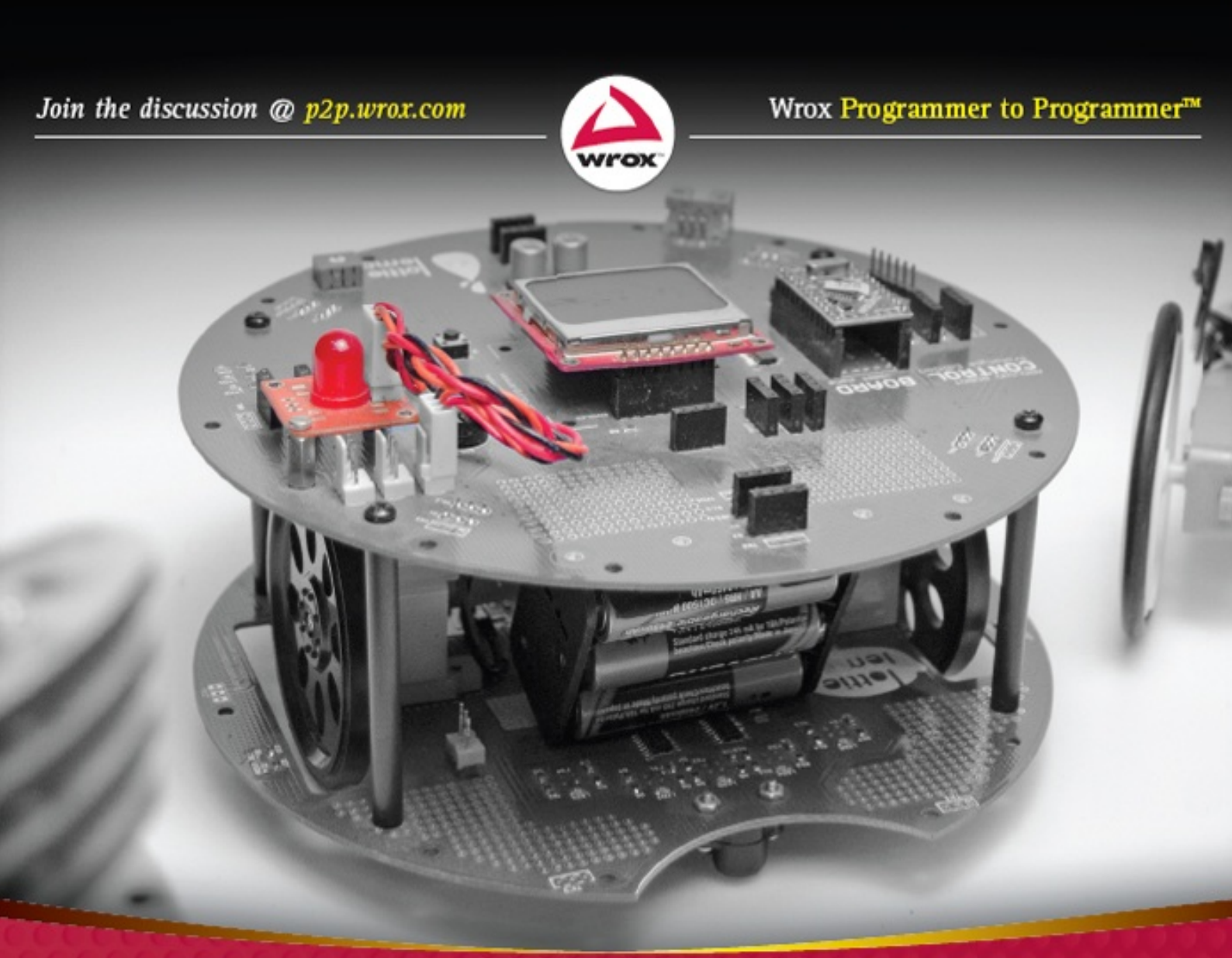
Join the discussion @ p2p.wrox.com



Wrox Programmer to Programmer™

Professional Android™ Open Accessory Programming with Arduino™

Andreas Göransson, David Cuartielles Ruiz

CONTENTS

Part I

Welcome to the Wonderful World of Accessories

Chapter 1

Introduction to Android Open Accessory

WHAT'S IN THIS CHAPTER?

- Introduction to the Android Open Accessory standard
- Getting to know the Arduino project
- Understanding the Open Hardware culture

If you ask your colleagues what Android really is, you will probably hear something about Linux, Jav Virtual Machines (JVMs), or various devices; you might even hear some statistical reports on mark shares of Android in comparison to other mobile operating systems.

We would rather introduce Android as a way to explore the world of connected devices. This is, i essence, what Android Open Accessory (AOA) is all about — making your Android phone connect t and communicate with, any other device around it!

In this chapter you get a background and overview of the Android project, the Android Ope Accessory framework, and the electronics platform called Arduino. All of these technologies are use throughout this book.

I, ANDROID

Technically, there is a lot to know about the Android system and all of its layers and components. Bu because several books are already available that thoroughly discuss all the technical aspects of th Android system inside and out, you won't get too much technical information in this chapter. Yo will, however, become a bit more familiar with the sparks that brought Android to life.

If you want to get deeper into the technical workings of Android, we recommend *Beginning Andro 4 Application Development* by Wei Meng-Lee published by Wiley in 2012 if you are a beginner, o *Professional Android Application Development 4* by Reto Meier published by Wiley in 2012 if you a a more seasoned developer; both are excellent books.

The Three Laws of Android

The classic sci-fi author Isaac Asimov created some well-known rules within robotics, called th Three Laws of Robotics. In his fictional stories, these three laws define what a robot can and cannot d when interacting with humans.

Similarly to these laws, the Android Open Source Project (AOSP) is guided by a set of ideals th define why Android exists, how Android will continue to develop, and the roles for all th stakeholders in the project. In this section, you get a brief summary of the ideas that formed Andro into what it is today. Just like Azimov created three laws for his robots, in this chapter we summariz the ideals of AOSP into three laws; let's call them the Three Laws of Android.

NOTE *If you're interested in getting more detailed information on the Android Open Source Project and the Open Handset Alliance you should explore these websites in more detail*. http://source.android.com/about/index.html, www.openhandsetalliance.com/ *and* http://developer.android.com/index.html.

Law #1: Android Must Be Open and Free

The Android project was started back 2003 by a small company called Android, Inc., before the ter smartphone was widely recognized by the average user as the device we think of today — a devi with a large touchscreen, high-speed Internet connection, GPS, and other fun stuff.

The sole purpose of this company was to create a mobile phone jam-packed with different kinds sensors that would allow the phone to sense its surroundings. In essence, the company wanted create a smarter phone.

Some years later, in 2005, Google got involved (actually, Google bought the company and made it wholly owned subsidiary of Google, as it does in so many cases), and two years after this acquisitio (in 2007) the Open Handset Alliance (OHA), which curates the development of Android, was unveile sporting a total of 35 initial members. The OHA shared a common idea — the idea that openne improves innovation.

