TECHNOLOGY IN ACTION™

Beginning Android ADK with Arduino

LEARN HOW TO USE THE ANDROID OPEN ACCESSORY DEVELOPMENT KIT TO CREATE AMAZING GADGETS WITH ARDUINO

Mario Böhmer

For your convenience Apress has placed some of the front matter material after the index. Please use the Bookmarks and Contents at a Glance links to access them.

Apress•

Contents at a Glance

CHAPTER 1

Introduction

In May 2011, Google held its annual developer conference, the Google IO, to present its newest technologies to approximately 5,000 attendees. In addition to improvements in its already well-known technologies such as the Google APIs or the core search technology, Google placed the focus on two major themes: Chrome and Android. As always, the newest advances in the Android Platform were presented and discussed, but what Google announced a bit later in the Android keynote was a bit of a surprise: Google's first standard for Android devices to communicate with external hardware. The Android Open Accessory Standard and the Accessory Development Kit (ADK) will be the key for communicating with hardware and building external accessories for Android devices. To encourage development, Google handed out ADK hardware packages to interested attendees and showed some examples of ADK projects, such as a treadmill which transmitted data to a connected Android device and a huge tilt labyrinth, which could be controlled with an Android device. Shortly after the event, the first DIY projects surfaced which already showed the great potential of the ADK.

Since I couldn't attend the event, I had no chance to get my hands on one of those kits; at the time, there was only one distributor for the Google ADK boards and this distributor wasn't prepared for such a big demand. That didn't stop me from building an alternative myself and from experiencing the joy of this new field in Android development. Over time, many more distributors have produced derivatives of the original Google ADK boards, which are, for the most part, cheaper and only provide the basics to get you started hacking your project together.

You probably just want to dive right in, but first you should learn about the specifics of the ADK and set up your development environment. You wouldn't build a house before you knew how to do it or without having the proper tools, would you?

What Is the ADK?

The Accessory Development Kit (ADK) is basically a micro-controller development board that adheres to the simple Open Accessory Standard Protocol created by Google as a reference implementation. Although that could be any board fulfilling the specification to be ADK compatible, most boards are based on the Arduino design, which is an open hardware platform created in 2005. Those boards are USB-enabled micro-controller boards based on the Arduino Mega2560 and the implementation of the Circuits@Home USB Host Shield. However, there are other board designs known to be ADK compatible, such as PIC-based boards or even plain USB host chip boards such as the VNCII by FTDI. Google decided to build its reference kit upon the Arduino Mega2560 design and provided the software and hardware resources as open source. This was a clever move because the Arduino community has grown tremendously over the last years, enabling designers, hobbyists, and average Joes to easily make their ideas come to life. With the ever-growing communities of both factions of Android and Arduino enthusiasts, the ADK had a pretty good start.

To communicate with the hardware boards, an Android-enabled device needs to fulfill certain criteria. With Android Honeycomb version 3.1 and backported version 2.3.4, the necessary software APIs were introduced. However, the devices also have to ship with a suitable USB driver. This driver enables general USB functionality but, in particular, it enables the so-called accessory mode. The accessory mode allows an Android device that has no USB host capabilities to communicate with external hardware, which in turn acts as the USB host part.

The specification of the Open Accessory Standard stipulates that the USB host has to provide power for the USB bus and can enumerate connected devices. The external device has to provide 500mA at 5V for charging purposes of the Android device according to the USB 2.0 specification.

The ADK also provides firmware for the development board which comes in the form of a set of source code files, libraries, and a *demokit sketch*, which is the Arduino term for a project or source code file. The firmware cares about the enumeration of the USB bus and finding a connected device that is accessory mode–compatible.

Google also provides an example app for the Android device that easily accesses and demonstrates the capabilities of the reference board and its sensors and actuators. If you are working with a derivative board that doesn't have the same variety of sensors, you still can work with the example app, but you might want to strip the code down to only the basic part of the communication.

When you set up an ADK hardware project you are building a so-called *Android accessory*. Your hardware project is an accessory for the Android device such as, for example, a keyboard would be for a PC, with the difference being that your accessory provides the power for the whole system. Accessories need to support the already mentioned power supply for the device and they must adhere to the Android accessory protocol. The protocol dictates that the accessory follows four basic steps to establish a communication to the Android device:

1. The accessory is in wait state and tries to detect any connected devices.
2. The accessory checks for accessory mode support of the device.
3. The accessory tries to set the device in accessory mode if it is necessary.
4. If the device supports the Android accessory protocol, the accessory establishes the communication.

If you want to learn more about the ADK and the Open Accessory Standard have a look at the Android developer pages at `http://developer.android.com/guide/topics/usb/adk.html`.

Hardware Development Boards

This section will give you an overview of the variety of ADK-compatible development boards that are currently on the market. Note that I can't guarantee the completeness of this list because the community advances at such a pace that new boards could pop up at any time. I will concentrate on the most popular boards out there as of this writing.

The Google ADK

The Google ADK is the reference kit presented at the Google IO in May 2011 and it was the first board adhering to the Open Accessory Standard. The kit comes with the ADK base board and a demo shield, as shown in Figure 1-1.

Figure 1-1. *Google ADK board and Demo Shield*

The base board (Figure 1-2) contains the DC power connector, the USB connector (A-type receptacle) to connect your phone or tablet to, and the micro USB connector (micro B-type receptacle) to connect to your computer for programming and debugging purposes. It has an ATmega2560 AVR chip from Atmel mounted on top, optimized for C-compiled code, which makes it pretty fast and easily programmable instead of comparable microcontrollers that have to be programmed in the assembler language. The ATmega2560 has an internal flash memory of 256 Kbytes and an 8-bit CPU and it operates at 16MHz. It provides 8KB of SRAM and 4KB of EEPROM. The IO ports of the ATmega chip control 16 analog pins that provide 10 bits of input resolution enabling analog-to-digital conversion of 1,024 different values. They measure from ground to 5V by default. The chip has 54 digital pins with 14 of them being PWM (pulse width modulation) enabled to allow, for example, dimming of LEDs or controlling servos. In the middle of the board is a reset button to reset the program execution on the board. The board's operating voltage is 5V. Although you can power the board via a USB cable, you should consider using a power adapter if you intend to control servos or drive motors.

