Making a Derivative

Alicia Gibb

“Its province is to assist us in making available what we are already acquainted with.”

—Ada Lovelace, on the Analytical Engine

This chapter gives an example of the source files and a physical object that you can copy, modify, make, and sell as a derivative under the Open Source Hardware Definition. This chapter first discusses derivatives and attribution, and then walks through a simple open source hardware kit named Blinky Buildings that readers are encouraged to alter or modify. Appropriate methods for creating a derivative are discussed. (The Blinky Buildings hardware kit can be purchased at www.bit.ly/blinkybuildings or at www.Sparkfun.com.) Readers can follow along with the instructions, thereby making their own derivative kit. You may have also noticed that this kit is referenced in other chapters throughout this book. The skills used in creating a derivative board consist of modifying the source files and understanding how to appropriately label derivative files and give credit. The Blinky Buildings kit is labeled with the open source hardware logo, meaning it is okay to copy and create derivatives from it. If you attempt to copy and create derivatives of hardware that is not open source, you may receive a cease and desist letter from the originating company. To be safe, look for the open source hardware logo, and stick to creating derivatives from what you know to be open.

Derivatives and Open Source Hardware

One of the reasons people open source their hardware is to allow derivatives to be built from that hardware. People create derivative hardware for many different reasons, ranging from personalized features to economic advantage. The Open Source Hardware Definition makes the following statement about derived works:

4. Derived Works. The license shall allow modifications and derived works, and shall allow them to be distributed under the same terms as the license of the
original work. The license shall allow for the manufacture, sale, distribution, and use of products created from the design files, the design files themselves, and derivatives thereof.

Cleary stated in the definition is the approval to create hardware from the original design files, to make copies and distribute the design files themselves, or to create a derivative from the original design. Because open source hardware grants the right to make copies, the terms “clone” and “counterfeit” get thrown around a lot when talking about derivative works. Here are the definitions of these open source hardware terms when referencing derivatives:

**Derivative:** A derivative is open source hardware that has been altered or modified but is based on an original design by another person or company.

**Clone or Copy:** A clone or copy is an open source hardware product that has been directly copied and conforms with the Open Source Hardware Definition because it does not infringe on the trademarks of other companies.

**Counterfeit:** With a counterfeit piece of open source hardware, the trademark has been copied onto a clone or derivative piece of hardware and does not abide by the Open Source Hardware Definition because the trademark is not owned by the person or company creating the derivative. Proper attribution does not include copying trademarks. Copying trademarks is illegal.

There are many examples of open hardware derivatives. In particular, the 3D printing and Arduino communities are great places to find open hardware and their derivatives. Keep in mind that Arduino itself is a derivative of Wiring, developed by Hernando Barragan, and Processing, developed by Ben Fry and Casey Reas. Some derivatives have small changes from the original; others have large changes. Changes for derivatives generally fall within four categories: (1) The function of the device is altered; (2) the form of the device is modified; (3) the change is economic, with the creator selling the same product at a different—usually lower—price point; or (4) the change enables a better design for manufacture (DFM), making it easier to manufacture or supply parts. The economic and DFM changes often go hand in hand and can be difficult to separate. All of these changes are permitted within the Open Source Hardware Definition, including a combination of the four.

An example of a board that changed drastically in both form and function is the LilyPad, which was created by Leah Buechley. The LilyPad was mashed up with the Arduino board, altered in both form and function so that it could be sewn into textiles. This particular derivative was quite extreme in the amount of changes made to the original Arduino hardware. The reason the alterations were so drastic was that Leah invented a sewable microcontroller prior to the development of the Arduino product. (For more on the history of the LilyPad, see the anecdote in Chapter 9.) When Leah’s design was put together with the Arduino board, one could argue that the Arduino’s shape, the form factor of pinouts, the thickness of the PCB, the typical construction materials used, and the main purpose of the board were all altered. This particular Arduino derivative’s function was to be embedded in wearables—a vastly different use than the Arduino team had previously imagined for their
microcontroller. The circular, thin (not to mention purple) LilyPad is to be sewn into wearables with a needle and thread rather than solder and wire.

Of course, not all derivatives are this different. In fact, some are even more or less copies of the original.

