Getting Started with UDOO

Become an efficient maker by designing and building amazing prototypes with the UDOO platform and Android

Foreword by Maurizio Caporali, Product Manager and Cofounder of UDOO

Emanuele Palazzetti
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 ‘Turn On the Engines’
- A synopsis of the book’s content
- More information on Getting Started with UDOO

About the Author

Emanuele Palazzetti is a software developer with strong JavaScript knowledge and a solid background in Python development. Part of his job entails taking care of backend web applications mostly built with the Django web framework. In the past few years, he has delved into many well-known frontend frameworks, such as AngularJS and EmberJS, to strengthen his JavaScript knowledge. Being a former Java developer, he leveraged his previous knowledge and started working on Android; by combining his interest on electronics and embedded devices with the well-known mobile platform, he managed to build several prototypes and small physical applications.

He currently works at Evonove, a small company where he has been leading a number of frontend software projects mostly based on the AngularJS framework. He is also an open source advocate, active contributor, and speaker at Python and Android conferences. He writes about JavaScript and other development stuff at http://palazzetti.me.

Many people have helped this book happen. Reshma Raman, Acquisition Editor at Packt Publishing, gave me the opportunity to write this book and Ruchita Bhansali, Content Development Editor at Packt Publishing, provided great support and encouragement during book writing and reviewing. Many thanks to the entire team at Packt Publishing that worked hard and provided suggestions to improve the content of this book.

Thanks to Maurizio Caporali, Primiano Tucci, Edward Sayadian, and Christian Gil González for their valuable and welcomed feedback, which was critical for better clarity and accuracy of the material in this book.

Thanks to Massimiliano Pippi, a colleague and a friend of mine, for his suggestions and feedback during the initial draft.
Getting Started with UDOO

Since the beginning of the 2000s, a global rekindled interest in hardware manufacturing occurred, due to many advances in engineering and microelectronics, which granted the proliferation of new kinds of inexpensive manufacturing tools. People of all ages, even children, started to convert their broken devices, old toys, and every unused piece of hardware, into new amazing objects. This unconventional approach to design and creating something new was characterized by a new way to express creativity, and this was the key factor that created the maker culture.

This was the maker revolution, a movement that changed our world radically. Open source projects provided all the required tools to unleash the creativity to build something, without the need for a strong knowledge in programming and engineering, nor a set of expensive components. Indeed, one of the most important successes achieved by the maker revolution was the capability to move prototypes manufacturing from small or big industries to our homes.

In February 2012, another open source project, called UDOO, started a prototyping board featuring Linux and Android operating systems, with the goal of combining the winning characteristics of Arduino and Raspberry Pi in one single board. After a year working on this project, in April 2013, the UDOO board joined the Kickstarter crowdfunding platform, and the feedback from the maker community was overwhelmingly positive—the project was funded in just 2 days.

Makers all over the world liked the project so much that they decided to contribute, not only through Kickstarter pledges, but even through useful ideas and advice during the board design phase. The result of the help provided by the maker community is a powerful prototyping board to build interactive and creative ideas we always wanted.

This book will teach you how to build your first hardware projects using the UDOO board as a fast prototyping tool. Starting with a simple application that involves the use of basic electronic components, you will work through different projects that let you build electronic circuits, with improved interactions and offered by the Android operating system.
What This Book Covers

Chapter 1, *Turn On the Engines*, walks you through the setup of the UDOO platform and the required development environment. It begins with an introduction of the board, showing its peculiarity and capabilities that distinguish it from the others; then it guides you through the installation of the Android operating system. The last part, explains how to configure the development environment, both for Arduino and Android, to launch the first Hello World Android application.

Chapter 2, *Know your Tools*, teaches how the Android application is capable of controlling connected devices. It begins with some onboard Arduino characteristics and then explains how to create the first Android application that is capable of communicating with the integrated Arduino device. It then shows how to create a fully functional circuit using a breadboard for fast prototyping.

