Android Development: Necessary Software

Michalis Katsarakis katsarakis@csd.uoc.gr
M.Sc. Student

Tutorial: hy439 & hy539 http://www.csd.uoc.gr/~hy439/

Introduction

This is a concise installation guide for the software needed to start developing Android applications. The software needed is

1. Android SDK
2. ADT Plugin
3. Some Android SDK packages
4. USB drivers (not always)

Some of the sections “Google USB Driver” and “Setting up a Device for Development: 3. Set up your system to detect your device” may not longer be necessary. Despite that, do not simply skip them. Instead, verify that you can run the application described in section “**Checking that everything works fine**” on your Android device.

Android SDK

The Android SDK provides you the API libraries and developer tools necessary to build, test, and debug apps for Android. Choose the SDK package for your OS from the link below: http://developer.android.com/sdk/index.html

If you choose a .zip or .gzip package, you will have to extract it. It is popular to place the extracted directory in your home folder (e.g., `/home/michalis/android-sdk-linux_x86`)

Eclipse

If you haven’t already installed the eclipse IDE, you can download it from http://www.eclipse.org/downloads/packages/eclipse-classic-421/junosr1 and install it.

ADT Plugin for Eclipse

Android Development Tools (ADT) is a plugin for the Eclipse IDE that is designed to give you a powerful, integrated environment in which to build Android applications.

To install the ADT plugin, follow the steps below:

1. Start Eclipse, then select **Help** > **Install New Software**.
2. Click **Add**, in the top-right corner.
3. In the Add Repository dialog that appears, enter "ADT Plugin" for the *Name* and the following URL for the *Location*:
4. https://dl-ssl.google.com/android/eclipse/
5. Click **OK**.
6. If you have trouble acquiring the plugin, try using "http" in the Location URL, instead of "https" (https is preferred for security reasons).
7. In the Available Software dialog, select the checkbox next to Developer Tools and click **Next**.
8. In the next window, you'll see a list of the tools to be downloaded. Click **Next**.
9. Read and accept the license agreements, then click **Finish**.
10. If you get a security warning saying that the authenticity or validity of the software can't be established, click **OK**.
11. When the installation completes, restart Eclipse.

Once Eclipse restarts, you must specify the location of your Android SDK directory:

1. In the "Welcome to Android Development" window that appears, select **Use existing SDKs**.
2. Browse and select the location of the Android SDK directory you recently downloaded.
3. Click **Next**.

For more detailed instructions, see http://developer.android.com/sdk/installing/installing-adt.html.

Android SDK packages

Either using the  button on the Eclipse IDE or by executing "`/home/michalis/android-sdk-linux_x86/tools/android`" launch the Android SDK Manager. This application enables the installation of packages for the android SDK. You will have to install the following:

1. The tools folder (both "Android SDK Tools" and "Android SDK Platform-tools")
2. Android 2.1 (API 7):
    a. SDK Platform
    b. Google APIs (in case you need to include maps in your apps)
3. Android 2.3.3 (API 10)
    a. SDK Platform
    b. Google APIs (in case you need to include maps in your apps)

In each Android version folder, there is also a package called "Samples for SDK". It contains some sample projects, which are very helpful examples, in case you want to practise further.

During the tutorial, we will develop an Android app using the Android OS version 2.1, in order to ensure that all devices are compatible. You can see the current distribution of various Android OS versions following the link: http://developer.android.com/about/dashboards/index.html.

Google USB Driver

The driver is for Windows only and provides the necessary drivers for the following devices:

- ADP1 / T-Mobile G1*
- ADP2 / Google Ion / T-Mobile myTouch 3G*
- Verizon Droid*
- Nexus One
- Nexus S

* *Or similar hardware on other carriers*

All other devices require Windows drivers provided by the hardware manufacturer, as listed in the OEM USB Drivers document. The Galaxy Nexus driver is also distributed by Samsung (listed as model SCH-I515).

Note: If you're developing on Mac OS X or Linux, then you do not need to install a USB driver. To start developing with your device, also read Using Hardware Devices.

