Android for the BeagleBone Black

Design and implement Android apps that interface with your own custom hardware circuits and the BeagleBone Black

Andrew Henderson  Aravind Prakash
In this package, you will find:

- The authors biography
- A preview chapter from the book, Chapter 1 ‘Introduction to Android and the BeagleBone Black’
- A synopsis of the book’s content
- More information on Android for the BeagleBone Black

About the Authors

Andrew Henderson has over 15 years of experience developing software for the Linux desktop and embedded Linux and Android systems. He is currently a PhD candidate at Syracuse University, with research interests in the areas of system security and dynamic analysis. He maintains multiple open source projects for the BeagleBoard and BeagleBone platforms.

I want to thank Cheryl, Olivia, and my father for all of their encouragement and support during the time that I spent researching and writing the material for this book. I would also like to thank Dr. Heng Yin, Dr. Wenliang Du, and Dr. Ehat Ercanli of Syracuse University for lending their knowledge and guidance to my Android OS and BeagleBone/BeagleBoard research.

Aravind Prakash is a PhD candidate at Syracuse University. His interests lie in system and mobile security, with emphasis on program analysis. He has published in multiple top-tier computer-security conferences. He brings with him over a decade of programming experience from companies such as Microsoft, McAfee, and FireEye.
Android for the BeagleBone Black

The broad availability of Android-based devices has generated a large amount of interest in developing software applications, or apps, that target Android. Luckily, a powerful and low-cost hardware platform is available that allows you to quickly and easily test your apps on real hardware: the BeagleBone Black. With a focus on small size and a wide variety of expansion and interfacing opportunities, the BeagleBone Black provides a lot of processing power at a very low price. It also provides an opportunity to app developers that once belonged only to those that were expert hardware hackers or owners of expensive hardware development kits: the chance to write Android apps that interact with custom hardware circuits.

Whether you are brand new to hardware interfacing or a seasoned expert, Android for the BeagleBone Black provides you with the tools that you need to begin creating Android apps that communicate directly with your custom hardware. From the very beginning, this book will help you understand Android's unique approach to hardware interfacing. You will install and customize Android, build circuits that interface with your BeagleBone Black platform, and build native code and Android apps that use that hardware to communicate with the outside world. By sequentially working through the examples in each chapter, you will learn how to create multithreaded apps that are capable of interfacing with multiple hardware components simultaneously.

Once you have explored the variety of example circuits and apps in this book, you will be well on your way toward becoming an Android hardware interfacing pro!

What This Book Covers

Chapter 1, Introduction to Android and the BeagleBone Black, walks you through the process of installing the Android OS to your BeagleBone Black board. It also provides you with a list of hardware components that you will need to perform the activities throughout this book.

Chapter 2, Interfacing with Android, introduces you to several aspects of the BeagleBone Black's hardware and Android's Hardware Abstraction Layer. It describes how to make a few modifications to both your development environment and Android installed on your BeagleBone Black to allow Android apps to access the various hardware features of the BeagleBone Black.

Chapter 3, Handling Inputs and Outputs with GPIOs, guides you through building your very first hardware interfacing circuit and explains the details of a basic Android app that can communicate with it. This is your first step toward building much more complex apps that interact with the world outside your BeagleBone Black.
Chapter 4, Storing and Retrieving Data with I2C, expands on the basics from Chapter 3, Handling Inputs and Outputs with GPIOs, and explains how asynchronous background threads within your apps are used to communicate with hardware. It guides you through building a circuit that interfaces a nonvolatile memory chip to the BeagleBone Black and the implementation details of an app that interacts with the chip.

Chapter 5, Interfacing with High-speed Sensors Using SPI, explores creating apps that perform high-speed interfacing using a temperature and pressure sensor interfaced to the BeagleBone Black.

Chapter 6, Creating a Complete Interfacing Solution, combines the lessons learned about GPIO, I2C, and SPI interfacing from the previous chapters to create a single, complex hardware and software solution that uses all three interfaces to react to hardware events that originate from the outside world.