Chapter 3, *Testing your Physical Application*, explains the main concept behind physical application testing. The first part shows how to build a circuit that is testable from a software application. It then shows how to implement a diagnostic mode to test whether the connected circuit is working correctly.

Chapter 4, *Using Sensors to Listen to the Environment*, begins by explaining how sensors work and how they can be used to make the prototype context aware. It then shows how to build a heartbeat monitor, coding an Arduino sketch to read the sensor's data, and an Android application to visualize the computed results.

Chapter 5, *Managing Interactions with Physical Components*, teaches how to manage user interactions. It starts by explaining some components that can be used to let the outside world interact with the system. It then shows how to build a web radio with a physical controller to manage the prototype volume and to change the current station. In the last part, an Android API is used to playback Internet radio streaming.

Chapter 6, *Building a Chronotherm for Home Automation*, explains how to use some UDOO capabilities for home automation. It shows the creation of a Chronotherm using a circuit to detect environment temperature, and an Android user interface to visualize sensor data and to change the desired temperatures for each time interval.

Chapter 7, *Using Android APIs for Human Interaction*, adds more functionality to the application from the previous chapter extending the settings management to store different presets using voice recognition and synthesis to manage users' interactions.
Chapter 8, *Adding Network Capabilities*, extends the Chronotherm application again with the capability to collect forecast data through a RESTful web service. In the last part, it shows how to use collected data to provide more functionalities to the Chronotherm.

Chapter 9, *Monitoring your Devices with MQTT*, teaches the main concepts of the Internet of Things and the MQTT protocol, used to exchange data between physical devices. It then shows how to set up a cloud-based MQTT broker, capable of receiving and dispatching the Chronotherm temperature updates. The last part shows how to write a standalone Android application to receive the sent data from the Chronotherm.

This a bonus chapter and is downloaded from the following link: https://www.packtpub.com/sites/default/files/downloads/1942OS_Chapter_9.pdf
Any idea should start with a prototype. It doesn't matter whether it's a game, a web or mobile application, or a generic software component. Every time we want to deliver something to our final users, we have to create a prototype first. This is the most important step because it's when we start to face our first difficulties and when we may change some important aspects of our project.

If we are writing a software component, the first prototype isn't too expensive because what we need is our time and passion. However, this isn't applicable when our project has some hardware parts because it could be too expensive to afford all the required components. This statement was true until programmers, engineers and open source lovers started to release projects such as Arduino.

Fast prototyping boards let people realize projects with cheap or reused old components, and this, together with the Do It Yourself (DIY) philosophy, allows the creation of a huge community that spreads all over the world. This is where the UDOO board plays an important role in the makers' community: the hardware prototyping ecosystem, together with the traditional way to write software applications, represents a powerful combination for interactive projects creation.

In this chapter, we will explore more details about the UDOO board, focusing on the elements that are important to get started. In particular, we will cover:

- Exploring the UDOO platform and its main characteristics
- Setting up the board with the Android operating system
- Configuring the development environment for Arduino and Android
- Bootstrapping a simple Android application
- Deploying an Android application
Introducing the UDOO platform

The UDOO board is designed to offer us great flexibility with the tools, the programming language, and the environment in which we build the first prototype. The main goal of the board is to take part in the era of the Internet of Things and this is why an embedded Atmel SAM3X8E ARM Cortex-M3 processor is its first building block.

This processor is the same that powers the Arduino Due board and it's fully compliant to Arduino pinout. The result of this feature is that the board is compatible with all Arduino Due shields and most of Arduino Uno shields, so developers can convert and reuse their old programs and circuits.

The UDOO I/O pins are 3.3V compliant. For instance, if you're using a sensor powered by 5V that outputs the signal to UDOO pins at 3.3V, then you're fine. On the other hand, if the sensor outputs the signal to UDOO at 5V, it will damage your board. Every time you're using a shield or a sensor, be aware of provided output voltage to UDOO pins. This precaution is the same for a traditional Arduino Due board.