For more detailed instructions, see: http://developer.android.com/sdk/win-usb.html.

Setting up a Device for Development

1. Turn on "USB Debugging" on your device.
2. On the device, go to **Settings > Applications > Development** and enable **USB debugging** (on an Android 4.0 device, the setting is located in **Settings > Developer options**).
3. Set up your system to detect your device.
    - If you're developing on Windows, you need to install a USB driver for adb. For an installation guide and links to OEM drivers, see the OEM USB Drivers document.
    - If you're developing on Mac OS X, it just works. Skip this step.
    - If you're developing on Ubuntu Linux, you need to add a udev rules file that contains a USB configuration for each type of device you want to use for development. In the rules file, each device manufacturer is identified by a unique vendor ID, as specified by the ATTR{idVendor} property. For a list of vendor IDs, see USB Vendor IDs, below. To set up device detection on Ubuntu Linux:
        i. Log in as root and create this file: /etc/udev/rules.d/51-android.rules.

           Use this format to add each vendor to the file:
           SUBSYSTEM=="usb", ATTR{idVendor}=="0bb4", MODE="0666", GROUP="plugdev"

           In this example, the vendor ID is for HTC. The MODE assignment specifies read/write permissions, and GROUP defines which Unix group owns the device node.

           Note: The rule syntax may vary slightly depending on your environment. Consult the udev documentation for your system as needed. For an overview of rule syntax, see this guide to writing udev rules.
        ii. Now execute:
           chmod a+r /etc/udev/rules.d/51-android.rules

When plugged in over USB, can verify that your device is connected by executing adb devices from your SDK platform-tools/ directory. If connected, you'll see the device name listed as a "device."

If using Eclipse, run or debug your application as usual. You will be presented with a **Device Chooser** dialog that lists the available emulator(s) and connected device(s). Select the device upon which you want to install and run the application.

If using the Android Debug Bridge (adb), you can issue commands with the -d flag to target your connected device.

Setting up a Virtual Device for Development

In your early steps, you can test your applications using the Android Emulator.

An Android Virtual Device (AVD) is an emulator configuration that lets you model an actual device by defining hardware and software options to be emulated by the Android Emulator.

The easiest way to create an AVD is to use the graphical AVD Manager, which you launch from Eclipse by clicking **Window > AVD Manager**. You can also start the AVD Manager from the command line by calling the android tool with the avd options, from the **<sdk>/tools/** directory.

You can create an AVD with the following parameters:

- Name: Android21
- Target: Google APIs (Google Inc.) - API Level 7
- SD Card: 64 MiB
- Skin: Built-in: WVGA800
- Hardware:
    - Abstracted LCD density: 240
    - Max VM application heap size: 24

For more details about Android Virtual Devices, see http://developer.android.com/tools/devices/index.html.

Checking that everything works fine

Now that all necessary software is installed and configured, it is time to create the simplest Android application possible and run it.

From Eclipse, click:

1. **File > New > Other... > Android > Android Application Project**
2. Enter:
    a. Application Name: HelloWorld
    b. Project Name: HelloWorld
    c. Package Name: gr.uoc.csd.hy539.helloworld
    d. Build SDK: Android 2.1 (API 7)
    e. Minimum Required SDK: API 7: Android 2.1 (Eclair)
    f. [Unchecked] Create custom launcher icon
    g. [Unchecked] Mark this project as a library
    h. [Checked] Create Project in Worksapce
    i. Click Next
3. Select
    a. [Checked] Create Activity

    b. BlankActivity
    c. Click Next
4. Type
    a. Activity Name: MainActivity
    b. Layout Name: activity_main
    c. Navigation Type: None
    d. Hierarchical Parent:
    e. Title: MainActivity
    f. Click Next
5. In case a compatibility library is needed, install it
6. Click Finish
7. Connect your Android Device to your computer, using the USB cable.
   Note: Skip this step, if you prefer to use an AVD
8. On the Package Explorer of the Eclipse IDE (left side), right click the newly created project **“HelloWorld” > Run As > Android Application**

If everything goes well, your “HelloWorld” app should be successfully launched!