# Tutorial on Basic Android Setup

EE368/CS232 Digital Image Processing, Spring 2015

Windows Version

## Introduction

In this tutorial, we will learn how to set up the Android software development environment and how to implement image processing operations on an Android mobile device. Android is an open-source platform developed by Google and the Open Handset Alliance on which interesting and powerful new applications can be quickly developed and distributed to many mobile devices. There is a large, growing community of Android developers and a vast selection of Android devices, which includes smartphones, tablets, and TV setup boxes. Android also comes with an extension library of useful functions, including functions for user interfaces, image/bitmap manipulation, and camera control that we will frequently use in EE368/CS232. We look forward to seeing your novel image processing algorithms and applications running on Android devices as the quarter progresses.

The tutorial is split into two parts. In the first part, we will explain how to download and install the Android software tools onto your computer. Then, in the second part, we will explain how to develop image processing programs that can run on an Android mobile device.

Estimated time to complete this tutorial: 2 hours

## Part I: Creating the Software Development Environment [1]

We will use the Google Android SDK and the Eclipse IDE to design, implement, and debug Android-compatible programs in this class.

*Important:* Starting from December 2014, the official IDE became Android Studio, and Eclipse is not officially supported anymore. However, there is currently no official support for the NDK (native development kit, necessary for using native code) yet, so we strongly encourage students to use Eclipse for this class.

### *Downloading and Installing Java JDK*

The Java JDK from SUN/Oracle is required for development.

1. Download the latest version of the JDK from this website:
   `http://www.oracle.com/technetwork/java/javase/downloads/index.html`

2. Execute the downloaded installer.

[1] Parts of this tutorial borrow explanations from the official Android developers' website (developer.android.com).

*Downloading the Android SDK*

1. Download the SDK Tools from this website:
   `http://developer.android.com/sdk/index.html#Other`

2. Unzip the downloaded file to a convenient location on your hard disk, for example:
   `C:\\Android\\android-sdk-windows`

*Downloading and installing Eclipse with the ADT plugin*

1. Download the latest Eclipse version for your system at this link:
   `https://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/lunasr2`

2. Unzip the downloaded file to a convenient location on your hard disk, for example:
   `C:\\Android\\eclipse`

3. Open the Eclipse application, situated in that directory.

4. When asked to choose a default workspace, pick a folder that is easy to remember and access, for example:
   `C:\\Android\\workspace`

5. Verify that Eclipse starts properly and an IDE window like in Figure 1 is shown. You can display the primary development console by choosing Window > Show View > Console, so that a console like in Figure 2 appears.

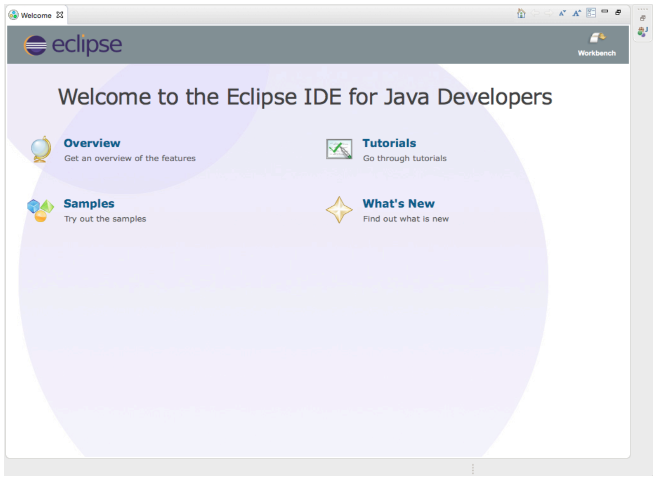
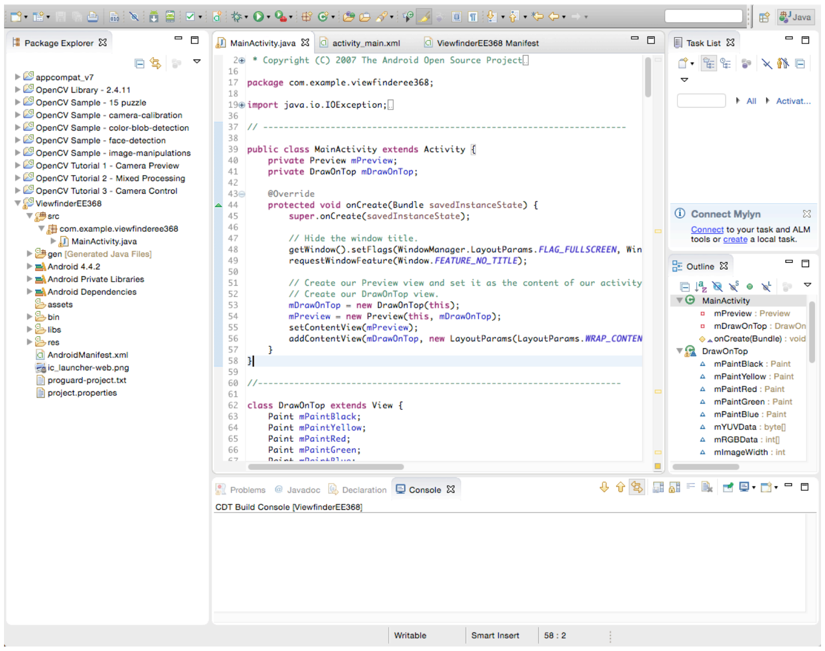