The second building block is a powerful Freescale i.MX 6 ARM Cortex-A9 processor, which is shipped in Dual and Quad core versions. The official supported operating system is UDOObuntu, which is a Lubuntu 12.04 LTS armHF based operating system that ships out of the box with many pre-installed tools to be up and running quickly. Indeed, after your first boot, you have a fully configured development environment to program the onboard Arduino from the board itself.

Despite that, what makes UDOO really different from other boards is the Android support. With the capability to run smoothly, this operating system is a great opportunity for novice or experienced Android developers because they can create a new kind of real-world application that is powered by the Android user interface, its powerful design patterns, and even by other developers' applications.

Developers can choose to write their real-word applications using the Linux operating system. In this case, they can write web services or desktop applications using many well-known programming languages such as Python, Javascript (Node.js), Php, and Java. However, we will focus on application development under Android.
The latest building block is related to all I/O components. UDOO could be purchased with an internal Wi-Fi and a Gigabit Ethernet, which are recognized by both Linux and Android. It also offers HDMI (High-Definition Multimedia Interface) output connection and is shipped with an integrated Transistor-Transistor Logic (TTL) to Low-Voltage Differential Signaling (LVDS) expansion slot so that developers can connect an external LVDS touch screen.

During the course of this book, it is assumed that you're connecting UDOO to an external monitor through the HDMI cable. However, if you own an external LVDS panel, you can proceed with the connection just before the Our first run section in this chapter. To let Android use the external panel, you should follow some steps that you can find in the official documentation at http://www.udoo.org/faq-items/how-do-i-set-up-my-lvds/.

Another great component that is officially supported is the camera module that is easy to plug in the board and can be used for projects that require computer vision or image analysis. The last integrated component is the audio card with a fully functional audio playback and recording through an external microphone.

The mix of these building blocks, together with the Internet access and many Android APIs, gives us the opportunity to build real-world applications that listen to the environment and talk to devices with a board that can take part in the Internet of Things.

### Downloading and installing Android

We have an idea about a list of UDOO components that we may use to start building amazing projects. However, before we continue, we need to configure our board to run the Android operating system and also our development environment so that we can start writing and deploying our first application.

All prototypes you build in this book are based on Android KitKat 4.4.2 that is the latest supported version at the time of this writing. During the course of this book you will build many projects that make use of the Android Support Library to ensure compatibility with newer Android versions that the UDOO board will support.
The UDOO board doesn't have internal storage or a built-in boot program because it relies on external storage, a microSD card, in which you can install a bootloader and a compatible operating system. The easiest way to create a bootable microSD card, is to download and copy a precompiled image, though it's possible to create a clean operating system using released binaries and kernel sources.

http://www.udoo.org/downloads/ points to the official UDOO download page that contains the links for all the available precompiled images.

Among Linux images, we can find and download the latest supported version of Android KitKat 4.4.2. As stated earlier, UDOO comes in two different versions with dual and quad processors so we must download the right version, according to the owned platform.

**Installing from Windows**

To install the Android image from Windows, you need some extra tools to unpack and copy the image into the microSD card. The downloaded .zip file is in 7-Zip compression format, so you need to install a third-party unarchive program such as 7-Zip. When the extracting process is done, we have an uncompressed .img file ready for copy on an empty card.

To write the uncompressed image into our microSD card, perform the following steps:

1. Insert your microSD card in the built-in slot reader or external card reader.
2. Format the card using the FAT32 filesystem.
3. To write the image on the microSD card, we need to use the Win32DiskImager tool. Download it from http://sourceforge.net/projects/win32diskimager/.
4. Run the application, but bear in mind that if we are using Windows 7 or Window 8.x, we have to right-click on Win32DiskImager.exe executable and be sure to select the **Run as administrator** option from the context menu.
5. Win32DiskImager is a tool that writes raw disk images using low-level instructions. This means that you need to exactly follow the next steps and be really sure that you correctly select the output device. If this option is wrong, you might lose all your data from an unwanted storage memory.
6. Once the application starts, you can see the main window, as illustrated in the following screenshot:

![Image File](image.png)

7. From the application's main window, in the **Image File** box, choose the .img file previously extracted.

8. Select the microSD drive accurately on the **Device** dropdown and bear in mind that if we use the wrong drive, we can destroy all our data on the computer's hard disk.

