interface to present the actual RSSI for the active wifi connection. Move around and examine how this data changes depending on the actual position related to the Access Point.

UDP Interconnection

Summary

1. **In this section the Android UDP Interconnection is studied. First all all, the basic Java UDP Networking API will be presented through a client server example (i.e. the eHealth platform), secondly, the steps to design the client side user interface in Android are explained.**

STEP 1

Study the UDP client server examples provided for the eHealth platform (i.e. SimuladorSensorEHealthUDPWifi.java & SimuladorInterfazAndroidSensorEHealthUDPWifi.java)

STEP 2

Make sure the following permissions are added to the Android Manifest file:

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

STEP 3

Following the example initiated in the previous section, add two new TextView objects to the user interface to represent the “temperature” and the “calm” values provides by the eHealth sensor.

STEP 4

Run the “SimulatorSensorEHealthUDPWifi” program in your desktop computer. Make sure which is the IP address of the computer to adjust the client code accordingly.

STEP 5

Run the “SimuladorInterfazAndroidSensorEHealthUDPWifi” program in your desktop computer, and check the UDP interconnection is working properly.

STEP 6

Adapt the client side UDP code to your Android application and try it.

STEP 7

Please note the network TCP/IP interconnection must be launched through a thread, for not blocking the main one.

STEP 8

Once the networking thread is launched by using the “start” method, from the main thread wait for it to conclude by using the “join” method. Take into account that the user interface View must be updated from the main thread, and not from the networking one.