Another important concept of Android is the openness inside the system. Where other competin systems often restrict the capabilities of third-party applications and promote native application Android gives you the same freedom as the device manufacturers in developing for the systems.

The OHA has stated that the explicit goal of the Android system is to be the first open, complet and free platform created specifically for mobile devices.

Law #2: Android Must Be Adaptable

Through this openness and freedom rises the next law of Android; because the system is free f anyone to use, Android must also be highly adaptable. Not adaptable in the sense that anyone ca create their own version of the system, but adaptable in the sense that it must be capable of running o many kinds of devices and do it well.

This control of the project is called the Android Compatibility Program, which basically defin what it means for a device to be Android compatible. If a device doesn't comply with th requirements stated in the Android Compatibility Program, it can't take part of the shared ecosyste of Android.

NOTE *You'll actually find Android in just about any type of embedded device. It's used in phones, in tablet computers, and inside TVs. It controls printers and the media system in your car. Heck, you can even find it inside microwave ovens! This means that soon you will be able to write your own app for a microwave oven that sends an image of your cooked meal to your phone when it's ready, and share the app with your friends! Cooked by Android, mmm . . . yummy!*

This Android ecosystem is the backbone of its great market success over the past years. Because s many devices run Android, the potential number of customers for application developers is far beyon that of other popular systems today.

Law #3: Android Must Be Simple

Because the ecosystem of Android is the backbone of its success, the OHA considers you, th developer, one of its most important assets. If you cannot create stunning and innovative applicatio for Android, the whole system will fail in competition with other systems.

This is why the alliance strongly believes in empowering the developer, shortening the time fro your first app idea to your first market launch. Android achieves this through powerful developme frameworks and tools that are both simple in their nature and powerful in their actions.

In addition to the simple frameworks and tools, Android is known for its good documentation an many complete and open-source examples of using the available libraries. If you'd like to know mo about using a specific application programming interface (API), you can open the source of th example application through your favorite editor, or browse it online; and because the examp applications are all licensed under a very permissive open source license called Apache version you're allowed to use and build upon the example applications in your own commercial projects.

Also, because the Android SDK is built on Java you can often reuse a lot of code from projec you've been involved in before. However, when including code from normal Java projects you shou remember that one of the big changes in Android compared to other systems running Java is th rendering. For example, code written using the Swing framework cannot be compiled for Android.

All of these reasons make Android one of the simplest ways of getting started in smartpho application development, even for the complete newcomer.

The Android Philosophy

The Three Laws of Android act as a foundation on which the Android Philosophy is formed — philosophy that is influenced heavily by the concept called Open Innovation, a term coined by Hen Chesbrough in 2003.

He describes the traditional innovation process that formed most of today's powerful multination corporations like IBM or General Electric as fortresses in an otherwise barren knowledge landscape.

These fortresses were created out of a necessity; because knowledge was hard to come by, larg companies needed to invest heavily in research and development (R&D), an approach where the controlled the entire process of innovation from the very basic science to the finished product.

However, since then we've seen the knowledge landscape change drastically; more than 30 perce of the world's population is now connected to the Internet, workforce mobility has increased, an loyalty to our employers has decreased. This all points in one direction — the traditional R& departments find themselves in a situation where they stand to lose large resources spent o innovations that someone else is working on as well.

Enter Open Innovation; this new knowledge landscape has seen the corporate giants work more wi outside influences than before, either through consulting, the acquisition of new start-up companie or even cooperation over company borders.

NOTE *Eclipse, the most widely used integrated development environment (IDE) used for Android development, is another project heavily influenced by Open Innovation and Open Source ideas.*

Eclipse started as a project by IBM in the late nineties to develop a common platform for all IBM businesses, but because its partners weren't so enthusiastic about investing in the project, IBM decided to develop Eclipse under an Open Source license.

The move to an Open Source license was well received by the developer community, but it was still an IBM project, and this made many potentially critical contributors reluctant to commit large resources to the project in the event that IBM would close the project again. This marked the beginning of the Eclipse Foundation, an entity separate from IBM with the sole purpose of developing the Eclipse ecosystem.

At the time of writing this book, the Eclipse Foundation sported a total of 186 members, which makes it one of the most successful projects based on Open Innovation and Open Source to this day.

The Open Handset Alliance, and all of its members, sees the idea of Open Innovation as a critic new business model where sharing the risks and the rewards across company borders allows for muc faster and broader innovation, and in turn also renders a better experience for the user.

Other Popular Systems

When reviewing the Android system it would be good to compare it to other competing systems to g a better understanding of its place in the market. This section outlines the differences betwe Android and its most popular competitors, with a focus on developing accessories.

iOS

Based on the system found in other common Apple computer products, such as Mac Book, iOS is th version enhanced for Apple's handheld devices like the iPhone, iPod, and iPad. Although it wasn't th first smartphone system widely available, it was arguably one of the pioneering devices that shap today's smartphone market.

iOS is built as a proprietary, not licensable, system; this means that only Apple may develop an deploy it. Third-party developers require special developer licenses to create native applications for and the screening process for an application is also extensive, going as far as the general concept the application.

Since iOS version 3.0 there is support for external accessories through the External Accessory (EA framework. However, much like many of Apple's products, developing an accessory is a daunting tas that requires approval and often a serious investment by the developer. While this filtering ensures high-quality product and a style that conforms to Apples ideals with a high finish, this severely limi the possibility of exploration of the field by hobbyists.

Windows Phone

Not to be confused with its predecessor Windows Mobile, Windows Phone is a completely ne operating system by Microsoft released in 2010. Notably, the biggest difference is the new us

interface developed for the system.

Windows Phone is also a proprietary system owned and developed by Microsoft; however, it can b licensed to device manufacturers for deployment on their handsets — something that made a big bu in the industry in 2011 when Nokia announced its plans to adopt Windows Phone as its princip smartphone strategy.

As a developer you'll need to acquire a developer license to develop and publish applications f Windows Phone; and the applications also need to pass a validation and certification process b Microsoft. Unfortunately there's no official APIs available to develop external accessories yet, bu with the efforts put into the Windows Phone system we can only assume that there will be framework in the future for connecting your Windows Phone to your environment.

BlackBerry

Developed by Research in Motion, the BlackBerry devices saw great success in the beginning of th millennium because of the emphasis placed on communication. They were among the first mobi devices to focus on e-mail and push notifications on mobile devices, and this has become the signature feature over the years. And there is support for accessories since BlackBerry version 7.0.0.

The BlackBerry operating system is proprietary and non-licensable just like iOS, meaning that on Research in Motion will develop devices with it installed. Developing for BlackBerry is free, howeve selling applications on App World requires a vendor license; any applications that are published mu also pass a review before they're accepted.