Chapter 7, Where to Go from Here, describes a few more of the hardware interfaces available on the BeagleBone Black, explains how to create more permanent Android hardware/software solutions, and gives you a few ideas for future projects to explore.
In this book, you’ll learn how to install Android to a microSD card for use with the BeagleBone Black and create Android apps that interface with external hardware that is connected to the BeagleBone Black. You will develop software that receives input from the outside world via buttons and sensors, stores and retrieves data from external memory chips, and lights external LEDs. Better yet, you’ll learn how to do this in a flexible way that can be easily integrated into your apps.

As you explore the world of interfacing hardware with Android, you will discover that it encompasses many different areas of expertise. Understanding electronic circuits and knowing how to interface them with the BeagleBone Black, understanding the Linux kernel, and developing Android apps are a few such areas. Luckily, you don’t have to be an expert in these areas to learn the basics of interfacing hardware with Android. We have done our best to guide you through the examples in this book without requiring you to have an in-depth knowledge of the Linux kernel or electronics theory.

In this chapter, we will cover the following topics:

- Looking back on Android and BeagleBone Black development
- Shopping for the hardware essentials
- Learning about the hardware you’ll interface with
- Installing Android on the BeagleBone Black
Looking back on Android and BeagleBone Black development

The Android operating system has been taking the world by storm. Ever since its introduction to the world in a beta release in 2007, it has grown to become the dominant mobile phone OS. Aside from mobile phones, it has also been used for tablets (such as the Barnes & Noble Nook eReader and the Tesco Hudl tablet) and a variety of other embedded multimedia devices. The OS has added new features and evolved over the years, but it still has the same primary design principles as it did when it was first conceived. It provides a lightweight OS with a touchscreen interface that gives quick and easy access to multimedia applications while using minimal resources.

Aside from its general popularity, Android has a number of advantages that make it an excellent OS for your projects. The source code of Android is open source and freely available from http://source.android.com. It is free for you to use in any products that you create. Android uses the popular Linux kernel, so any expertise that you already have with Linux will aid you in your Android development. There is a well-documented interfacing API that makes developing for Android simple and straightforward.

The broad availability of Android-based devices has generated a large amount of interest in developing software applications, or apps, that target Android. It has become easier to develop Android apps. Eclipse **Android Development Tools** (ADT) allows app developers to prototype software and then execute that software within an emulated Android device environment. However, the emulated device differs from real hardware in subtle (and sometimes dramatic) ways in terms of speed and appearance. Luckily, a powerful and low-cost hardware platform is available that allows you to quickly and easily test your apps on real hardware: the BeagleBone Black.

The **BeagleBone Black** (BBB) hardware platform, produced by **CircuitCo** for the BeagleBoard.org nonprofit organization, is a newcomer to the open source hardware scene. First produced in 2013, this low-cost, ARM-based single board computer is an improvement over the original BeagleBone platform. The BBB is an improvement over the original BeagleBone board that offers increased processing power, built-in HDMI video, and either a 2 or 4 GB (depending upon the BBB's revision) on-board eMMC memory. With a focus on small size and a wide variety of expansion and interfacing opportunities, the BBB provides a lot of processing power at a very low price. The following image shows a typical BBB:
Android runs on the inexpensive BBB, which makes it an excellent hardware platform to use to explore Android and develop your own custom Android projects, for example, if you had an idea for an Android kiosk device, a hand-held gaming console, or some other multimedia device. The combination of Android and the BBB will allow you to prototype such devices quickly and cheaply.

Now that we have taken a quick look at the BBB and Android, let’s take a look at the hardware that you will need to make the most out of both of them.

**Shopping for the hardware essentials**

When you purchase your BBB, you will only receive the board and a USB cable to power and communicate with it. Before you begin any serious software development for hardware-interfacing projects with the BBB, there are a few additional pieces of hardware that you will need. In our opinion, the best place to purchase these items is AdaFruit (www.adafruit.com). Almost everything here is available from this single source, and their customer service is very good. In fact, many of the items listed here are available as a BeagleBone Black starter kit from AdaFruit (product ID 703). The starter kit does not contain a 3.3 V Future Technology Devices International (FTDI) cable, but it does include the BeagleBone Black itself.
The FTDI cable
A 3.3 volt FTDI cable (product ID 70) allows you to view all of the serial debug output of the BBB. If you are performing any serious development, you must have one of these cables. The cable is necessary if you wish to observe the boot process of the BBB (both bootloader and kernel output as the system is initialized), and it also provides a console shell into Linux and Android. This shell helps you to troubleshoot boot problems, as you will always have a method of interacting with the system when a network connection is unavailable, or when no communication services are up and running.