Figure 1. Initial start-up screen of the Eclipse IDE.



```java
 * Copyright (C) 2007 The Android Open Source Project

package com.example.viewfinderee368;

import java.io.IOException;

// ----------------------------------------------------------------------

public class MainActivity extends Activity {
    private Preview mPreview;
    private DrawOnTop mDrawOnTop;

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

        // Hide the window title.
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, Win
        requestWindowFeature(Window.FEATURE_NO_TITLE);

        // Create our Preview view and set it as the content of our activity
        // Create our DrawOnTop view.
        mDrawOnTop = new DrawOnTop(this);
        mPreview = new Preview(this, mDrawOnTop);
        setContentView(mPreview);
        addContentView(mDrawOnTop, new LayoutParams(LayoutParams.WRAP_CONTEN
    }
}

//----------------------------------------------------------------------

class DrawOnTop extends View {
    Paint mPaintBlack;
    Paint mPaintYellow;
    Paint mPaintRed;
    Paint mPaintGreen;
```

```
CDT Build Console [ViewfinderEE368]
```

Figure 2. Development console within the Eclipse IDE.

6. To install the ADT plugin, necessary to develop Android applications, click Help > Install New Software.

7. In the new window, enter the following URL in the top text box:
   `https://dl-ssl.google.com/android/eclipse/`

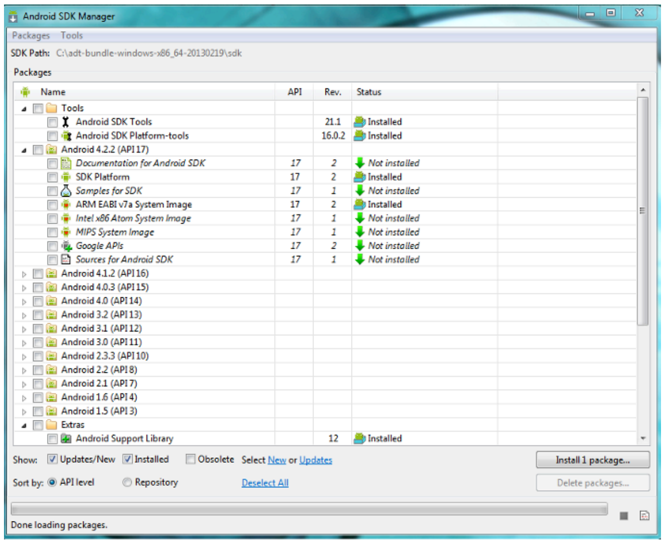
8. In the dialog below, select the checkbox for “Developer Tools” and click Next. Accept all license agreements. Click Finish.