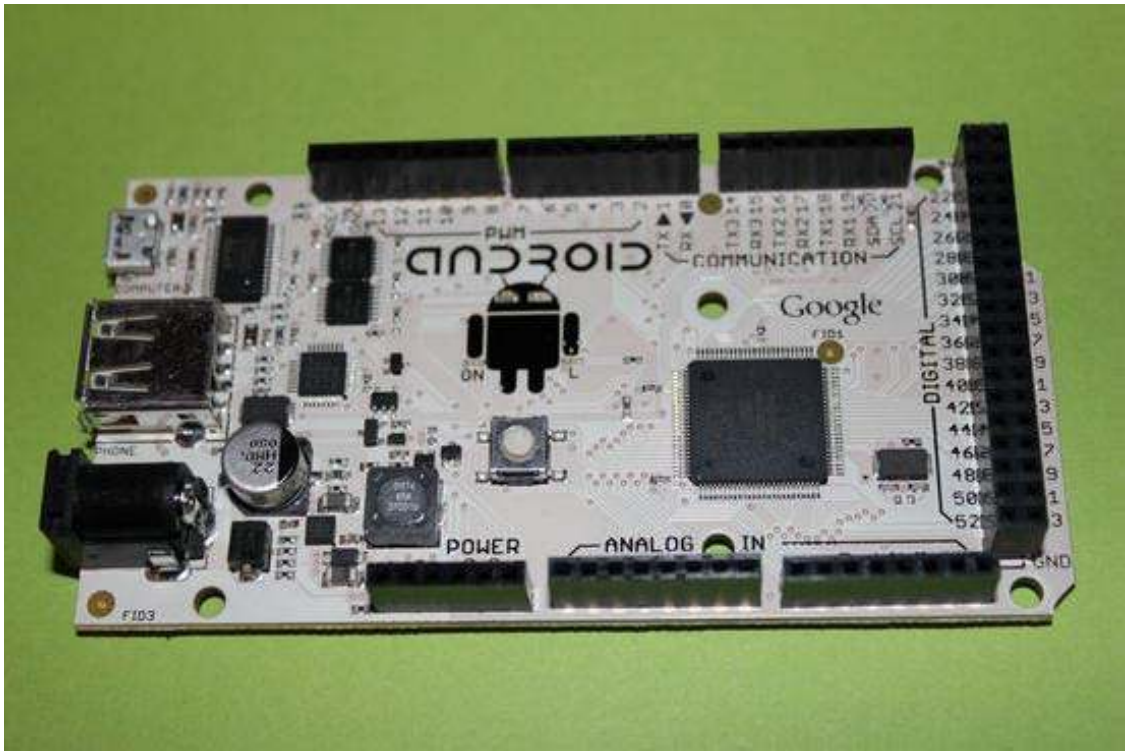

Figure 1-2. *A closer look at the Google ADK board*

The *Demo Shield* is an additional board containing a broad variety of different sensors and actuators. *Shield* is an Arduino term for an extension board that can be put on top of an Arduino base board. The connection is made via stackable pin headers. The IO pins of the base board are mostly delegated to the pins of the shield so they can be reused. However, certain shields might occupy pins to operate their sensors. The demo shield itself is presoldered with male pin headers so no additional shields can be stacked on top. This doesn't come as a surprise, since the shield uses most of the pins to let the base board communicate with all of its sensors. Since the shield hides the reset button of the base board, it contains one itself so that you can still make use of the reset functionality. The most important parts, however, are the sensors and actuators and there are a lot of them.

- One analog joystick
- Three buttons
- Three RGB LEDs
- A transistor functioning as a temperature sensor
- An IC with an integrated photo diode for light sensing

- A capacitive touch area in the form of the Android logo
- Two relays with screw terminals which can switch external circuits with 24V up to 1A
- Three servo connectors

The Google ADK was originally produced by a Japanese company for the Google IO. It can be ordered at `www.rt-net.jp/shop/index.php?main_page=product_info&cPath=3_4&products_id=1`. At a price of approximately $400 (not including sales tax), it is one of the priciest boards out there.

The Arduino ADK

The Arduino ADK (Figure 1-3) is an ADK-compatible base board from the makers of the Arduino series themselves. It is also based on the ATmega2560 and only differs slightly from the Google reference board.

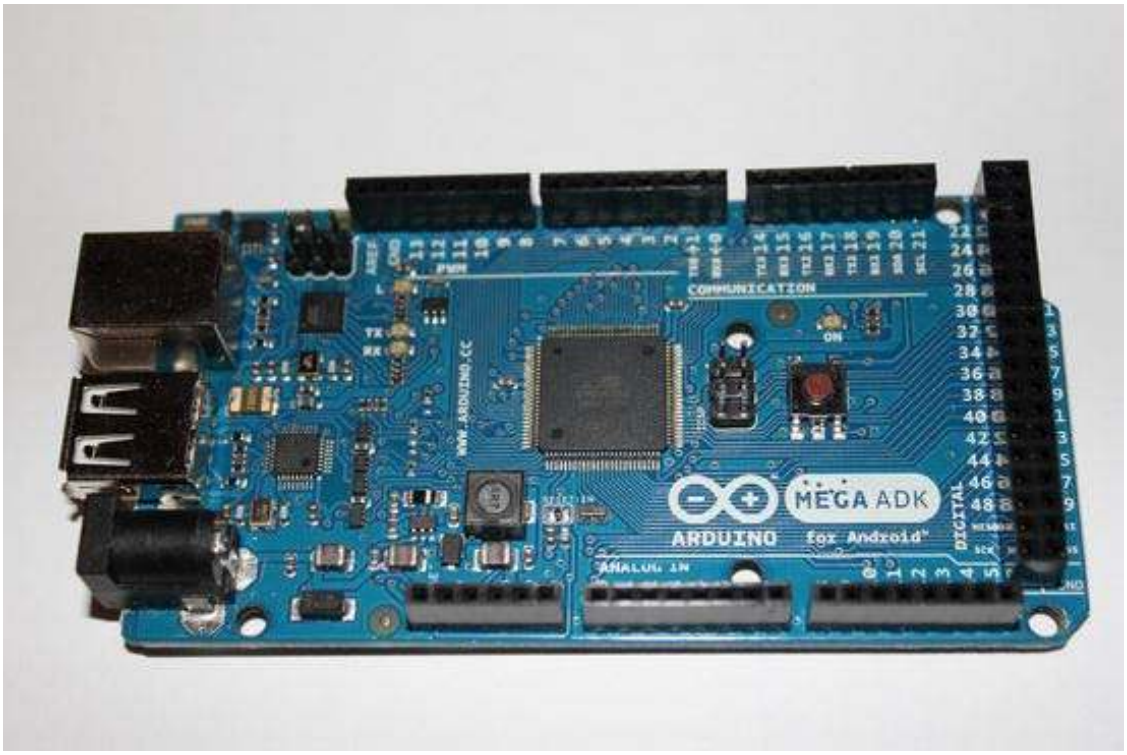

Figure 1-3. *Arduino ADK board*

The Arduino ADK board also has a DC power connector and a USB connector (A-type receptacle) mounted to connect to an Android device. The programming and debugging connector, however, differs in being a standard USB connector (B-type receptacle). The reset button is situated at the far end of the board and the ATmega chip sits in the middle of the board. The IO pin layout is exactly the same as in