Power supply
While the BBB can be powered via the USB cable, this method supplies barely enough power to run the BBB. If you are using external capes, or are otherwise attaching external circuits that draw power from the 5 volt pins of the BBB, you must use an external power supply. BeagleBoard.org specifies that the power supply must be a 2 amp, 5 volt DC power supply with a 2.1 mm barrel connector that is center positive. AdaFruit sells a power supply that conforms to the BBB's requirements (product ID 276).
Breadboard and the mounting plate

Experimenting with electronics becomes much simpler if you are able to easily and quickly construct circuits without the worry of soldering. Because of this, we recommend that you invest in a breadboard and some breadboarding jumper wires (product ID 153). Your breadboard doesn't have to be anything big or fancy, but you should use at least a standard half-size breadboard (product ID 64) for the projects given in this book.

AdaFruit Proto Plate (product ID 702) is an additional item that we recommend that you purchase. Proto Plate is a plastic plate that both the BBB and a half-size breadboard mount onto. This helps you avoid accidentally stretching or disconnecting the wires that connect electronic circuits to the BBB. Using Proto Plate makes relocating your BBB and breadboard simple and painless.

MicroSD cards

If you do much work with the BBB, you'll always want to have a few extra microSD cards around! Android will fit onto an 8 GB microSD card with plenty of free space available to hold your own apps. You can write an Android image to a larger microSD card, but most premade Android system images will only consume the first 4–8 GB of space on the card. As most laptops and desktop PCs don't directly accept microSD cards, you should own at least one microSD-to-SD card adapter. Luckily, one of these adapters is typically packaged with each microSD card that you buy.

Learning about the hardware you'll interface with

The best way to learn about interfacing Android software with hardware is to learn while having real hardware components connected to your BBB. This way, your software will talk to actual hardware and you can directly observe how your apps respond to physical interaction with your system. We have selected a variety of electronic components that will be used throughout the book to demonstrate various aspects of hardware interfacing. You are welcome to use as many or as few of these components as your interests and budget permit. It can be expensive to purchase all of these components at once, but make sure to buy all of the components necessary for each chapter if you are interested in implementing the examples in that chapter.
General-purpose components

In Chapter 3, Handling Inputs and Outputs with GPIOs, and Chapter 6, Creating a Complete Interfacing Solution, you will use a variety of electronic components such as pushbuttons, LEDs, and resistors to interface with the BBB. Many of these items can be purchased from any electronics supplier, such as DigiKey (www.digikey.com), Mouser Electronics (www.mouser.com), and SparkFun (www.sparkfun.com). Both Digikey and Mouser offer so many variants of each available component that it can be difficult for an inexperienced hardware hacker to pick the right components to buy. Because of this, we will recommend a few products from SparkFun that will give you suitable components needed to complete the exercises in this book. You are welcome to select your components from another supplier if using a different one is more convenient for you.

Our examples require only three components: a resistor, a pushbutton switch, and an LED. We suggest purchasing a 1K ohm, 1/6 (or 1/4) watt resistor (part #COM-08980), a 12 mm push button switch (part #COM-09190), and any small LED (3–10 mm in size) that can be triggered by around 3 volt or less (part #COM-12903 is a good assortment of 5 mm LEDs).

The AdaFruit memory breakout board

In Chapter 4, Storing and Retrieving Data with I2C, and Chapter 6, Creating a Complete Interfacing Solution, you will interface with a 32 KB Ferroelectric Random Access Memory (FRAM), which is a nonvolatile memory IC, to store and retrieve data. We have selected AdaFruit Breakout Board (product ID 1895) that contains this IC. The breakout board already contains all of the necessary components to interface the IC to the BBB, so you need not worry about many of the low-level details involved in creating a clean, noise-free connection between each IC and the BBB.