Symbian

With market shares of around 70 percent at its peak, Symbian was the most widespread operatin system used for mobile devices; however, it has seen a steady decline over the past few years becaus of its failure to deliver a compelling user experience in competition with iPhone and Android.

Symbian, in comparison to iPhone and Android, has been deployed mostly on the older-style featu phone, even though it later released an updated smartphone version with all the traditional featur you would expect. For the older-style phone, you developed Java Micro Edition programs that wou run on top of the Symbian system, which is very different from how Android apps run.

The Symbian system was developed mainly by Nokia until 2011 when the switch with Windov Phone took place; since then the consulting firm Accenture has been charge of the development an maintenance of the Symbian system. Since 2010 the Symbian system has been published under th Eclipse Public License (EPL), this transition was also reported as the largest move from proprietary Open Source in history.

Preinstalled Applications

Most devices come with a set of preinstalled applications suitable for users new to smartphones. Oth mobile operating systems often protect these native applications and hinder any third-par application from taking over. But in Android, you're free to develop an application to replace an existing preinstalled app.

The preinstalled applications include, but are not limited to:

- Web browser
- E-mail client

- Phone
- Contacts book
- Notepad
- Play Store
- Camera
- Clock
- Google Maps

Of course, these applications vary from one device to another; often you'll see some manufacture providing their own version of any of these applications that they perhaps feel is improved in som fashion or specifically tailored to the look and feel of that specific device.

WHAT IS ANDROID OPEN ACCESSORY?

During Google I/O 2011, Google introduced the Android Open Accessory standard as the official supported way for developers to easily create and handle communication between an Android devic and any number of peripherals. Before this standard was announced, there were a couple of (let's ca them creative) solutions to allow you to create accessories for Android devices.

One of these creative solutions was a project called IOIO, a design that allows Android devices communicate with a specific USB-enabled Arduino microcontroller. IOIO manages this connectio through a very neat little trick with TCP sockets and the Android Debug Bridge (ADB) — normall used to develop and debug Android applications — and because ADB is available in all Androi devices, so too is the ADB solution.

> **NOTE** *Even though using the ADB in this fashion works well on all Android devices since all devices require the ADB interface, it's not an ideal solution in todays infected reality.*
>
> *Even the most security-aware of us have at some point come in contact with digital viruses or malware; enabling your smartphones Debugging mode opens the gates wide for all kinds of malware to be installed by your home computer when connecting your phone to it — which is not an uncommon sight when you want to back up your data, such as photos, apps, and contacts.*

When talking about Android Open Accessory, you should separate two things. The first is th Android Open Accessory framework, which is the protocol that controls the communication betwee two USB devices; and the second is the Accessory Development Kit, or ADK for short, which is th hardware and software needed to make an Android-compatible accessory.

In short, the USB libraries introduced in Android 3.1 enable you to create applications th communicate with USB devices, either custom-built ones or common off-the-shelf PC peripherals.

> **NOTE** *Actually, the Android Open Accessory framework is available from Android SDK 10 (version 2.3.4) as a compatibility package called* `com.android.future.usb`. *You can find, and explore, this compatibility package inside an external jar file called* `usb.jar` *inside the add-ons folder; see* `<android-sdk-folder>\add-ons\addon-google_apis-google-10\libs`.

Android USB in Short

At the time of writing this book, two kinds of accessories were available for Android. The first is US Host mode, which is very hardware dependent, meaning it will only work on Android devices th have this mode enabled. However, on the devices that support USB Host mode, you can connect an PC peripheral to your Android device and use it in your app.

For devices that don't support USB Host mode, there is the Android Open Accessory mode whic provides the bulk communication channels required to talk to your hardware accessory.

In the Accessory mode the roles have been switched so that the Android device is now actually th USB Slave, and the accessory is the USB Host. You get a more detailed review of these modes later i this book.

Android supports the following interactions over the physical USB port:

- USB Host mode since Android SDK 12; using this mode, the Android device assumes the role of the Host.
- USB Accessory mode since Android SDK 12, backported to SDK 10. When using this mode the Android device assumes the role of the Accessory.

Developing Android Accessories

Another important aspect of the Android Open Accessory framework is the development cost, both i resources and in time. It shares the same ideals as the rest of the Android Open Source Project:

- It's open
- It's free
- It's simple

WHAT IS ARDUINO?

Arduino is an Open Hardware project started in 2005 that tries to bring the world of digital electroni to education, research, and the maker community. Arduino stands for ease of use, openness, an world-wide availability.

Arduino started as a simple prototyping circuit board, a small computer, running at 16 MHz. It ha no screen and no keyboard, and therefore requires an external computer to program it. That comput has to run a piece of software called the Arduino IDE that helps with writing, compiling, an uploading programs into the board. The board is then autonomous; it doesn't require the computer the IDE to continue executing the uploaded code.

You need documentation when learning about almost anything. The third leg of the Arduin ecosystem is therefore a series of reference files and tutorials for people to teach themselves about th use of digital technology. All the documentation is gathered around the Arduino websi (www.arduino.cc). The official documentation is generated in English and then translated to oth languages by a community of volunteers.

A whole range of different Arduino boards is available to suit your prototyping needs in differe ways. For example, if you were interested in just reading a sensor and plotting its value as part of a application in your computer, you would need the Arduino Uno board, with 14 digital input/outp pins and six analog inputs. If you were about to build a small wearable computer that beeps when th

temperature reaches a certain value, you would use the Arduino Lilypad, which can be stitched on fabrics using conductive thread. If you were in the need of a small server offering information abo the quality of the air in a room, you could use an Arduino Ethernet board connected to a public number.

For the purpose of this book, we are going to use the Arduino Mega ADK and the Arduino Du boards as our microcontroller boards. Both boards allow connecting USB devices to the board. In o case we will connect Android devices (phones and tablets) to the circuit. Those boards bring in th possibility of controlling physical objects from a phone or reading a biometric sensor and sending th data to the phone for storage.

Besides the microcontroller boards, you will find the so-called Arduino Shields. The shields a boards to be stacked on top of the Arduino board, offering some more specific functionality. You ca find very simple ones including a couple of potentiometers and some buttons, to ones that offer a GP gyroscope, and GPRS connectivity.

> **NOTE** *Shields work with almost any kind of Arduino board, but beware that some Arduino boards are developed for specific purposes and because of this the Shields may not fit these boards.*

There is a whole world of possible shields for Arduino out there. They are manufactured by multip vendors in just about every country in the world. This is also one of the strengths of the Open Sour model — because all the software is open, it is possible for anyone to create new hardware add-o and write drivers for them to run on the Arduino board as libraries.

HOW DOES AOA WORK WITH ARDUINO?