9. Restart Eclipse after all the packages have been downloaded and installed.

*Updating the Android SDK*

1. In Eclipse, select Window > Preferences, and open the Android tab. Enter the SDK location. In the example given above, it would be:
   `C:\\Android\\android-sdk-windows`

2. In Eclipse, select Window > Android SDK Manager. A window like that in Figure 3 should pop up.


Android SDK Manager
Packages Tools
Show: [✓ Updates/New] [✓ Installed] Obsolete Select New or Updates
Sort by: [✓ API level] Repository
Deselect All
Install 1 package...
Delete packages...

SDK Path: C:\adt-bundle-windows-x86_64-20130219\sdk

Packages

| Name | API | Rev. | Status |
|---|---|---|---|
| Tools | | | |
| Android SDK Tools | | 21.1 | Installed |
| Android SDK Platform-tools | | 16.0.2 | Installed |
| Android 4.2.2 (API 17) | | | |
| *Documentation for Android SDK* | 17 | 2 | Not installed |
| SDK Platform | 17 | 2 | Installed |
| *Samples for SDK* | 17 | 1 | Not installed |
| ARM EABI v7a System Image | 17 | 2 | Installed |
| *Intel x86 Atom System Image* | 17 | 1 | Not installed |
| *MIPS System Image* | 17 | 1 | Not installed |
| *Google APIs* | 17 | 2 | Not installed |
| *Sources for Android SDK* | 17 | 1 | Not installed |
| Android 4.1.2 (API 16) | | | |
| Android 4.0.3 (API 15) | | | |
| Android 4.0 (API 14) | | | |
| Android 3.2 (API 13) | | | |
| Android 3.1 (API 12) | | | |
| Android 3.0 (API 11) | | | |
| Android 2.3.3 (API 10) | | | |
| Android 2.2 (API 8) | | | |
| Android 2.1 (API 7) | | | |
| Android 1.6 (API 4) | | | |
| Android 1.5 (API 3) | | | |
| Extras | | | |
| Android Support Library | | 12 | Installed |

Done loading packages.

Figure 3. Android SDK Manager in Eclipse.

1. In this new window, check the boxes of the packages you want to install or update. It is necessary to install/update the following packages:
   ```
   Tools > Android SDK Tools
   Tools > Android SDK Platform-tools
   Tools > Android SDK Build-tools
   Android 5.1 (API 22) > SDK Platform
   Extras > Android Support Repository
   Extras > Android Support Library
   ```
   Finally, if you plan to use the emulator (not necessary if you have already a device) you need to download a system image, for example
   ```
   Android 5.1 (API 22) > ARM EABI v7a System Image
   ```

2. Click “Install <number> packages”, choose “Accept License” for all items listed, and click “Install”. The selected packages will now be downloaded and copied to your Android SDK installation folder. You can monitor the download/installation progress at the bottom of the Android SDK Manager window.

3. Add the location of the “tools” and “platform-tools” subfolders for the Android SDK to your system PATH. For help on editing the PATH, please follow the tips here:
   `http://www.computerhope.com/issues/ch000549.htm`

*Linking Your Phone to Your Computer*

1. Connect your phone to your computer via USB. Turn on your phone.

2. Go to the home screen.

3. Select Settings > Applications > Development and then enable USB debugging.

4. On Windows, you will need to install the appropriate USB driver. Read this page for more information:
   `http://developer.android.com/tools/extras/oem-usb.html`

   Note: For the Motorola Droid, you can also download the latest USB driver from:
   `http://www.motorola.com/Support/US-EN/Support-Homepage/Software_and_Drivers/USB-and-PC-Charging-Drivers`

   Note: You may need to restart your computer after installing the USB driver in order for the phone to show up in Eclipse.

---

**Part II: Developing Image Processing Programs for Android**

Now that the Google Android SDK, the Java Runtime, and the Eclipse IDE are all set up on your computer, we are ready to start writing image processing programs that can run an Android-compatible mobile device.