9. Click on the **Write** button and wait for the process to complete in order to have a bootable Android operating system in the microSD card.

### Installing from Mac OS X

To install the Android image from Mac OS X, we need a third-party tool to unpack the downloaded .zip file, because it's in 7-Zip compression format and we can't use the built-in unarchive software. We have to download software such as Keka, which is freely available at [http://www.kekaosx.com/](http://www.kekaosx.com/).

If we love the Mac OS X terminal, we can use the Homebrew package manager that is available at [http://brew.sh/](http://brew.sh/).

In this case, from the command line, we can simply install the p7zip package and use the 7za utility to unpack the file as follows:

```bash
brew install p7zip
7za x [path_to_zip_file]
```
To proceed and write the uncompressed image into our microSD card, perform the following steps:

1. Launch the Terminal application and enter into the folder in which we have downloaded and extracted the Android image. Assuming Downloads is the name of that folder, we can issue the following command:
   ```bash
   cd Downloads
   ```

2. Get the list of all mounted devices with the following command:
   ```bash
   df -h
   ```

3. The list of all systems and internal hard drive partitions will be similar to the following screenshot:

   ![Screenshot](image.png)

4. Connect the microSD card using the built-in or external card reader.

5. Format the microSD card through the Disk Utility application that is already available in our system. Launch it and select the correct disk from the list on the left.

6. From the main panel of the window, choose the Erase tab from the upper menu and select the MS-DOS (FAT) filesystem in the Format dropdown. When you are ready, click on the Erase button.

7. From the Terminal application, launch the previous command again:
   ```bash
   df -h
   ```
8. The list of mounted partitions has been changed, as we see in the following screenshot:

```
Filesystem  Size  Used  Avail  Capacity  Used  %Used  Mounted on
/dev/disk0s2 112Gi 107Gi 5.0Gi   96%  280986313 1313755 96%  /
devs 200Ki 200Ki 0Bi  100% 724 0 100% /dev
map-config 0Bi 0Bi 0Bi  100% 0 0 100% /net
map-auto_home 0Bi 0Bi 0Bi  100% 0 0 100% /home
/dev/disk1s1 7.4Gi 2.3Mi 7.4Gi  1% 0 0 100% /Volumes/EMPTY
```

9. We can assume that the missing device, during the first run, is our microSD card, so we have to bear in mind the new value under the **Filesystem** column. If you look at the previous screenshot, our partition name is /dev/disk1s1 while it isn’t /dev/disk0s2 because it’s our hard disk.

10. Once we have found the correct partition, we have to unmount it using the following command:

```
sudo diskutil unmount /dev/[partition_name]
```

11. To write the image into the microSD card, we must find the raw disk device so that we can erase and write the Android image into the card. Assuming that the partition name found before was /dev/disk1s1, the related raw disk will be /dev/rdisk1.

```
We are going to use the `dd` tool. This command writes raw disk images using low-level instructions. This means that you need to exactly follow the next steps and be really sure that you choose the correct disk device, because if it is wrong, you can lose all your data from an unwanted storage.
```

12. Write the image previously extracted into the microSD card using `dd` with the following command:

```
sudo dd bs=1m if=[udoo_image_name].img of=/dev/[raw_disk_name]
```