Let’s take the Arduino example one step further by considering a derivative of the derivative. Adafruit’s Flora is a derivative of the LilyPad (which is derivative of the Arduino board). The Flora derivative has the same form factor as the LilyPad—it is circular in shape and flat, and has copper petals around the exterior for ease of sewing—but has a different function, with a different chip being on board than is found in the original LilyPad. The Flora hardware introduced the ATmega32U4 chip into wearables with different functionalities than the ATmega328 on the LilyPad (such as allowing for a USB hookup rather than using an FTDI cable). Because these designs are all open source, the LilyPad developer was then able to roll the Flora’s changes back into their design, and now LilyPad also offers an ATmega32U4 product. Naturally, both these products can compete in the marketplace, because they are open source hardware, nobody is suing over rights; rather, everyone is focused on innovating. You can access the source files for LilyPad and Flora and compare and contrast the design files for yourself:

Original LilyPad files linked from SparkFun’s product page: www.sparkfun.com/products/9266

Flora derivative files listed in Github: github.com/adafruit/Adafruit-Flora-Mainboard

New ATmega32U4 LilyPad design rolling in the Flora’s ATmega32U4 improvements: https://www.sparkfun.com/products/11190

This is how derivatives of open source work! People build off improvements and ideas from others rather than reinventing the wheel each time. This process moves innovation forward at a more efficient and more productive pace.

Why the LilyPad Arduino Has “Arduino” in Its Name

The fact that the LilyPad carries the Arduino brand name is a very important point to note. The name Arduino is a trademark held by the Arduino company. Leah Buechley made an agreement with the Arduino company to license its Arduino trademark for a fee. This arrangement should not be confused with Leah giving the Arduino team attribution for their original board. Arduino has tried to make an important distinction in its trademark over the years. Although it is an open source project, the logo and company name are trademarked, much as any other company in the open source hardware space (and even in open source software, for that matter) can obtain a trademark for its products. We use trademarks because trademarks protect consumers and say something about the quality of the brand they are buying, rather than to protect the intellectual property of the hardware. Unless you obtain a license from Arduino, as Leah did to enable her project to be called a LilyPad Arduino, you cannot use the word “Arduino” in the name of your derivative as a way to give credit or attribution because it is a trademarked name.¹ You can

help the community understand correct attribution of Arduino derivatives by attributing Arduino in your README file or your project description.

**Giving Correct Attribution**

The Open Source Hardware Definition states the following about attribution:

The license may require derived documents, and copyright notices associated with devices, to provide attribution to the licensors when distributing design files, manufactured products, and/or derivatives thereof. The license may require that this information be accessible to the end-user using the device normally, but shall not specify a specific format of display. The license may require derived works to carry a different name or version number from the original design.

When creating your derivative, you will want to give credit to the original design without infringing on the trademark of one of original creator. As Michael Weinberg reminds us in Chapter 3, “Including a ‘share alike’ provision in a CC license is not a polite request that anyone who builds upon the work contribute back to the commons; rather, it creates a legal requirement.” This goes for attribution provisions as well. Due to the murky nature of licensing hardware, we tend to read the source files (which can be licensed cleanly with copyright or a copyright alternative) to understand the intention to list attribution or share it alike with the same license.

Attribution is like citing someone else’s work in a research paper; it is not copying and pasting the logo of the original creator and applying it to your board. Attribution can also be thought of as giving the work provenance. In the art world, giving correct provenance means identifying who had a particular piece of art before you owned it. In open source hardware, the equivalent is who hacked on that particular design file or piece of hardware before you. List their names just as you would in a citation or provenance document.

**Ego or Accuracy?**

Call it ego or call it accuracy, but the open source community loves credit. Credit, or attribution, is one of the many benefits to sharing your project openly. Getting attribution for something you created is at the root of most open source licensing structures, be it in hardware or software.

Accurate attribution is important to the life of your project. Giving accurate attribution lets the community know what your project was built on. Contributors, be they original creators or makers of derivatives, may be known within the community for their quality, work style, community involvement, approach, knowledge on a particular subject, and so on. Listing creators for your derivative gives users more information and certain expectations about your derivative.