the Google board and it has the same analog and digital pin characteristics. The Arduino ADK, however, has two ICSP 6-pin headers for In-Circuit Serial Programming (ICSP) of microchips. Sharing the same pin layout and form factor, the Arduino ADK and the Google ADK are compatible with the Demo Shield and other Arduino based shields.

The Arduino ADK is made in Italy and can be ordered directly from the Arduino site at `http://store.arduino.cc/ww/index.php?main_page=product_info&cPath=11_12&products_id=144` or from one of its numerous distributors worldwide found at `http://arduino.cc/en/Main/Buy`.

At a price of about $90 (not including possible shipping costs and taxes), it is way more affordable than the Google ADK for the average hobbyist and hardware hacker.

The IOIO

The IOIO (pronounced yo-yo) board (Figure 1-4) is a PIC micro-controller–based development board developed by Sparkfun Electronics before the announcement of the Open Accessory Standard.

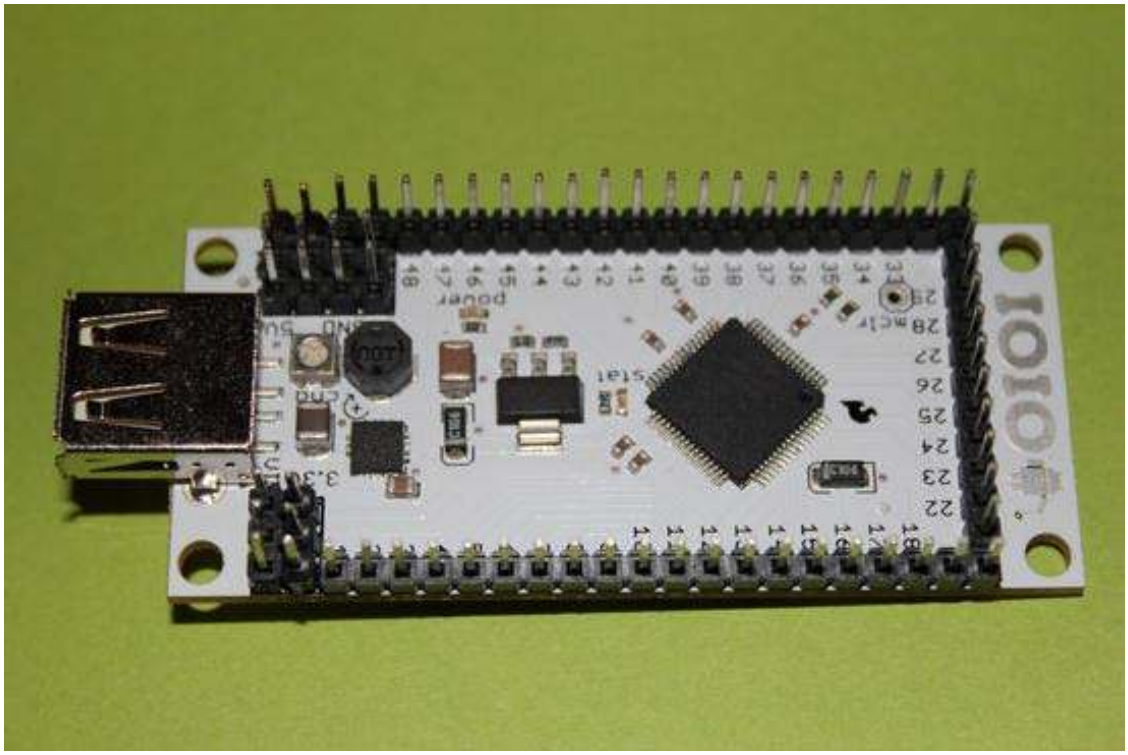

Figure 1-4. *Sparkfun IOIO board*

The IOIO board was designed to work with all Android devices with version 1.5 and above. The original firmware design was targeted to work with the Android Debug Bridge (ADB), which is normally used within the development process of an Android application for debugging processes and for file system operations. After the announcement of the Open Accessory Standard the IOIO was updated with a new firmware to support both the Open Accessory Protocol and, as a fallback, the ADB protocol to still

support older devices. The firmware is still in beta as of the time of writing this book. Since you need to update the firmware of the board through a PIC programmer in order to make the board ADK compatible, it might not be the perfect choice for an inexperienced tinkerer.

The hardware specifics of the board are as follows. The IOIO has a form factor of about a quarter of the size of a regular ADK-compatible board, which makes it one of the smallest boards available. Nevertheless, it nearly keeps up with the numerous IO pins of its big brothers. Many of the overall 48 IO pins have several operating modes, which can make the pin assignments a bit confusing.

From the 48 IO pins, all pins can be used as general purpose input output ports. Additionally, 16 of those pins can be used as analog inputs, 3 pairs of pins can be used for I²C communication, 1 pin can be used as a peripheral input, and 28 pins can be used for peripheral inputs and outputs. Normally, the pins are 3.3V tolerant only, but 22 pins are capable of tolerating 5V inputs and outputs. The I²C pins provide a fast and simple two-wire interface to communicate with external integrated circuits such as sensor boards.

Apart from the IO pins the board provides 3 Vin pins for power supply of the board. On the bottom side of the board you can solder an additional JST connector to connect a LiPo battery as the power supply. An operating voltage of 5V to 15V should be supplied. Additionally, it has 3 pins for 3.3V output, 3 pins for 5V output, and a 9 pin-area for ground.

The only connector on this board is the required USB (A-type receptacle) connector. That is because programming the hardware is not necessary, unlike for the other ADK-compatible boards, which need C-compiled code for the hardware part. The IOIO provides a firmware that implements all necessities. You only need to write the Android part by using a high-level API for easy pin access.