A full example of the previous command is as follows:

```
sudo dd bs=1m if=[udoo_image_name].img of=/dev/rdisk1
```
13. When we launch the command nothing seems to happen, but actually, `dd` is writing the Android image in the background. Once the process is complete, it outputs the transferred bytes report, as shown in the following example:

```
6771+1 records in
6771+1 records out
710656640 bytes transferred in 1395.441422 secs (5088466 bytes/sec)
```

14. Now we have our bootable Android operating system and we can eject the microSD card with the following command:

```
sudo diskutil eject /dev/[raw_disk_name]
```

## Installing from Linux

To install the Android image from Linux, we need a third-party tool to unpack the downloaded `.zip` file. Because the file is in 7-Zip compression format, we need to install the `p7zip` package porting from the command line using the package manager of our distribution. Then we can use the `7za` utility to unpack the file or any other graphical unarchiver that makes you comfortable.

We can proceed to write the uncompressed image into our microSD card using the following steps:

1. Open the Linux Terminal and enter into the folder where we have downloaded and extracted the Android image. Assuming the file is in our `Downloads` folder, we can issue the following command:

   ```
   cd Downloads
   ```

2. Attach the microSD card using the built-in or external card reader.

3. Find the correct device name through the following command:

   ```
   sudo fdisk -l | grep Disk
   ```

4. The output is a filtered list of all the devices found, and it contains, among others output lines, something like:

   ```
   Disk /dev/sda: 160.0 GB, 160041885696 bytes
   Disk /devmapper/ubuntu--vg-root: 157.5 GB, 157454172160 bytes
   Disk /dev/sdb: 7948 MB, 7948206080 bytes
   ```

   In this case, `/dev/sda` is our hard disk while `/dev/sdb` is our microSD card. If this is not your case and you are using an internal card reader, it's possible that the device is named `/dev/mmcblk0`. 
When you've found the right device name, keep it in mind so that we can use it later.

5. Find all the mounted partitions of the above device through the following command:

```
mount | grep [device_name]
```

6. If the previous command generates an output, find the partition name available in the first column of the output and unmount any listed partitions through the following command:

```
sudo umount /dev/[partition_name]
```

`dd` is a tool that writes raw disk images using low-level instructions. This means that you need to exactly follow the next steps and be really sure that you choose the correct disk device because, if it is wrong, you can lose all your data from an unwanted storage memory.

7. Write the image previously extracted to the above device name using the `dd` command:

```
sudo dd bs=1M if=[udoo_image_name].img of=/dev/[device_name]
```

Assuming `/dev/sdb` is our microSD card, the following is a full example:

```
sudo dd bs=1M if=[udoo_image_name].img of=/dev/sdb
```

8. When we launch the command nothing seems to happen, but actually, `dd` is writing the image in the background. Once the process is complete, it outputs the transferred bytes report, as follows:

```
6771+1 records in
6771+1 records out
7100656640 bytes transferred in 1395.441422 secs (5088466 bytes/sec)
```

9. Now we have our bootable Android operating system and we can eject the microSD card with the following command:

```
sudo eject /dev/[device_name]
```
Our first run
Once we have a bootable microSD card, we can insert it into our UDOO board, use our external monitor or LVDS panel and connect a mouse and a keyboard. After the power is switched on, the Android logo shows up and when the loading process is finished, we can finally see the Android home interface.

Setting up the development environment
Now that Android in our UDOO board is fully functional, it's time to configure the development environment. Every project we're going to build is composed of two different running applications: the first is the physical application composed of an Arduino program capable of controlling an external circuit through UDOO I/O pins; the second one, is an Android application that will run on the board and that deals with the user interface.

Because we have to write two different applications that interact with each other, we need to configure our development environment with two different IDEs.

Installing and using Arduino IDE
Before we can start to upload our programs, we need to install the microUSB serial port driver so that we can correctly communicate with onboard Arduino. The USB to the Universal Asynchronous Receiver/Transmitter (UART) driver, compatible with the CP210x converter available on the board, can be downloaded from http://www.silabs.com/products/mcu/pages/usbtouartbridgevcpdrivers.aspx.