AOA includes a set of libraries that allow bidirectional communication between Android devices an Arduino boards. Arduino boards use the USB port as a way to communicate to computers. It possible to make a board be listed as USB keyboard or mouse. You can even find examples on how t turn your Arduino board into a USB MIDI device.

> **NOTE** *MIDI stands for Musical Instrument Digital Interface. It's a specification that allows different devices — mainly musical instruments — to connect to one another. In short it is a special modification of a serial port working at 31.250 bps with a set of rules for how to encode different controls and sound actuators.*
>
> *Physically, the different devices connect through 5-pin DIN connectors. It is only since the 2000s that MIDI-to-USB converters have allowed interfacing these devices with state-of-the-art laptop computers. Recently, many of the MIDI instruments implement MIDI over USB and have removed the DIN connectors. Arduino Uno, Mega and Mega ADK can be reprogrammed to become native MIDI over USB devices. You can read more about it at:* www.arduino.cc/en/Hacking/MidiWith8U2Firmware.

Some Android devices allow connecting keyboards, mice, and so on, and it should be possible connect your Arduino board to, for example, your Android tablet that way. Technically, Androi devices are computers running a derivative of the Linux operating system. Conceptually, for the fin user, tablets and phones aren't computers, or if they are, they cover a different need and therefore the

aren’t perceived as such.

For this reason, Google introduced the idea of accessories as devices that can be connected Android devices to enhance their functionality in a slightly different way to how a mouse relates to PC. At low level, the PC is a USB hub, whereas the mouse acts as USB client. The Android accesso is a USB hub and the phone/tablet is the client.

This makes sense from a conceptual point of view for many reasons. Consider, for example, that yo buy an accessory to measure your blood pressure. This is not very far from the kinds of products w are going to see coming to the market soon. Once you get it, you need an application to get it to wor What the accessories do is to inform the phone about the name of the artifact, the manufacturer, th software version, the name of the application, and, most importantly, the URL where you ca download the application directly into your phone. You will not need to make a search in the Goog Play; the accessory can tell your phone where to search for the app without having to type anything.

In this book we explore the connectivity via a USB cable between your Arduino board and yo phone. On the Arduino ADK side there is a library that controls the USB Host functionality. Th allows detecting when an Arduino board was connected to your phone. It also allows for sending an receiving data between the two devices. On top of that host functionality there is yet another layer th tells the phone exactly which app should be launched when connecting the accessory, wh manufactured it, and the version number. This is the information the phone will use to decide what launch when and when to launch it.

It is also possible to create Android accessories that work over Bluetooth. You should, however, no confuse a Bluetooth accessory with a USB accessory. Except for the obvious differences in hardwa layer, the Bluetooth accessory works with the common Bluetooth API available since Android SDK 5

WHAT CAN YOU DO WITH AOA?

It is possible to design accessories that will read data from the environment using sensors not include in your Android device, such pulse-rate monitors, air pressure sensors, gyroscopes, or pyroelectric ga detectors.

You can use actuators like servo motors, solenoids, peltier cells, steppers, or piezo-elements. Yo can build robots using the phone to control them, or you can unlock your apartment door from distance by sending a text message.

If you want to hook up your phone to control a wireless sensor network communicating over ZigBe you can use AOA to proxy your communication through an ADK board with a wireless shield on it.

NOTE *ZigBee is a specification for a series of protocols using low-power digital radios. It is intended for personal area networks like a heartbeat sensor communicating with your phone or a wireless temperature sensor to be installed on your balcony.*

ZigBee devices define mesh networks to transmit data over longer distances. ZigBee networks use intermediate devices to reach more distant ones.

ZigBee runs in the same piece of the spectrum as Wi-Fi and Bluetooth. Comparing the features of the three of them, you could conclude that:

- *Wi-Fi is used for high-speed data transfers like browsing a video archive, where many devices connect simultaneously.*
- *Bluetooth is a one-to-one cable replacement. The latest standard, version 4, allows for very low-power radios, which makes it very suitable for small personal area networks.*
- *ZigBee is used for mid-size networks with low bandwidth requirements and high reconfiguration needs like wirelessly monitoring livestock.*

You can read more about ZigBee at: www.zigbee.org.

The possibilities are endless, but you want to concentrate on choosing projects that will allow you learn at the right pace. Therefore, we have selected a series of projects that gradually increase complexity.

We start with small tasks like reading a temperature sensor and gradually move into controllin motors to then integrate them with distance sensors, adding more intelligence to the system.

In short, AOA allows you to:

- Connect external sensors to your Android
- Control the world through actuators
- Debug your apps from your development environment

WHAT CAN'T YOU DO WITH AOA?

You should avoid building devices that deal with monitoring human constants under critic situations. The tools presented here are not suitable, without the proper level of expertise from yo side, to monitor a machine connected to your body. You should take into account that the system used in this book are not reliable under special circumstances. Neither the Arduino Mega ADK, n any of the Android phones and tablets presented here were designed to be used in rough environmen (like at too high temperatures or under water). The AOA protocol running on top of those systems is simple communication protocol, not robust enough for those conditions either.

WHY IT MATTERS THAT GOOGLE CHOSE ARDUINO

Open Hardware establishes that individuals, institutions, and companies should have a similar set rights to create, publish, and share hardware designs as they can within the Open Source/Free Softwa movements. The big difference is that Open Hardware refers to physical goods, whether circu

boards, chairs, or mechanical parts.

When talking about circuit boards, the Open Hardware movement has been around for a while. Sinc the first computer clubs in the ’80s, people were sharing board designs and firmware. It is th development of computers as a business that brought patents into the game of hardware that affecte the way it was used in some fields like education.

The following quote is taken from the OpenSparc project (see www.openspark.net). It represents th understanding of what Open Hardware was meant to be for many until the mid-2000s. Institution companies, and other hardware-interested bodies thought that it could be open from the point of vie that Field Programmable Gate Arrays (FPGAs) and other chips would represent the configuration o their logical gates in the form of source code. In other words, Open Hardware was perceived as a evolution of open source code, but not as something that would refer to the actual physical object.

> *Small amounts of computer hardware Intellectual Property (IP) have been available for many year in open-source form, typically as circuit descriptions written in an RTL (Register Transfer Level language such as Verilog or VHDL. However, until now, few large hardware designs have been available in open-source form.*

When Arduino showed up in 2005, it was presented as a piece of Open Hardware. Back then, th Arduino website invited people to download and use the reference design of the board to produc derivatives and learn about hardware by building it themselves. Due to a series of online debates, th Arduino Team (the core group of Arduino designers and maintainers of the project), realized that there was no legal way to protect physical objects like circuit boards. The decision was made to use Creative Commons (CC) license to protect the digital file or the blueprint that was needed to manufacture the boards.

From a legal point of view, there is no difference between the file or illustration that shows th circuit design and a poem or a musical score. And if the latter could be protected with CC licenses, th boards could as well.