*Hello World Example*

First, we will build a simple Android program in Eclipse. This simple example will also help you to become familiar with how to create an Android project, how to (auto) compile source code, and how to run the generated executable on the mobile device. Please follow the instructions on this page to develop the "My First App" program. (The tutorial was recently replaced by the corresponding Android Studio tutorial, so we give you a link to an outdated snapshot of that page)
`http://goo.gl/KSp3Oe`

In the external "My First App" tutorial, they only run the "My First App" program in an emulator. Additionally, we will now also run the program on the actual Android phone. Make sure your phone is properly linked to your computer.

1. In Eclipse, select the project MyFirstApp in the Project Explorer. Then, select Project > Properties > Android and select the highest Android version if it is not already selected. Thanks to backwards compatibility and support libraries, it does not matter if your device does not support this particular version.

2. Known issue: If you chose as "minimum required SDK" an API < 14, the additional support library "appcompat_v7" will be created. A known bug in Eclipse may show an error on that library, preventing you to build your project. To solve the problem, select the "appcompat_v7" library in the Project Explorer and press F5 (or File > Refresh) to refresh it. You can now clean it and rebuild it by choosing Project > Clean and then Project > Build project. The error should be gone.

3. Select the project MyFirstApp again, then Run > Run Configurations > Android Application > MyFirstApp > Target. Choose "Always prompt to pick device".

4. Select Run, and in the Device Chooser dialog, select your Android phone. The "My First App" program will be sent to and automatically started on your phone, and you should see the screen similar to Figure 4 on your phone.

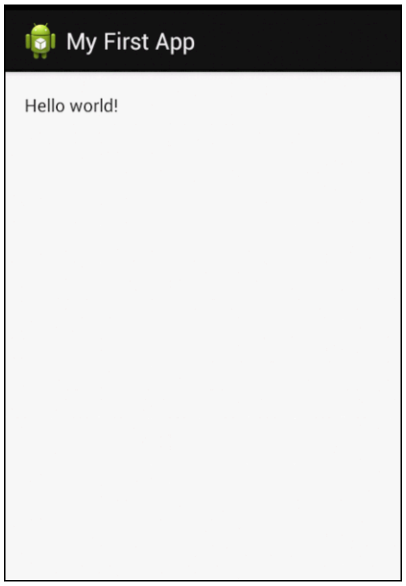


Figure 4. "My First App" program running on an Android phone.

*EE368 Viewfinder Example*

Now, having grasped the fundamentals of building and running an Android application, we will create a more complicated project involving the onboard camera and real-time image processing.

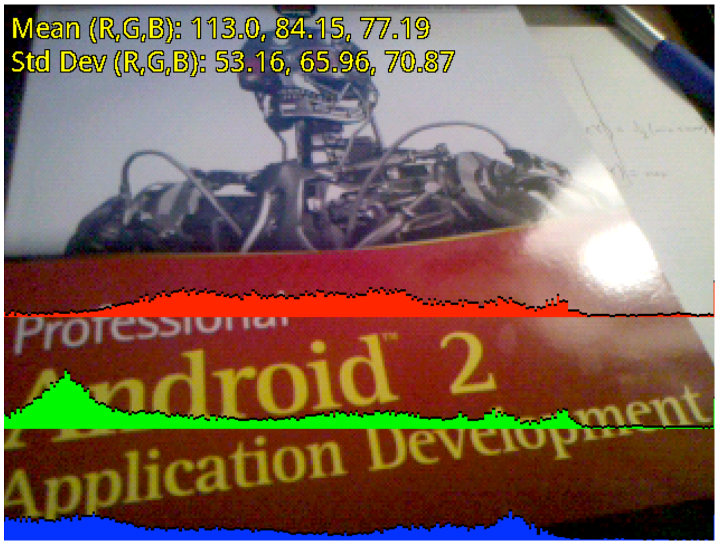
1. Create a new Android project with the following parameters.
   Application name: Viewfinder EE368
   Project name: ViewfinderEE368
   Package name: com.example.viewfinderee368
   Minimum Required SDK: API 8: Android 2.2 (Froyo)
   Target SDK: API 8: Android 2.2 (Froyo)
   Compile With: API 22: Android 5.1
   Theme: None
   Create Activity: Blank Activity