One interesting component of the board is a small trimmer potentiometer that can limit the charging current of the Android device so that it won't draw too much power when the board is in battery mode. The IOIO has a PIC micro-controller chip instead of the AVR chip most of the other boards use. The PIC24FJ256-DA206 chip operates at 32MHz, has 256KB of programmable memory and 96KB of RAM.

The IOIO was developed by Sparkfun Electronics and can be ordered via the Sparkfun web site at `www.sparkfun.com/products/10748` or through one of its distributors.

With a price of about $50 before shipping and taxes, it is one of the cheapest boards out there but not one of the friendliest to beginners.

The Seeeduino ADK Main Board

The Seeeduino ADK board (Figure 1-5), also derived from the ATmega board, looks quite similar to the standard Arduino ADK board but, at second glance, it has some nice extra features.

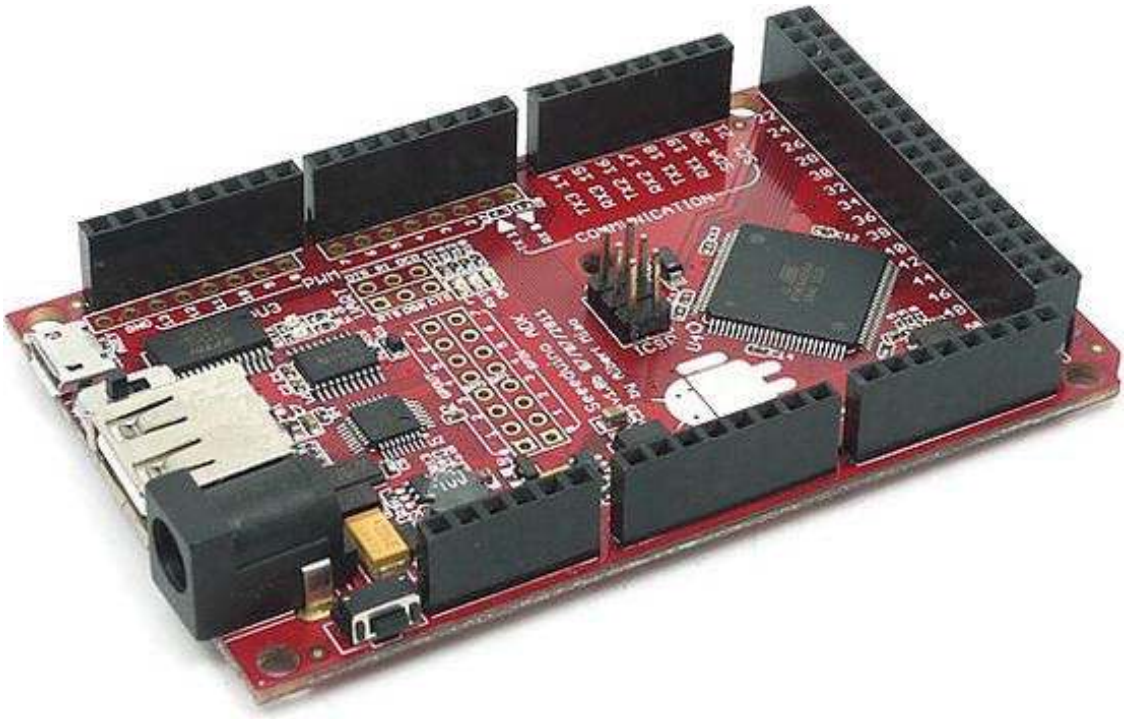

Figure 1-5. *Seeeduino ADK board (image courtesy of Seeedstudio)*

It has 56 digital IO pins with 14 of them being PWM capable, 16 analog input pins, and 1 ICSP header. The connectors on the board are of the same type as in the original Google design. It has a DC power connector, a USB connector (A-type receptacle), and a micro USB connector (micro B-type receptacle).

The biggest difference with most other Atmega-like boards is that the Seeeduino ADK board already ships with the MicroBridge firmware so that it works in ADK mode with Android devices with OS version 2.3.4 and above and in ADB mode with devices that have OS versions previous to version 2.3.4, much like the IOIO does.

The Seeeduino ADK board was developed by Seeedstudio and can be ordered at the company's web site at `www.seeedstudio.com/depot/seeeduino-adk-main-board-p-846.html` or from one of their distributors.

It is priced at $79 (before shipping and taxes), which makes it a very affordable but powerful board.

More ADK Possibilities

After you have seen the most common boards with ADK support out there, you'll probably wonder if that's all there is. Although the Open Accessory Standard is only about a year old, the number of boards already available is incredible, with many still to come in this young but rapidly evolving field of open source hardware. There are still plenty of other possibilities for developing with the Open Accessory Standard. Some represent pure DIY (do-it-yourself) approaches, while others are extensions for boards that have been in use since before the ADK came out.

One early approach was to port the ADK to the common Arduino Uno or Duemilanove. The only thing you needed was an additional USB host shield to connect the Android device to. I was one of those early DIY hackers who went in that direction. At the time, it was the only affordable alternative to the original Google reference board. Nowadays, I wouldn't recommend it; there are already perfect all-in-one boards that don't need additional shields, hacking, or stripping of code. If you still want to use your regular Arduino there are a lot of shops carrying USB host shields you can use:

- www.circuitsathome.com/products-page/arduino-shields/usb-host-shield-2-0-for-arduino/
- www.sparkfun.com/products/9947
- www.dfrobot.com/index.php?route=product/product&filter_name=usb%20host%20shield&product_id=498
- http://emartee.com/product/42089/Arduino%20ADK%20Shield%20For%20Android

You may have read about the possibility of enabling communication with Android devices running an OS version lower than 2.3.4, which some boards provide. If also you want to support that in your projects you should have a look at the microbridge project that uses the ADB to establish the communication. Check the project page for further details at http://code.google.com/p/microbridge/.

Some of the all-in-one boards also come bundled as a kit to let you tinker away with a bunch of sensors. These kits usually provide some of the same sensors that the Google Demo Shield features.

The Arduino store sells an ADK Sensor kit that consists of an Arduino ADK Mega board with a Mega Sensor Shield. The sensor shield has 22 3-pin connectors to easily connect sensor modules without having to worry about wiring and setup. For more information go to http://store.arduino.cc/eu/index.php?main_page=product_info&cPath=11&products_id=140.