The FRAM Breakout Board with its header strip (source: www.adafruit.com)
The AdaFruit sensor breakout board

In Chapter 5, Interfacing with High-speed Sensors Using SPI, and Chapter 6, Creating a Complete Interfacing Solution, you will interface with a sensor IC to receive environmental data. We have selected an AdaFruit breakout board (product ID 1900) that contains these ICs. These breakout boards already contain all of the necessary components to interface the ICs to the BBB, so you need not worry about many of the low-level details involved in creating a clean, noise-free connection between each IC and the BBB.

Preparing the breakout boards

Each breakout board comes with a header strip. This header strip must be soldered into each breakout board so that they can be easily connected to the breadboard. This is the only soldering that is required to complete the exercises in this book. If you are unfamiliar with soldering, there are numerous tutorials online that explain techniques for effective soldering. If you feel uncomfortable soldering the header strips, ask a friend, instructor, or colleague to assist you with the process.

A few online soldering tutorials that we suggest that you check out are:
- https://www.youtube.com/watch?v=BLfXXRfR1zY
- https://learn.sparkfun.com/tutorials/how-to-solder---through-hole-soldering

Installing Android on the BeagleBone Black

The Android OS is a complex piece of software that is constructed out of many components built from a very large codebase. It can be a difficult and time-consuming task to build Android from source, so you will be using a premade Android image from the BBBAndroid project (www.bbbandroid.org) throughout this book.

BBBAndroid is a port of Android Open Source Project (AOSP) KitKat Android to the BBB. There are a few different distributions of Android available for the BBB, but we selected BBBAndroid because it uses the 3.8 Linux kernel. This kernel includes the Cape Manager (capemgr) functionality as well as a few other tools that will assist you in interfacing hardware to Android apps. Other flavors of Android on the BBB use the 3.2 Linux kernel, which is much older and lacks capemgr support. Chapter 2, Interfacing with Android, discusses the capemgr functionality in more detail. The 3.8 kernel is a good balance between enabling the newer features for the BBB while avoiding any potentially unstable, cutting-edge features.
The BBB can boot its OS in a few different ways:

- **Onboard eMMC**: The OS resides within the onboard eMMC storage. The Angstrom or Debian OS that comes installed on your BBB boots from the eMMC out of the box.

- **MicroSD card**: The OS resides on a microSD card that is inserted into the BBB. If a bootloader is installed on the microSD card, the bootloader installed on the onboard eMMC notices the presence of the microSD and will boot from that instead. In addition, booting from the microSD card is forced when the user boot button is held down during BBB power up.

- **Over the network**: The bootloader is capable of downloading a kernel over the network via TFTP. The OS can actually be downloaded at boot time though this is usually only done during commercial product development. This is an advanced feature that is outside the scope of this book.

The BBBAndroid image is designed to be written to and booted from a microSD card. As the image creates a fully bootable system on the microSD card, you will not have to hold the BBB’s user boot button during power on to boot into Android. Simply insert the microSD card into the BBB and you’ll boot into Android automatically.

Using a microSD card-based OS is advantageous for us because you can easily mount the card on a Linux PC to modify the Android filesystem as you see fit. If the OS is installed in the eMMC, it can be hard to access the OS to change arbitrary files in the filesystem. The system must be running to access the eMMC contents, so making a change that corrupts the system or makes it unbootable makes accessing the eMMC to fix the problem difficult.

**Downloading a premade Android image**

The main page of the BBBAndroid website provides a download link for the most recent premade image. Like any open source project, details about the version number and size of each image are likely to change over time as bugs are found and changes are made. However, the latest and greatest will be available via the website.

BBBAndroid's images are compressed using the xz compressor utility to save time when downloading, so the image must be decompressed prior to writing it to a microSD card. The tools used to decompress and write the image will vary depending upon the OS that you are using. While the compressed image might only be a few hundred MB in size, the uncompressed image will be 8 GB.
Prior to beginning the decompression of the image, make sure that you have enough hard drive space to hold the uncompressed image.

Creating your Android microSD card using Windows

Under Windows-based OSes, the compressed image can be uncompressed using tools such as 7-Zip or WinRAR and then written to the microSD card using the tool Win32 Disk Imager. All of these tools are freely available for download. To prepare an Android microSD card, follow these steps:

1. For this example, you'll use the WinRAR application. Download WinRAR from www.rarlab.com and install it. WinRAR will integrate with the Windows Explorer shell of the Windows desktop.