Here we have to choose the correct version according to our operating system. Once the download is complete, we can extract the archive and double-click on the executable to proceed with the installation. When the installation process is complete, we may need to restart the system.

Now that the microUSB bridge driver is working, from the Arduino website, we have to download the IDE 1.5x beta because, at the moment, the beta version is the only one that supports Arduino Due boards. The link http://arduino.cc/en/Main/Software#toc3 points directly to the latest version.
To upload a new program, UDOO needs to receive the signals ERASE and RESET from the serial port before and after the upload, respectively. On the official Arduino Due board, this action is performed by the integrated ATmega16U2 microcontroller, which is missing in the UDOO board. The Arduino IDE will take care of this process, but if in the future you prefer to use another IDE, you will have to take care on your own.

Installing in Windows
To install in Windows, we have two different options: use the provided installer or use the archive file for nonadministrator installation. If we choose to use the installer, we can double-click on the executable. When the installer asks us which components we want to install, be sure to select all the checkboxes. If we choose to use the archive file instead of the installer, extract the file and put the result directory into your users folder.

Installing in Mac OS X
To install in Mac OS X, we need to download the archive version. If we run an OS X version greater than 10.7, we can download the Java 7 version. In any other cases, or if you are not sure, download the Java 6 version.

When we finish the download, we have to double-click on the archive to proceed with the extraction and then drag-and-drop the Arduino application icon inside our Applications folder.

Installing in Linux
To install in Linux, we need to download the archive version supported by our 32 or 64 bit architecture. Once the download has finished, we can extract the IDE and put it inside our home folder or in any other folder of your choice.
First launch

Now that we have finished configuring the communication driver and the IDE with the correct patch, we can launch and see the Arduino IDE, as shown in the following screenshot:

![Arduino IDE Screenshot](image)

Installing and using Android Studio

UDOO with an Android operating system acts like any other traditional Android device. This means we can use the standard toolchain, build system, and IDE used for the development of smartphones or tablets applications. At the moment, the available toolchains are related to two main IDEs: Eclipse and Android Studio.

Eclipse is an open source IDE with an advanced plugin system that allows you to easily extend many of its core capabilities. This brought Google to develop an Android Development Tool (ADT) plugin to create an integrated development environment in which developers can write, debug, and package their Android applications.
Android Studio is a more recent project whose first beta was released in May 2013 while the first stable release was in December 2014. Based on IntelliJ IDEA, a well-known Java IDE, it’s powered by the Gradle build system that combines the flexibility of Ant with the dependency management of Maven. All these characteristics, together with the increasing number of plugins, best practices, Google Cloud Platform integration, and third-party services integration such as Travis CI, make Android Studio a great choice for the development of future projects.

All Android projects covered in this book are built using Android Studio, and if you are a novice or an experienced Android developer and Eclipse is your usual IDE, this could be a great opportunity to try the new Android Studio.

The first thing to do is to download the latest version of Android Studio for your operating system from https://developer.android.com/sdk/.

When the download starts, we’ll be redirected to the installation instructions related to our operating system, and when we finish the installation, we can start the IDE. During the first run, the IDE will make all the required checks to retrieve and install the latest available SDK, virtual device, and build system to let you start developing the first application. In the Setup Wizard - SDK Settings page, be sure to select the Android SDK and Android Virtual Device components and then click on Next. In the next page, you should accept all Android licenses and then click on Finish.

Once the IDE is installed, we can start Android Studio. The following screenshot shows the main window when a project is not opened:
Running your first Android application

Now that Android is installed in our UDOO board and that all development environments are configured, we can start to write and deploy our first Android application. The following is the default pattern of other developers when they start to dive into a new technology. We are going to write and deploy a simple Android application that prints Hello World!.