SeeedStudio also has a kit called Grove ADK Dash Kit. Like the Arduino kit, it also provides an easy plug-and-play mechanism to start right away and it features a broad set of sensors for all kinds of purposes. It is available at http://www.seeedstudio.com/depot/grove-adk-dash-kit-p-929.html.

If you still want a kit based on the original Google design but importing the Japanese original is not an option, you can also consider the following German clone, which is nearly an exact clone with the minor improvement of providing a gold-plated touch area that has better conductivity and less oxidation. It is also a bit more affordable than the original and, depending on where you live, the shipping costs may be lower. Check out www.troido.de/de/shoplsmallgbuy-android-stuffsmallg/produkte/view/1/1 for more information.

Which Board Should You Use?

Now that you have read about the variety of boards supporting the Open Accessory Standard that are already out there you might wonder which board is the right one for your own project. This is always a hard question, for which there is no single answer. You should plan your project thoroughly ahead of time to analyze which board fits best.

If you are a beginner in the world of hardware development and ADK, you should stick to the boards that are most commonly used out in the wild. As of this writing, that would be the Google ADK board, which was given out to hundreds of developers attending the Google IO 2011. If you are not one of the lucky ones to have received one of these boards and your budget is pretty tight—which is usually the case—consider the standard Arduino ADK board. Both of these boards are used in most hacker and maker projects I have seen so far and they have a huge community built around them to help you if you are in need.

Table 1-1 gives you an overview of the boards under discussion.

Table 1-1. *Comparison of the Most Common ADK-Enabled Boards*

ADK Boards	Google ADK	Arduino ADK	Seeeduino ADK	Sparkfun IOIO
Processor	ATmega2560 ATmega	2560	ATmega2560 PIC	24FJ256
CPU clock speed	16 MHz	16 MHz	16 MHz	32 MHz
Flash memory	256 Kbytes	256 Kbytes	256 Kbytes	256 Kbytes
RAM	8 Kbytes	8 Kbytes	8 Kbytes	96 Kbytes
Digital IO pins	54 (14 PWM)	54 (14 PWM)	56 (14 PWM)	48 (28 PWM)
Analog input pins	16	16	16	16
Input voltage	5.5V - 16V	5.5V - 16V	6V - 18V	5V - 15V
Connectors DC	power	DC power	DC power	USB A-type
	USB A-type	USB A-type	USB A-type	
	USB micro B-type	USB B-type	USB micro B-type	

Supported Android Devices

The Open Accessory Standard was introduced as part of the Android API in Android Honeycomb version 3.1 with the rollout of more and more Android-enabled tablets. To not only support Honeycomb devices, Google decided to backport the necessary classes to version 2.3.4 making them available for phones also. The newer functionality was backported as a Google API add-on library. This library is basically a JAR file that has to be included in the build path of your Android project.

The first candidates to have received the necessary version updates and which supported the Open Accessory mode were the Motorola Xoom and the Google Nexus S. Other devices were soon to follow, which quickly led to the well-known problem of fragmentation. Normally, fragmentation is mostly a problem when it comes to different versions of the operating system, but now the problem was that even though a device had the necessary OS version of 2.3.4 or 3.1, it was still possible that the Open Accessory mode wouldn't work on the device. How could that happen? Well, the problem was that it is not sufficient to update only the system software. The USB drivers of the device must be compatible with the Open Accessory mode. Many developers updated their device or even rooted it to install a homebrew mod like Cyanogen Mod to finally run version 2.3.4, only to find that the USB drivers of the device manufacturer weren't compatible.

However, there are a lot of devices that have been tested and are said to work perfectly with the Open Accessory mode, some of them officially, others driven by DIY mods. Here is a list of some devices that have been verified by the community to work with the ADK:

- Google Nexus S
- Google Nexus One

- Motorola Xoom
- Acer Iconia A100
- Acer Iconia A500
- LG Optimus Pad
- ASUS Eee Pad Transfomer TF101
- Samsung Galaxy Tab 10.1
- Samsung Galaxy S
- Samsung Galaxy Ace

Personally, I would recommend using a Google developer device like the Nexus S, as those devices have the best support for the newest APIs and functionalities.

Setting Up the Development Environment

You know a lot about the history of the ADK and the technical specifics, but before your ideas can come to life you need to set up your working environment. You will need to program software for your Android device as well as for your hardware board to let both parties communicate with each other and to control actuators or read sensor values. The programming is done with the help of two Integrated Development Environments (IDEs). To program Android applications, Google recommends using the Eclipse IDE. The Eclipse IDE is the most common IDE for Java development with one of the biggest communities, a variety of plugins, and excellent support. Since the hardware boards are based on the Arduino design, you will program them with the Arduino IDE to write so-called *sketches* that will be uploaded to the board. In order for those IDEs to work properly you also need the Java Development Kit (JDK), which has more functionality than the normal Java Runtime Environment (JRE) and which your system probably has already installed. You will also need the Android SDK to write your Android applications.

This step-by-step guide will help you to set up the necessary development environment. Please follow the steps for your operating system of choice exactly. If you run into any problems you can also refer to the official installation guides on the software sites.

The Java Development Kit

The first thing you will need is the JDK. Go to `www.oracle.com/technetwork/java/javase/downloads/index.html` and click the JDK download button (Figure 1-6).

Figure 1-6. *JDK download page*

Accept the license agreement and choose the file for your operating system (Figure 1-7). The x86 files are suitable for 32-bit operating systems and the x64 files have to be installed on 64-bit systems, so make sure to select the correct one.