The Arduino CC license allows other designers, educators, and users in general to take the referenc design, modify it, manufacture boards, and sell them as long as they respect the Arduino name, whic is a trademark registered by the Arduino Team. These terms are very simple and flexible. When takin the reference design to make your own, you just need to credit the initial design team, but you don need to pay anything, nor do you need to tell them in person.

In May 2011, Google’s Accessory Design team came to the point when they needed to exemplif how people could create new accessories for the Android operating system, and Arduino’s hardwar license made their purpose very simple. Google could take one of the Arduino reference designs (this case the Arduino Mega) and merge it together with yet another Open Hardware piece, the US Host shield by a company called Circuits@home (www.circuitsathome.com) to make their Goog ADK board.

It is actually an important milestone in the history of Open Hardware because it showed how b corporations could actually become players in the community and contribute back. Google’s desig was also open and licensed under the same parameters as the original Arduino board; this allowe many taking this new blueprint to use it as a starting point for their work.

In parallel to these series of events, a group of makers and companies had been meeting up in Ne York since 2010, putting together the Open Source Hardware Definition (se http://freedomdefined.org/OSHW) as an attempt to make a declaration of intentions on what Ope Hardware should be. A couple of months after Google’s announcement, the CERN institute i

Switzerland announced the first Open Source Hardware License (see http://ohwr.org/cernohl) as a fir attempt to produce a legal framework for people to protect hardware in an alternative way.

The Open Hardware culture is a movement growing from the bottom up that is involving entiti coming from all countries across the world and that tries to protect designers and makers so that the creations can be made available for others to use. This has a huge impact, especially in fields whe cost is an important factor, like education or technological deployment in humanitarian scenarios.

SUMMARY

Android is an operating system developed mainly for use in smartphones, but you can find it in oth devices as well such as printers. The way Android is developed and maintained is one of the reason for this wide range of applications, and it's also one of the main reasons it has excelled on th smartphone marked in the past few years when many proprietary systems are struggling.

Another reason for Android's success is its familiarity; the tools used to develop software ar already widespread in industry, education and among hobbyists. Both Android and Arduino rel heavily on Open Source, Open Hardware, and Open Innovation models and thriving communities develop and maintain.

Qualities of each of these models are leveraged when creating Android accessories, and in time w will hopefully see that the market for accessories will skyrocket because of this.

You also learned some key differences between the Android ecosystem and other popular mobil ecosystems. Where the others often hinder some stakeholders, Android delivers:

- Freedom for device manufacturers
- Freedom for application developers
- And, most importantly, freedom for the user

Hopefully, you will soon start to feel like an integral part of a community that drives th development of smartphone accessories forward . . . to the future!

Chapter 2

Setting up the (Arduino) Hardware

WHAT'S IN THIS CHAPTER?

- Introducing sensor technology
- Using actuators, motors, LEDs
- Working with platforms and architectures
- Powering up your projects
- Using Arduino ADK versus Google ADK boards
- Working with add-on boards: shields

This chapter deals with choosing the right physical tools for developing a project. Whether you wa to measure the temperature in a room or build a robot, you always have things to take into accou You need to consider what it is you want to monitor, the size of the prototype, or the computing pow you need.

You have probably heard the statement that everything is a computer nowadays. The use microcontrollers responds to the paradigm of so-called ubiquitous computing. Computers a everywhere: the average car has 70 processors, your microwave has one, and your cellphone has couple of them. You can find general-purpose processors that can be reprogrammed into maki almost anything. You can also find purpose-specific processors, aimed at solving a certain type issue such as USB-serial conversion, decoding MP3 sound files, or controlling the movement of motor.

In this chapter, you learn about sensor technology. Microcontrollers are just the intelligence of yo embedded project. To gather data about the world, you need devices that translate real-world data in digital information. *Sensors* are the interfaces that translate properties of the world such temperature, acceleration, or intensity of light. These are then translated into a voltage to be read b the microcontroller.

You are also introduced to the concept of an *actuator*. In the same fashion that sensors help reading the world, actuators "write" to the world. Motors and speakers are probably the most comm actuators. Both transform electrical impulses into mechanical movements.

Finally, this chapter discusses the issues with powering up your project: will you make somethi that runs on batteries or from a power socket? Dealing with power, battery charging, and simil issues is always complex. A full examination of the various options is not within the scope of th book; however, we describe the options for you and suggest an easy way to solve most of yo prototypes quickly.

CHOOSING MICROCONTROLLER BOARDS FOR YOUR PROJECT

This book explores the connection between one type of microcontroller ecosystem (Arduino) and th Android operating system for mobile devices. To start with, Arduino is just a microcontroller that ca run software made in C. Even if it is possible, Arduino doesn't run an operating system. To keep th system simple, it runs sequential programs performing commands one by one. In contrast, Android a whole operating system that can run on phones and tablets using many different vendors. A Android device has one or more microcontrollers and runs multiple processes in parallel.

You can find many types of microcontrollers from several different vendors in the market. Don't g confused between the microcontroller platform and the microcontroller itself. Microcontrollers are th chips, the black boxes on the circuits. Platforms are the ecosystem integrated by the microcontroll — its circuit, the programming environment, and the documentation.

In that way, some platforms depend very much on the processor's architecture, whereas others ca be implemented on technology from many different vendors. Arduino has been designed to run o different architectures. The API used to program the microcontrollers has been translated int processors coming from a whole series of vendors. Because you learn how to use Arduino boards i this book, we show only a quick analysis of the official Google ADK boards, which are full compatible Arduino boards.

When you want to add features to your project quickly, instead of building all the parts for it on breadboard, you can use pre-assembled add-on boards. In the Arduino world, these pre-assemble boards are called *shields*. A few examples of shields that might be of interest to you are describe later.