To bootstrap our first project, perform the following steps:

1. In the main window of Android Studio, click on **Start a new Android Studio project**.
2. In the **Application name** field, type **HelloWorld**; in **Company domain**, write your domain or **example.com** if you don’t have any at the moment. Then, click on **Next**.
3. In the form factors selection window, select **Phone and Tablet** and choose **API 19: Android 4.4 (KitKat)** in the **Minimum SDK**. Then, click on **Next**.
4. In the add activity page, for the purpose of this hello world application, select a **Blank Activity** option and click on **Next**.
5. In the **Activity Options** page, write **HelloWorld** in **Activity Name** and click on **Finish**.

In next chapters, we will create applications from scratch, so we have to bear in mind the previous steps because we are going to repeat the process multiple times in this book.

Now Android Studio will start to download all Gradle requirements to prepare our build system. When the process is completed, we get our first HelloWorld application.

Without writing any lines of code, we have already created a deployable application. Now we have to connect our UDOO board using a microUSB to the USB cable. If we take a look at the board, we see two different microUSB ports. The first one on the left, that we will use in the next chapters, connects our computer to the serial port of both processors, so we can use it to upload the Arduino program to the UDOO microcontroller or we can use it to access to the Android system shell. The activated communication for the serial port depends on the state of J18 jumper, whether it’s plugged or not. Instead, the microUSB port on the right connects our computer to the i.MX 6 processor that runs Android and that we will use to upload our Android applications. You can find more information about the processor communication at the official UDOO website [http://www.udoo.org/features/processors-communication/](http://www.udoo.org/features/processors-communication/).
To connect our computer to the Android operating system for the application upload process, we need to use the right microUSB port colored in black in the following screenshot:

Like we do in a traditional Android application, we can click from the top menu on **Run** and then on **Run app**. At this point, we need to choose a running device, but unfortunately, our list of available devices is empty. This problem occurs because of how the internal communication between processors works.

After the boot time, the connection is enabled between the two processors and plugging the microUSB cable will produce no effect. This happens because Android doesn't use the internal UART serial port during the communication with Arduino. It uses the **USB On-The-Go (OTG)** bus that allows devices to act as host and let other components such as flash drive, mouse, keyboard, or Arduino, in this case, to connect through it.
The i.MX 6 processor is physically connected to the OTG bus while the other side of the bus is connected both to Arduino and to the external microUSB connector. The current active connection can be changed using a software piloted switch. When the external OTG port is enabled, Android can communicate with an external computer via microUSB port but it can't send back any data to onboard Arduino. On the contrary, when the external OTG port is disabled, Android can communicate with Arduino but the connection with the computer is interrupted.

The latter is our actual configuration and we need to switch the OTG port in order to enable the external communication with our computer to complete the application deployment. From Android, we have to go to the Settings menu and choose Developer options. There we need to select the External OTG port enabled checkbox. If the USB cable is connected, a popup will ask us to allow USB debugging. If this is our main computer, we may want to select Always allow from this computer and then click on OK. If this option is not checked, the popup will be shown every time we connect UDOO to our computer.

Bear in mind that every time we need to deploy our Android application, we need to enable the external OTG port. On the contrary, when our application is deployed and we need Android to communicate with Arduino, we need to disable the external OTG port.

Now that our computer can see the UDOO board as a traditional Android device, we can try to deploy our application again. This time, in the Choose Device dialog, we can find a Freescale UDOO Android device. Select it and click on OK. Our first deployment is complete and now we can see the HelloWorld application on the connected monitor.

**Summary**

In this chapter, we learned some UDOO characteristics that distinguish this board from the others. One of the greatest differences is related to the full support of the Android platform that led us to install and configure the latest supported version on the board.

We explored the necessary tools to start developing real-world applications and we configured our development environment to write Android applications and Arduino programs.

We had a brief introduction about how the communication between the two processors works and how we can switch the OTG port to enable external access for our first deployment. In the next chapter, we will start a new Android application from scratch that is capable of using and controlling a physical device built through a set of prototyping tools.
Where to buy this book

You can buy Getting Started with UDOO from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.