Java SE Development Kit 7

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ Accept License Agreement ◉ Decline License Agreement

Product / File Description	File Size	Download
Linux x86 - RPM Installer	77.28 MB	jdk-7-linux-i586.rpm
Linux x86 - Compressed Binary	92.17 MB	jdk-7-linux-i586.tar.gz
Linux x64 - RPM Installer	77.91 MB	jdk-7-linux-x64.rpm
Linux x64 - Compressed Binary	90.57 MB	jdk-7-linux-x64.tar.gz
Solaris x86 - Compressed Packages	154.74 MB	jdk-7-solaris-i586.tar.Z
Solaris x86 - Compressed Binary	94.75 MB	jdk-7-solaris-i586.tar.gz
Solaris SPARC - Compressed Packages	157.81 MB	jdk-7-solaris-sparc.tar.Z
Solaris SPARC - Compressed Binary	99.48 MB	jdk-7-solaris-sparc.tar.gz
Solaris SPARC 64-bit - Compressed Packages	16.28 MB	jdk-7-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit - Compressed Binary	12.38 MB	jdk-7-solaris-sparcv9.tar.gz
Solaris x64 - Compressed Packages	14.66 MB	jdk-7-solaris-x64.tar.Z
Solaris x64 - Compressed Binary	9.39 MB	jdk-7-solaris-x64.tar.gz
Windows x86	79.48 MB	jdk-7-windows-i586.exe
Windows x64	80.25 MB	jdk-7-windows-x64.exe

Figure 1-7. *JDK platform downloads*

You may notice that there are no JDK files for Mac OS. Those are not distributed over the Oracle site because Apple provides its own version of the JDK. The JDK should come preinstalled on your Mac OS X system. You can verify that by typing `java -version` into a terminal window. You should see your currently installed Java version listed in the terminal window.

Installing on Windows

After you have downloaded the executable, open it and follow the instructions that will guide you through the installation process. Afterward, you should set the JDK path to your Windows `PATH` variable. The path variable is used to conveniently run executables from anywhere in your system. Otherwise, you would always have to type the complete path in order to execute something from the command line such as `C:\Program Files\Java\jdk1.7.0\bin\java`.

Eclipse also depends on the `JAVA_HOME` variable to be set. To set system environment variables you will have to do the following.

1. Right-click `My Computer` and select Properties as shown in Figure 1-8.

Figure 1-8. *Open System Properties dialog*

2. On the system Properties dialog select the `Advanced` tab. Near the bottom you should see the `Environment Variables` button (Figure 1-9), which opens the variables dialog to set User and System variables.

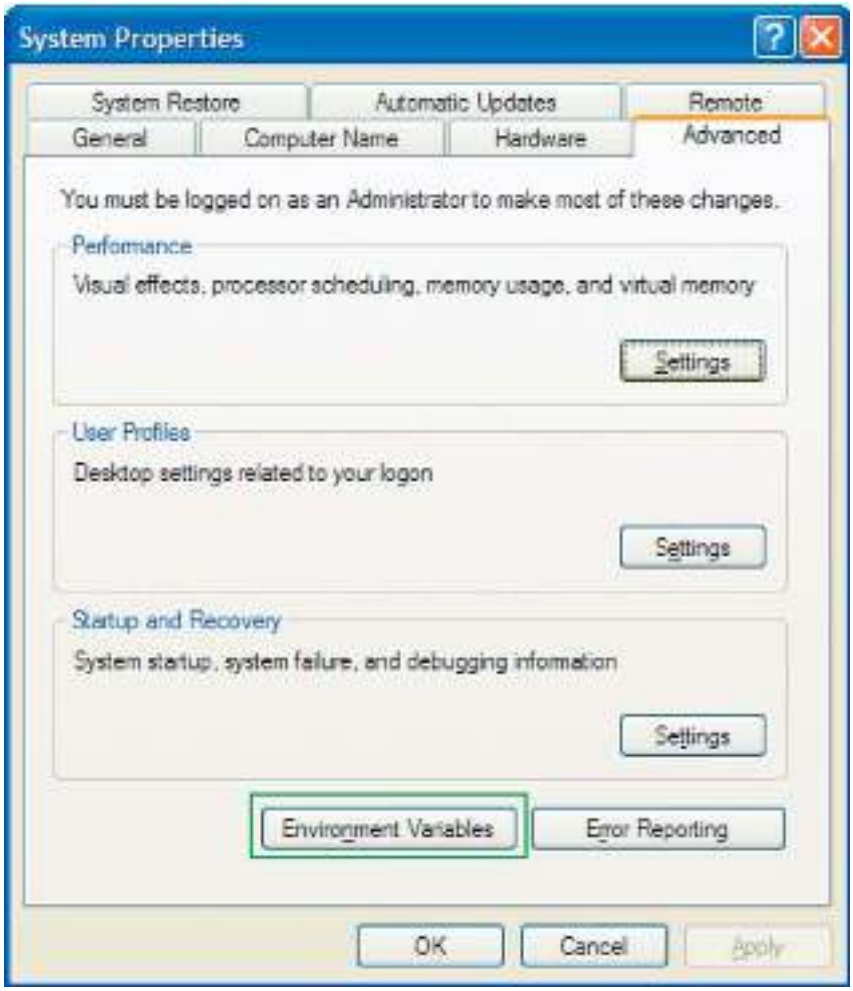

Figure 1-9. *Open Environment Variables dialog*

3. Click New in the System Variables area and insert the following:

Variable Name: `JAVA_HOME`
Variable Value: `C:\Program Files\Java\jdk1.7.0`

4. Click `OK`.

5. Additionally edit the `PATH` variable by selecting it (Figure 1-10) and click `Edit`. Insert `%JAVA_HOME%/bin;` in front of the other values.