2. Copy the text in the following document into AndroidManifest.xml. Make sure the name of the activity specified in the file matches the name of the activity generated when you created the project, e.g. MainActivity. This defines the main activities and permissions for this program.
   `http://ee368.stanford.edu/Android/ViewfinderEE368/AndroidManifest.xml.txt`

3. Copy the text in the following document into src : com.example.viewfinderee368 : MainActivity.java. This defines the classes in this program.
   `http://ee368.stanford.edu/Android/ViewfinderEE368/MainActivity.java`

4. Check to make sure everything is copied correctly into the project. If there are compilation errors, a red X will appear in the Package Explorer. If you have followed all the previous steps correctly and still get errors, it may be necessary to clean your workspace: Project > Clean > Clean all projects and to rebuild it.

5. Select Run and in the Device Chooser dialog, select your phone. You should see something like Figure 5 on your phone. Point the camera at different objects around you to see how the mean, standard deviation, and histogram of each color channel changes dynamically. You are augmenting the viewfinder in real time!



Mean (R,G,B): 113.0, 84.15, 77.19
Std Dev (R,G,B): 53.16, 65.96, 70.87

Figure 5. "Viewfinder EE368" program running on an Android phone.

6. You may notice that there are many deprecation warnings in the code. This is due to the fact that we compiled our project using Android 5.1, in which most Camera functions are deprecated. One workaround is to suppress the warnings manually. Another possibility is to build the project using another version of the SDK (you can get it on the SDK manager), but this may not work for older APIs – typically API ≤ 13 – since it may not be possible to build the compatibility library with the latest version of the SDK tools.

*Real-time Phone Debugging in Eclipse*

It is actually possible to view real-time messages from the phone in Eclipse, which can be very helpful for debugging and code development.

1. Select Window > Open Perspective > DDMS.

2. A new tab entitled "DDMS" should appear next to the default "Java" tab. Click on the "DDMS" tab.

3. Select your Android device from the list of devices.

4. Select Window> Show View > LogCat. The LogCat view shows a sequential list of real-time messages from the phone. In particular, error messages in red can be very useful when trying to debug a problem.

*Taking a Screenshot of the Phone*

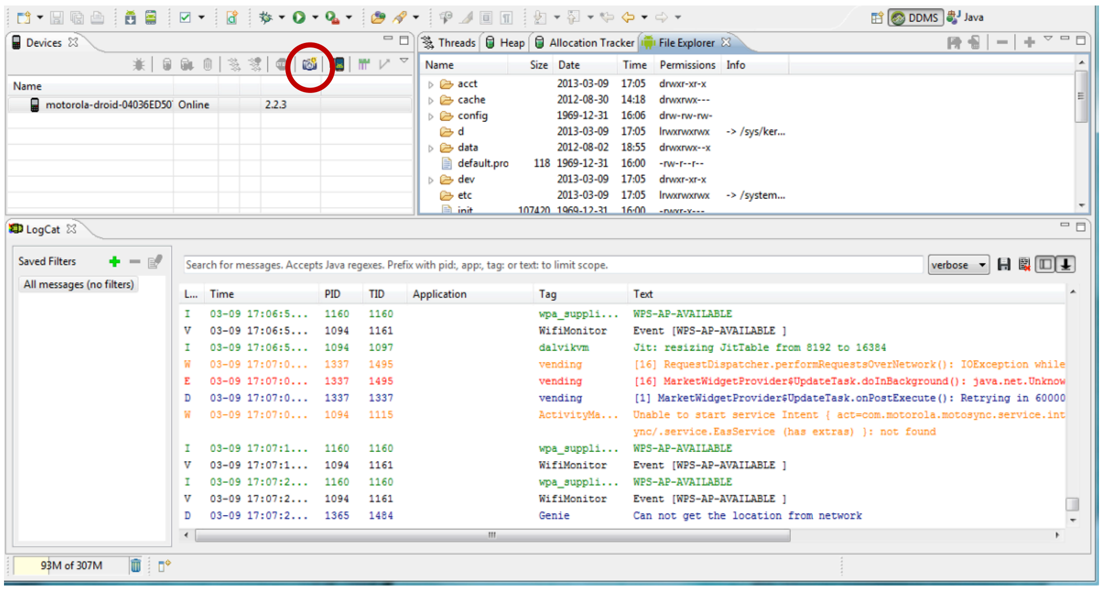
At some point, it may be useful to take a screenshot of the phone, e.g., to use as a figure in your project report.