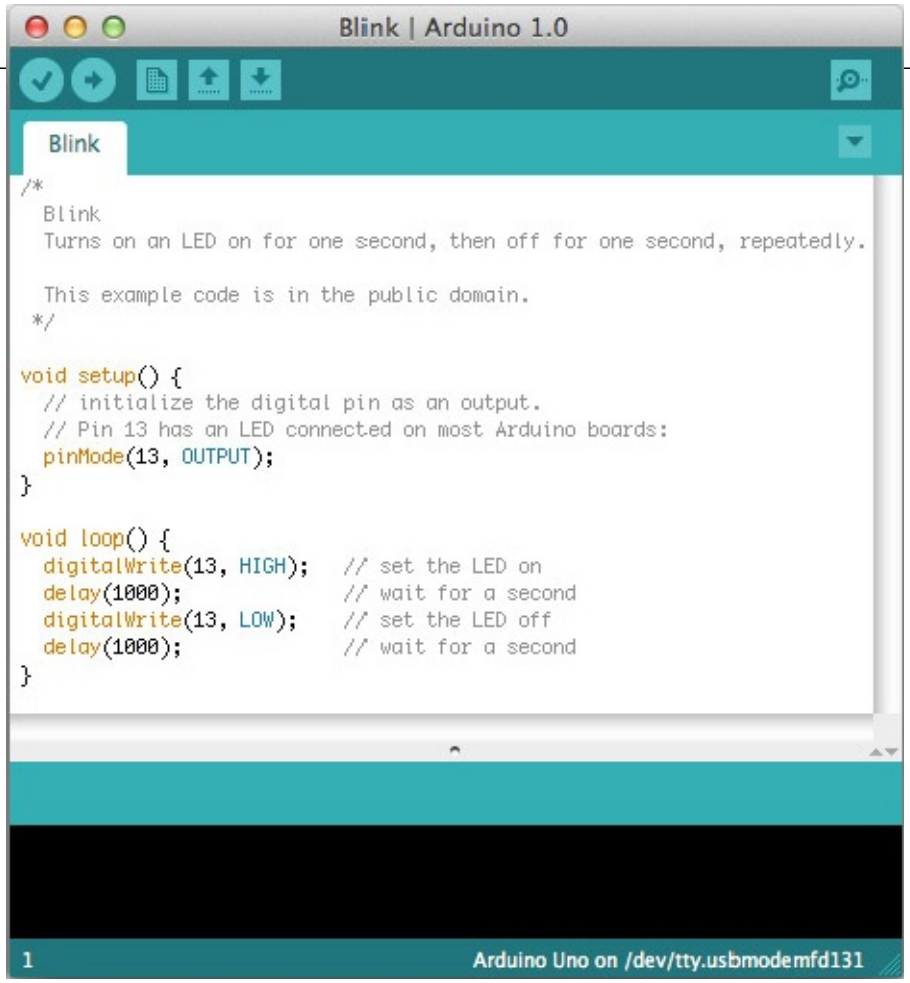
One Platform, Many Architectures

Arduino is a software abstraction that you can apply to many different architectures. You can mov the same software seamlessly between boards in an upwards-compatible way. Some of the newe programs might contain features that cannot run on older boards. Different manufacturers (Atme Freescale, ST, Microchip, Texas Instruments) are making microcontrollers. Those chips have to b integrated into platforms for people to use them. Those same vendors produce prototyping platform for engineers to try out the chips and decide whether or not those processors could be used as part of project.

> **NOTE** *At the time of writing this book, the Arduino project was producing boards using one single manufacturer: Atmel. The platforms made by Arduino made use of two different chip architectures. Some of the boards ran on 8-bit microcontrollers from the ATmega family, whereas the most recent ones ran on 32-bit microcontrollers with ARM Cortex-M3 cores.*

Arduino is an external actor that can use any of those manufacturers to create prototyping board One of the main goals when learning about digital technology through Arduino is that it should b easy to use. That ease of use is achieved through a simplified IDE and curated documentatio available online. Figure 2-1 shows the Arduino IDE running on a Mac computer. The IDE is cros platform and can run on Windows, Linux and Mac computers. The code should compile the same wa and the board should behave identically.

FIGURE 2-1: Arduino IDE v1.0.1 on Mac OSX



From an engineering point of view, you would analyze how to solve a project from the technic features in a brief. You would then choose the right microcontroller and IDE to program it.

Choosing a platform like Arduino enables you to start your project with one tool and, if that doesn completely fulfill your needs in terms of available inputs/outputs, size, or speed, you could move in using a different platform without changing your code. It also enables you to go the other way aroun You could start with your most powerful platform and once you are done, trim the project down make it fit in as small a form factor as you need.

Prototyping is an art that you learn by experience. It is very hard to predict everything you need take into account before you start a project. In many cases, you might get overwhelmed by trying anticipate everything that could happen. We have seen many projects die prematurely during th initial planning phase. We would recommend you get started building and tinkering and then mov between tools as your project evolves. Once you have made a couple of prototypes, you will have much better understanding of what is possible with each board and you will gradually get better predicting what you need in each case.

The following sections, as shown in Table 2-1, give an overview of different prototyping boards ar examples of using each one. Also, keep in mind that the Arduino boards are open-source hardwar which means that it should be possible to find platform vendors making Arduino-compatible boar with similar or equal functionality to the ones mentioned here.

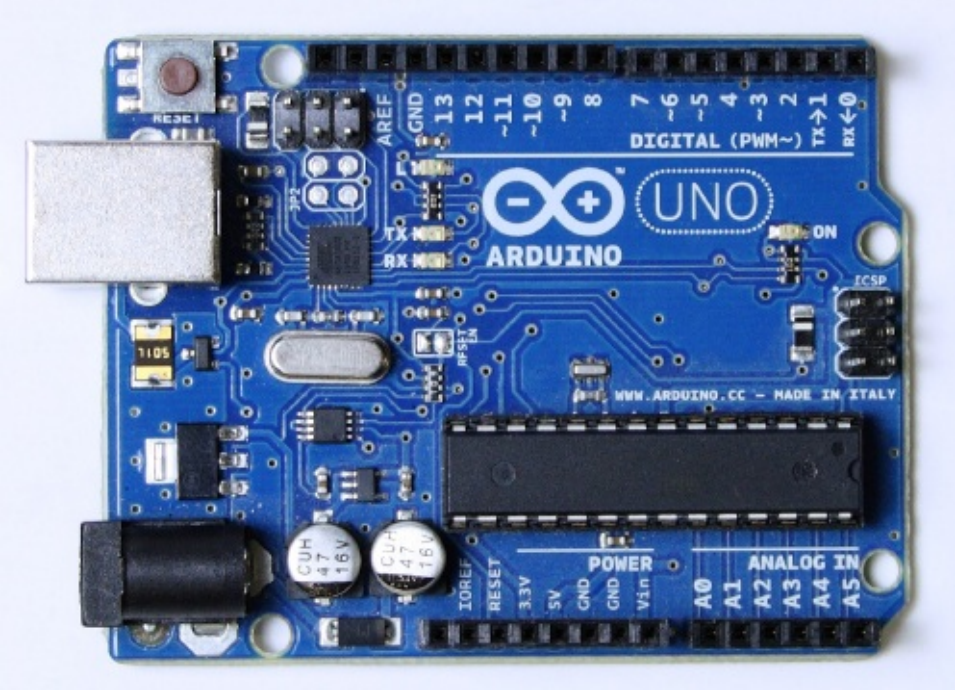
TABLE 2-1: Comparing the Boards Mentioned in this Chapter

BOARD	ARCHITECTURE	DIGITAL I/O	ANALOG I/O	ADK FUNCTIONALITY	SERIAL PORTS
Arduino Uno	8-bits	14 pins	6 analog inputs, 6 digital pins do PWM	No	1
Arduino Mega ADK	8-bits	54 pins	16 analog inputs, 16 digital pins do PWM	Yes	4
Arduino Due	32-bits	54 pins	12 analog inputs, 12 digital pins do PWM, 2 pins from DAC	Yes	4
Arduino Micro ADK	8-bits	11 pins	6 analog inputs, 6 digital pins do PWM	Yes	1
Google ADK	Arduino Mega ADK compatible				
Google ADK2	Arduino Due compatible				

Arduino Uno

This is the most extended board from the Arduino family. The board carries an 8-bit microcontrolle and it comes with 14 digital input/output pins and 6 analog inputs. Six of the digital pins can b programmed to send pulse width modulation (PWM). The Uno board also comes with intern peripherals able of running the UART, SPI, and I2C communication protocols. Programs using th Arduino Uno board (Figure 2-2) can be as big as 30 Kbytes and run at 16 MHz.