6. Click `OK`.

7. Now your JDK is set up and ready for work.

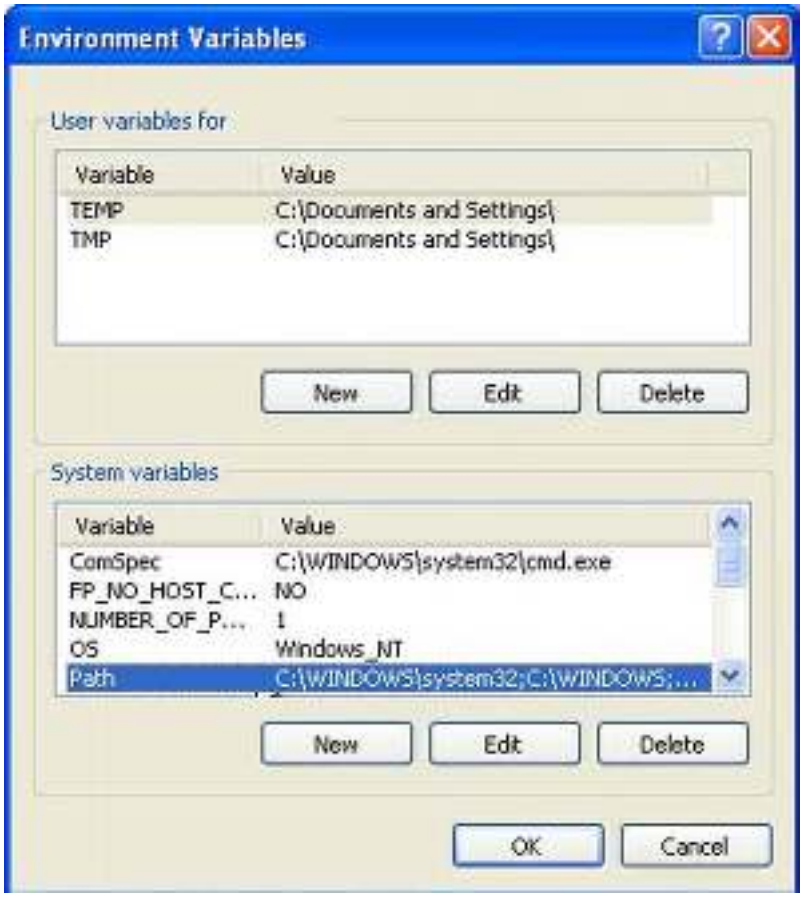

Figure 1-10. *Setting up Environment Variables*

Installing on Linux

Download the `tar.gz` file for your system (32-bit / 64-bit) and move the file to the location where you want the JDK to be installed. Usually the JDK is installed into `/usr/java/` but keep in mind that you need root permissions to install into that directory.

Unpack and install the JDK with:

```
# tar zxvf jdk-7-linux-i586.tar.gz
```

The JDK is installed in the `/jdk1.7.0` directory within your current directory.

To let your system know where you have installed the JDK and to be able to run it from anywhere in your system, you have to set the necessary environment variables. For that purpose, it is a good idea to create a short shell script which is placed in the `/etc/profile.d` directory.

Create a script called `java_env.sh` in that directory and add the following content to the script:

```
#!/bin/bash

JAVA_HOME=/usr/java/jdk1.7.0

PATH=$JAVA_HOME/bin:$PATH

export PATH JAVA_HOME
export CLASSPATH=.
```

The last thing to do is to set the permission on the newly created script so that the system will execute it at user login.

```
# chmod 755 java_env.sh
```

After a fresh login the environment variables will be set.

Installing on Mac OS X

As mentioned before, the JDK for Mac is not distributed via the Oracle download site. The JDK is preinstalled on Mac OS X but can also be downloaded through the Apple Store. If your environment variable for `JAVA_HOME` and `PATH` is not already set you can refer to the according steps used in the Linux installation, as Mac OS X is also Unix-based. You can check if the variables are set using the following command in a terminal window:

```
# echo $JAVA_HOME
```

And similar for the `PATH` variable:

```
# echo $PATH
```

The Android SDK

To be able to write Android applications you need the Android Software Development Kit, which provides all libraries and tools for all Android versions that are supported by Google right now.

You can download the SDK from the Android developer pages (Figure 1-11) at `http://developer.android.com/sdk/index.html`.

Download the Android SDK

Welcome Developers! If you are new to the Android SDK, please read the steps below, for an overview of how to set up the SDK.

If you're already using the Android SDK, you should update to the latest tools or platform using the *Android SDK and AVD Manager*, rather than downloading a new SDK starter package. See Adding SDK Components.

Platform	Package	Size	MD5 Checksum
Windows	android-sdk_r12-windows.zip	36486190 bytes	8d6c104a34cd2577c5506c55d981aebf
	installer_r12-windows.exe (Recommended)	36531492 bytes	367f0ed4ecd70aefc290d1f7dcb578ab
Mac OS X (intel)	android-sdk_r12-mac_x86.zip	30231118 bytes	341544e4572b4b1afab123ab817086e7
Linux (i386)	android-sdk_r12-linux_x86.tgz	30034243 bytes	f8485275a8dee3d1929936ed538ee99a

Figure 1-11. *Android SDK download page*

The site provides zipped archives of the SDK for Windows, Linux, and Mac. There is also another version for Windows, which is an executable and should guide you through the installation process. Since the initial setup is the same for all platforms, there will be no OS-specific steps.

After you have downloaded the SDK archive move it to a location of your choice and unzip it. You will see that both the `add-ons` and `platforms` directories are empty. Only the `tools` directory contains several binaries. The important file within that directory right now is the `android` script. It launches the SDK and AVD Manager (Figure 1-12).

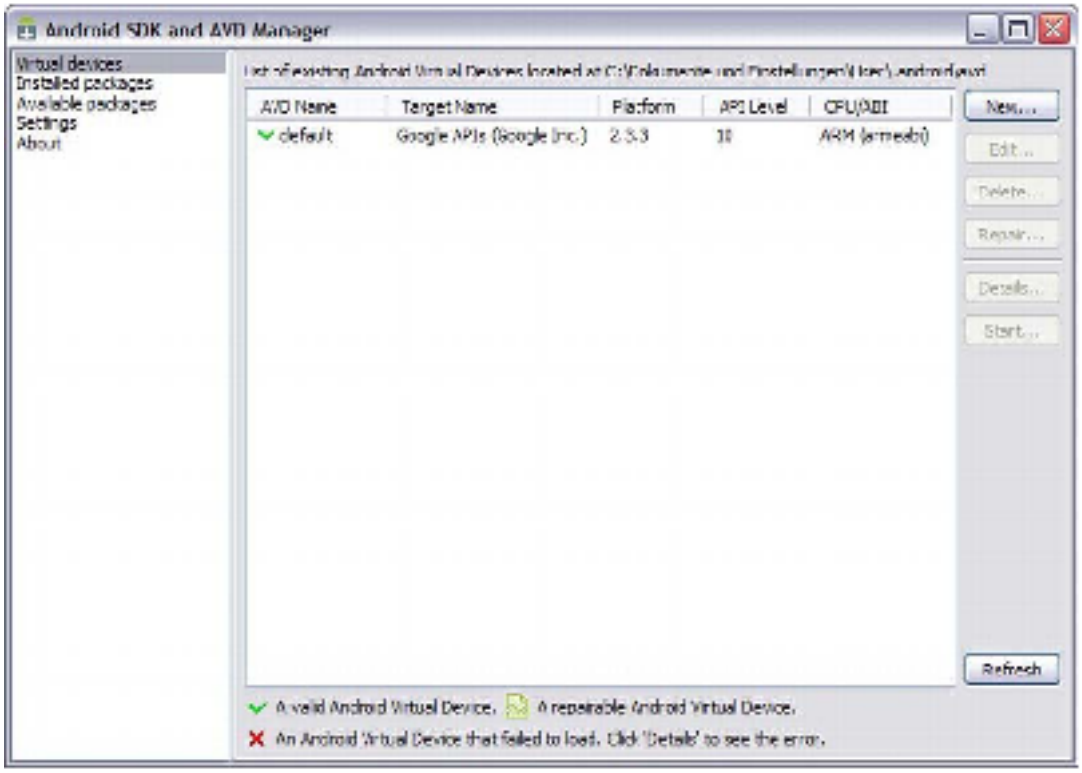

Figure 1-12. SDK Manager

The SDK Manager is the central core of the SDK. It manages the installed Android versions and it updates new versions and additional packages. With the SDK and AVD manager you can also set up emulators for testing Android applications.