1. Go to the DDMS perspective in Eclipse.

2. Select your Android device, as shown in Figure 6.

3. Click the camera icon (circled in red in Figure 6), and a panel like Figure 7 should pop up.

4. Finally, when you have the desired screen shown, click Save or Copy to extract the screen shot.



| Name | | | |
|---|---|---|---|
| motorola-droid-04036ED50 | Online | | 2.2.3 |

| Name | Size | Date | Time | Permissions | Info |
|---|---|---|---|---|---|
| acct | | 2013-03-09 | 17:05 | drwxr-xr-x | |
| cache | | 2012-08-30 | 14:18 | drwxrwx--- | |
| config | | 1969-12-31 | 16:06 | drw-rw-rw- | |
| d | | 2013-03-09 | 17:05 | lrwxrwxrwx | -> /sys/ker... |
| data | | 2012-08-02 | 18:55 | drwxrwx--x | |
| default.pro | 118 | 1969-12-31 | 16:00 | -rw-r--r-- | |
| dev | | 2013-03-09 | 17:05 | drwxr-xr-x | |
| etc | | 2013-03-09 | 17:05 | lrwxrwxrwx | -> /system... |
| init | 107420 | 1969-12-31 | 16:00 | -rwxr-x--- | |

Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.

```
L...  Time              PID   TID   Application  Tag            Text
I     03-09 17:06:5...  1160  1160               wpa_suppli...  WPS-AP-AVAILABLE
V     03-09 17:06:5...  1094  1161               WifiMonitor    Event [WPS-AP-AVAILABLE ]
I     03-09 17:06:5...  1094  1097               dalvikvm       Jit: resizing JitTable from 8192 to 16384
W     03-09 17:07:0...  1337  1495               vending        [16] RequestDispatcher.performRequestsOverNetwork(): IOException while
E     03-09 17:07:0...  1337  1495               vending        [16] MarketWidgetProvider$UpdateTask.doInBackground(): java.net.Unknow
D     03-09 17:07:0...  1337  1337               vending        [1] MarketWidgetProvider$UpdateTask.onPostExecute(): Retrying in 60000
W     03-09 17:07:0...  1094  1115               ActivityMa...  Unable to start service Intent { act=com.motorola.motosync.service.int
                                                                ync/.service.EasService (has extras) }: not found
I     03-09 17:07:1...  1160  1160               wpa_suppli...  WPS-AP-AVAILABLE
V     03-09 17:07:1...  1094  1161               WifiMonitor    Event [WPS-AP-AVAILABLE ]
I     03-09 17:07:2...  1160  1160               wpa_suppli...  WPS-AP-AVAILABLE
V     03-09 17:07:2...  1094  1161               WifiMonitor    Event [WPS-AP-AVAILABLE ]
D     03-09 17:07:2...  1365  1484               Genie          Can not get the location from network
```

Figure 6. Dalvik Debug Monitor panel.

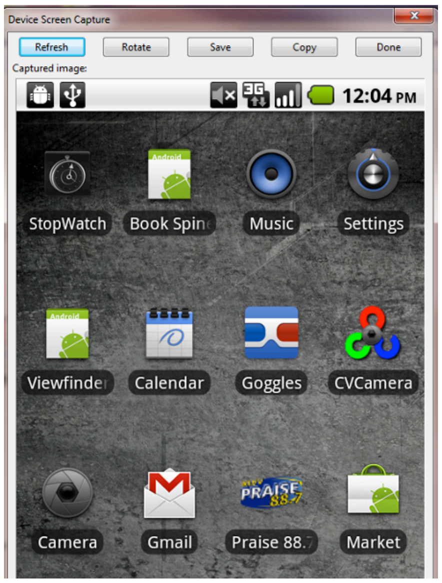


Figure 7. Device screen capture panel.