2. Download and install the Win32 Disk Imager application. It is available from the project's SourceForge page at http://sourceforge.net/projects/win32diskimager.

3. Right-click on the BBBAndroid image that you downloaded and select the Extract here option on the Explorer shell context menu. An uncompressed version of the image (8 GB in size) will be written to the same location as the compressed image. The decompression process might take several minutes.

4. Insert an 8+ GB microSD card into the system. The card will be detected by Windows as having a valid filesystem on it if it came preformatted (most cards are sold preformatted for your convenience). Irrespective of whether the card is formatted or not, a drive letter is assigned to it by Windows.
5. Browse to This PC and examine the devices shown under Devices and drives. The card should be shown. Make a note of the drive letter assigned to the card.

![Devices and drives (3)](image)

The microSD card will be shown with a drive letter under Windows (drive E in the image).

6. Launch Win32 Disk Imager. Enter the filename and path to the uncompressed image in the text field, or click on the folder icon to navigate to the file's location. Change the Device drop-down box to the drive letter of the microSD card that you identified in step 4.

![Win32 Disk Imager](image)

Win32 Disk Imager with the image file specified (note that the drive letter matches that of the microSD card).

7. Writing the image will take several minutes. Once the write has completed, remove the microSD card from your computer and insert it into your BBB.

8. Power on the BBB and Android will begin to boot. On the first boot, it will take several minutes for the top-level UI screen to appear. On subsequent boots, it will take only 30 to 60 seconds to reach the top-level UI screen.

Congratulations! Your BBB is now running the Android OS.
Creating your Android microSD card using Linux

Under Linux, the compressed Android image can be uncompressed using the `xz` command and written to the microSD card using the `dd` command. To prepare an Android microSD card, follow these steps:

1. Make sure that you have `xz` installed. For systems using `apt-get`, try installing the `xz-utils` package:
   
   ```
   $ sudo apt-get install xz-utils
   ```

2. Decompress the image using `xz`. Substitute the name of your image file (with the `.xz` file extension), as shown in the following command:
   
   ```
   $ xz --decompress [IMAGE FILENAME]
   ```

3. Once uncompressed, the image will lose its `.xz` file extension and have a size of 8 GB. Insert your microSD card into the computer. A device in the `/dev` directory will be assigned to your card. To determine which device it is, use `fdisk`:
   
   ```
   $ sudo fdisk -l
   ```

4. The `fdisk` utility will display all storage devices currently connected to your computer. One of the devices will report as being the same size as the microSD card. For example, if you insert an 8 GB microSD card, you will see something similar to this:
   
   ```
   Disk /dev/sdb: 8018 MB, 8018460672 bytes
   ```

   The exact storage capacity of the card varies slightly between manufacturers, but the size is approximately 8 GB. The device assigned to this card is `/dev/sdb`. Other devices listed by `fdisk` will be secondary storage devices (such as your hard drive). Before proceeding any further, make certain that you have identified the proper device file that belongs to your microSD card. If you select the wrong device, you will destroy the filesystem on that device!

5. Write the image to the microSD card using `dd`. Assuming that the device you identified in step 5 is `/dev/sdb`, use the following command to perform the write:
   
   ```
   $ sudo dd if=[NAME OF IMAGE] of=/dev/sdb bs=4M
   ```

6. Writing the image will take several minutes. Once the write has completed, remove the microSD card from your computer and insert it into your BBB.
Power on the BBB and Android will begin to boot. On the first boot, it will take several minutes for the top-level UI screen to appear. On subsequent boots, it will take only 30 to 60 seconds to reach the top-level UI screen.

Congratulations! Your BBB is now running the Android OS.

**Summary**

In this chapter, you learned about the hardware that you’ll need to develop software for the BeagleBone Black, the electronics components and devices that you will need for the exercises in this book, and how to install an Android distribution onto a microSD card for use on the BBB. In the next chapter, you will learn how Android interacts with hardware at the software level and how the BBB can be configured to interface with the hardware components that you will be using in this book.
Where to buy this book
You can buy Android for the BeagleBone Black 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.