At first startup it will connect to the Google servers to check for version updates and it will prompt you afterwards to install SDK packages (Figure 1-13). It is a good idea to just click Accept All and install to install all SDK versions and additional packages. This can take a really long time depending on your Internet connection. If you want to only install the packages that are absolutely necessary, just accept the following packages and reject the others:

- Android SDK Platform-tools
- Documentation for Android SDK
- SDK Platform for Android 2.3.3, API 10 and all newer ones
- Samples for SDK API 10 and all newer ones
- Google APIs by Google Inc., Android API 10 and all newer ones
- Google USB Driver package
- Android Compatibility package

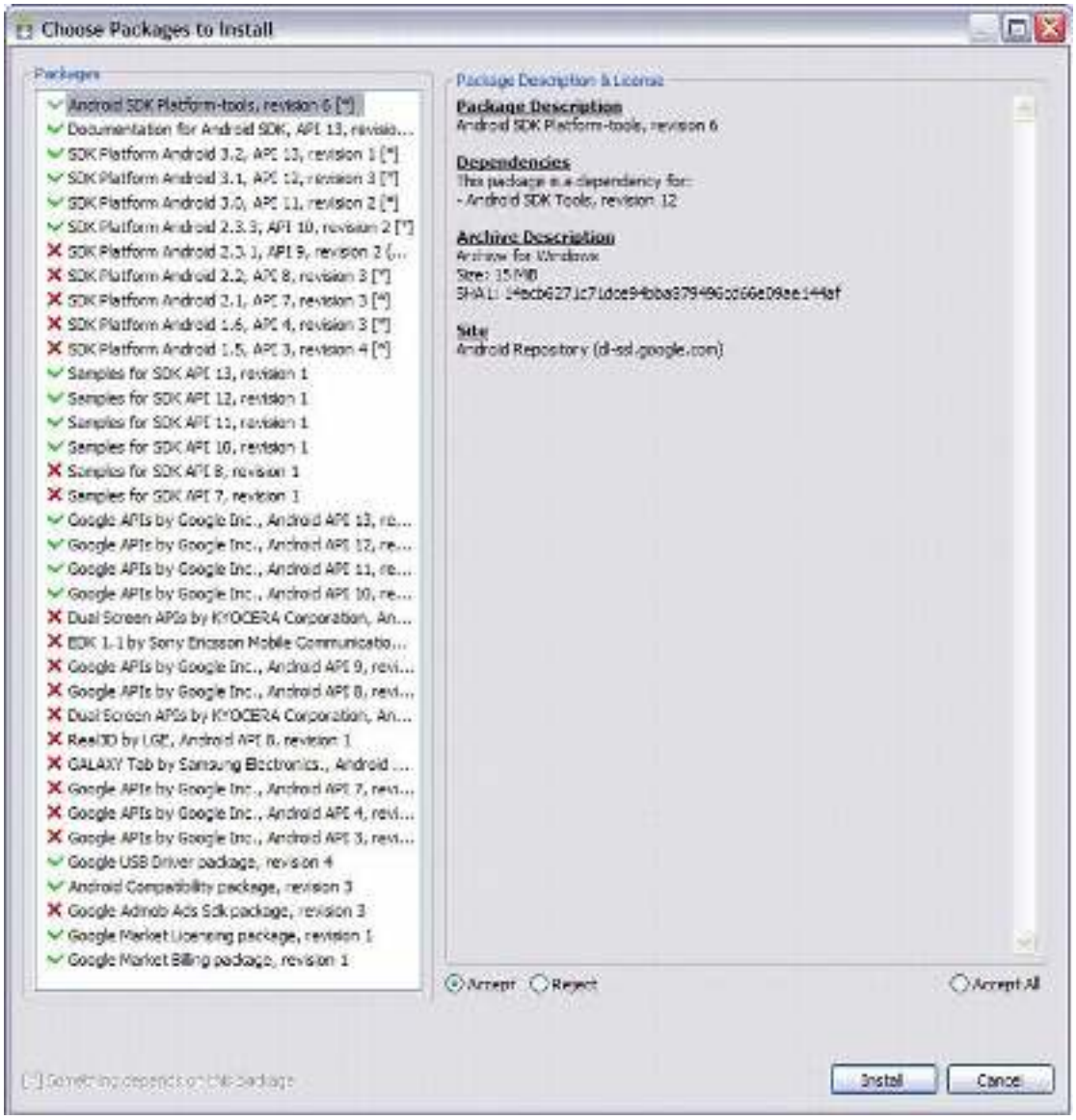

Figure 1-13. *SDK package installation*

You can uninstall and install the packages at any time by clicking installed and available packages to manage your SDK installation. When the SDK Manager is finished downloading all the necessary packages you will see that your SDK directory has grown and has some new folders in it. I will talk about some of them later, so there is no need to understand them now. The SDK is now ready for development.

The Eclipse IDE

The Eclipse Integrated Development Environment is one of the most commonly-used IDEs among software developers. It has a huge supportive community and some the most powerful plugins for all kinds of development scenarios. You will need Eclipse for developing your Android applications. The programming language for writing Android applications is Java. Though you are writing Java code, ultimately it will be compiled into Android-specific dex code and packaged into an archive file with the file ending `.apk`. This archive file will be put onto your Android device for installation.

To download Eclipse, go to `www.eclipse.org/downloads/`. Select your operating system in the upper-right corner of the distribution list (Figure 1-14).

Figure 1-14. *Eclipse download site*

You should choose the Eclipse Classic edition as it is not preconfigured for other purposes. Click the download button for your system type (32-bit/64-bit). Now select the default mirror page for the download or select a mirror nearest to you (Figure 1-15).

Figure 1-15. *Download mirror selection*