FIGURE 2-2: Arduino Uno board



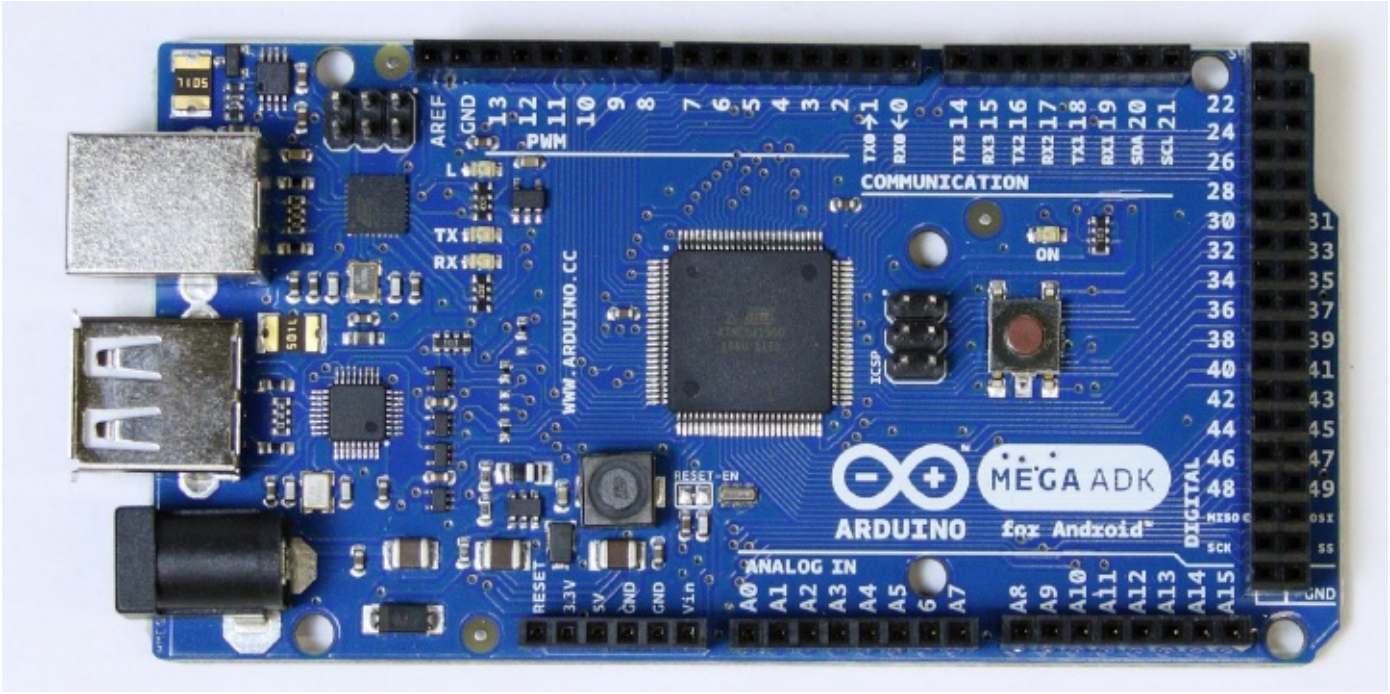
As you can see, Arduino Uno is a very versatile board that has become the Swiss army knife of th maker community. In literally five minutes you can plug this board into your computer and sta programming it. It has a disadvantage, though: You cannot use it to communicate directly with a Android device, because it lacks the USB Host functionality. However, shields are available that ca bring the USB Host functionality to the board, as well as some that could add Bluetoo communication. Both methods allow the microcontroller to talk to Android devices.

Arduino Mega ADK

The most extended of all the prototyping boards that can communicate with Android phones probably the Arduino Mega ADK. This board, shown in Figure 2-3, includes all the functionality of a Arduino Mega 2560 board plus the USB Host. The Arduino Mega 2560, and derivatives, includes

chip with 252 KB of available memory space, 54 digital input/output pins (of which 16 can u PWM), 16 analog inputs, and 4 serial ports (UART). It is very convenient for projects that use senso to communicate over serial (UART) or systems that require reading many inputs. Like its small brother (the Uno) also works at 16 MHz and uses an 8-bit processor.

FIGURE 2-3: Arduino Mega ADK board

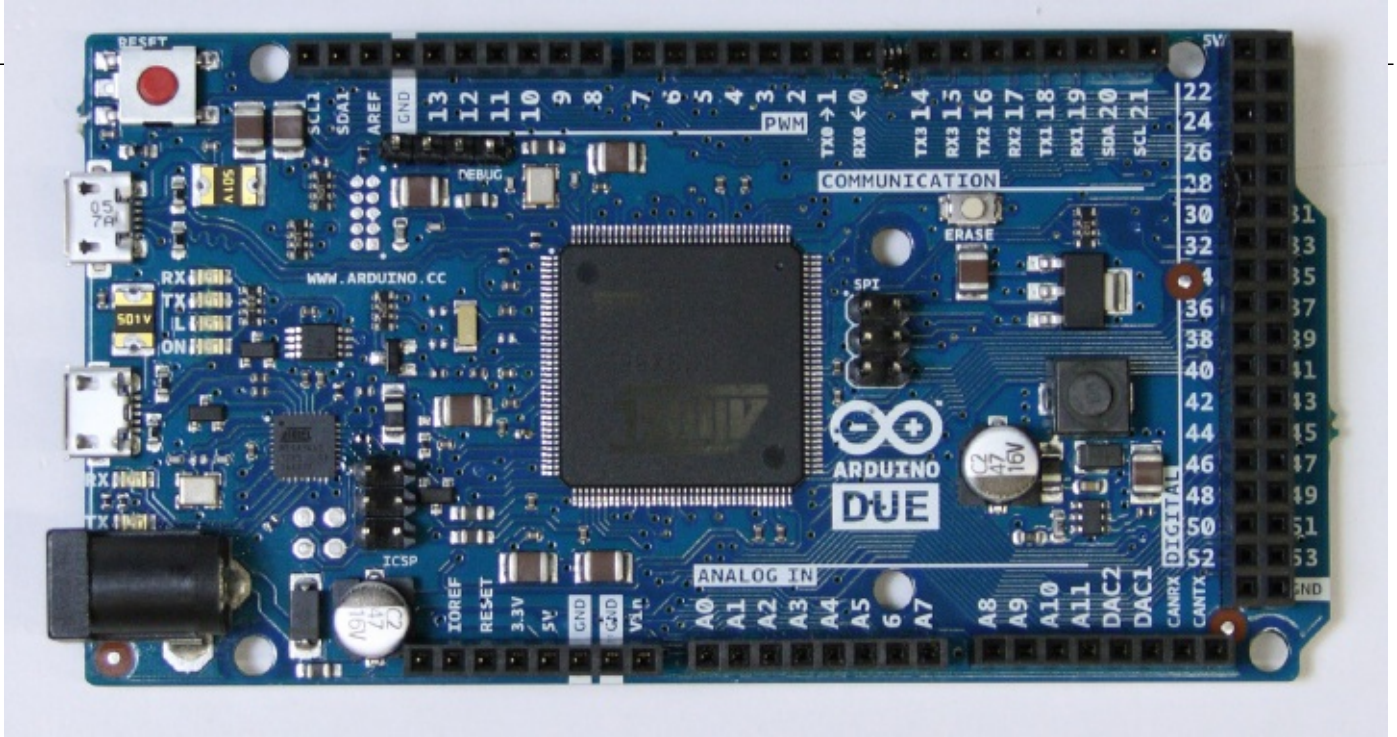


As mentioned, the Mega ADK adds to that the USB Host functionality. This means that you ca connect a USB device through an on-board female USB connector. It is, of course, possible to conne other USB devices to the board such as Bluetooth dongles, keyboards, or USB drives, but we are n going to explore those possibilities as part of this book. However, you can find multiple exampl online on that regard.

Arduino Due

Arduino Due (Figure 2-4) is the first board within the Arduino family to leave the 8-bit realm. It ru an ARM core type Cortex-M3 with 32-bit, internal Digital Signal Processor (DSP), a 12-bit Digital Analog Converter (DAC), UART, SPI, I2C, and other peripherals. Probably the most interesting of a its capabilities when it comes to the AOA is that the microcontroller has an internal USB OTG (C The Go) peripheral. This is what is used to connect to USB devices either as a host or as a client. other words, the chip includes the possibility to connect to the Android phone directly.

FIGURE 2-4: Arduino Due board



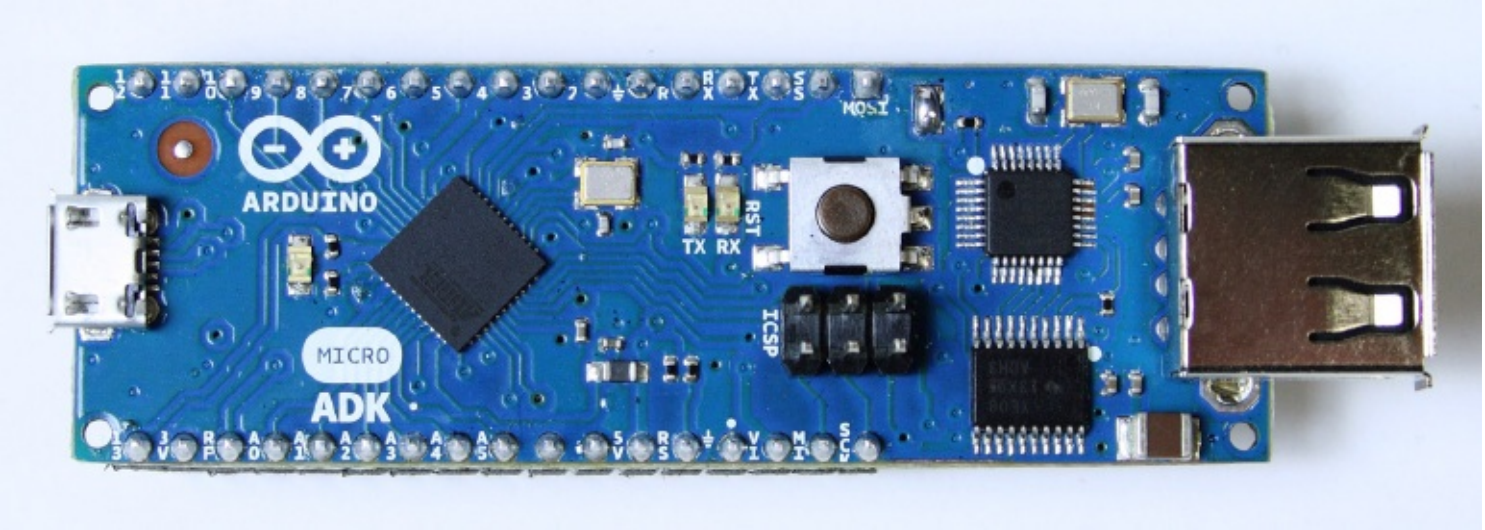
However, from a programming perspective, the experience is going to be the same as using th Arduino Mega ADK. The same code that runs in one of the boards should run on the other, except fo those cases when you might be using some of the extended features from the ARM core like DSP c internal DAC.

Arduino Micro ADK

Probably the only reason to not use any of the previously mentioned boards is their form factor. Bo the Arduino Due and the Arduino Mega ADK have the same shape and size (5.33 × 10.16 cm or 2.1 4 inch). The Arduino Uno is shorter (6.85 cm or 2.7 inch), but in turn, you need to add the USB Ho shield on top of it to achieve the desired functionality. Any of the combinations are optimal in term of the prototyping experience. They have enough ports and pins to accommodate almost any proje you could envision. But when you want to pack your project in a small form factor, you need rethink how to put things together.

This is where the Arduino Micro ADK board (Figure 2-5) comes in. Its smaller form factor (2.3 6.85 cm or 0.9 × 2.7 inch) makes it perfect for many projects. It offers fewer pins and you need power it up from a battery, but for wearable projects or those where you just want to control a coup of motors and hide everything in a small box, this might be the best tool for you. It also has son really nice features the other boards don’t have. Its programming port can transform the board into computer keyboard or a mouse. You could imagine making an app that would type SMS messages i your computer, or execute certain actions using the command line.

FIGURE 2-5: Arduino Micro ADK board



Arduino ADK vs. Google ADK Boards

Google’s official ADK boards (the ADK from 2011 and the ADK2 from 2012) are proofs of concep presented by Google. The company from Mountain View is not interested in manufacturing ar selling these platforms — it wants developers to get excited about making accessories for Andro devices and therefore make reference designs available for others to take over and bring to the marke

The Google ADK (shown in Figure 2-6) is a development kit presented by Google at its annu conference in May 2011. It is made of two parts:

FIGURE 2-6: Google ADK board