How far do you go back? Most projects don’t include credit to the inventors of the transistor when using one on their board, or to the inventors of the C programming language when using Arduino. That practice is accepted within the community. We generally do not step further back than the first or second layer of original creators, although there
will always be gray areas where credit is due. When in doubt, give credit. Even if your project no longer reflects any of the original design, you still may want to reference that previous versions were based on so-and-so’s contraption so that people do not feel left behind or forgotten. No one will fault you for giving too much credit to other people who wrote code or built hardware before you. Perhaps the open source hardware industry will eventually grow in such a way that our README files will start to look like movie credits and go on for at least seven minutes after the movie is over.

**Blinky Buildings Project**

The Blinky Buildings project is a simple kit that you can use as an example of how to create an open source hardware derivative. My intention in creating this kit was to ensure that the community has something to experiment with and gives them the rights to create their own derivative. The goal of this kit is to inspire different derivatives of buildings, which together create a whole world of Blinky Building kits. My Blinky Building kit is shaped like the Empire State Building (Figure 6.1a); in its enclosure (Figure 6.1b), yours

![Blinky Buildings: Empire State Building (b) with enclosure.](image)

(Source: Image CC-BY-SA Alicia Gibb)
can be shaped like a different building, city, or landscape structure. Your derivative Blinky Building may include any of the four alterations discussed earlier: modify the shape of the original, modify the function of the original, modify the economics, modify the DFM, or make your own copy.

**Source Files**

This section walks through which pieces of other people’s open source material I used to create my kit; it also explores my source files that are shared with you. The source files include a circuit board created with the free version of Eagle and a 3D printing file for the enclosure. You can find all these files at www.bit.ly/blinkybuildings or in Appendix F. You will need PCB layout software, such as the following options, to be able to replicate or build off the derivative file:

- Fritzing
- Eagle
- KiCad

You will also need a 3D printing software if you choose to print out or modify the enclosure:

- Blender (reviewed in Chapter 8)
- OpenSCAD
- SketchUp

When making an open source hardware project, the most important thing to consider is whether people can rebuild the project from your source files. If so, you have a successful open source hardware project! If not, you need to release more source code or include more documentation.

As described in Chapter 5, I started my project by laying out the design process. My design purpose was to elegantly blink 20 LEDs in the shape of the Empire State Building. Given the scope and the specifications and requirements, I decided I would need a small, low-cost chip and would have to charlieplex the LEDs to drive 20 of them.

---

2. Fritzing is an open source project licensed under GNU GPL v3, which can take you all the way from a schematic to making the PCB.
3. Eagle offers a freeware version of its software that can be used for boards as simple as this example but that is closed source software. However, this software is quite accessible in the open source hardware community, as the majority of users are familiar with Eagle.
4. KiCad is open source software for electronic designs such as schematics and PCB layout that is licensed under GNU GPL v2.
I discovered a project close to my needs that charlieplexed 20 LEDs in a falling snowflake pattern. The file was licensed as CC-BY-SA. This designation means the schematic can be copied or used for a derivative, but the new schematic must give attribution to the original and must also share alike with the same terms. In addition, this schematic came with recommended code, also licensed as CC-BY-SA. Before I did anything else, I contacted both of the original designers—the hardware schematic designer and the code author—and asked if it would be okay to make a derivative of their work and include that derivative in my book. The open source hardware definition does not require this step, but the best practices recommend it.

I started with the schematic in Figure 6.2, which was created by Davy Uittenbogerd (daaf84). The file can be opened with Fritzing: fritzing.org/projects/charlieplex-snowfallshooting-star-20-leds.

A link to the code to run this circuit is included on the Fritzing page. The code was written by Geoff Steele (strykeroz). This code can be found in this GitHub repository: github.com/strykeroz/ATTiny85-20-LED-snowflakes.

When I copied and altered the code, I added a statement at the top of the code (known as the comment block) explaining where the original code was downloaded from and who the original author was: Geoff Steele. This gives Geoff attribution. I added a brief statement about which parts of Geoff’s code I altered. I included comments throughout the code when I changed something as well. Geoff included which pin numbers correlate with which color of wire on the Fritzing schematic in the code. It is good practice to include basic instructions for the hardware pinouts in the comment block.

Here are the altered chunks of code, including the attribution in the comment block. To view the full code, refer to www.bit.ly/blinkybuildings.

```c
/* downloaded from http://code.google.com/p/avr-hardware-random-number-generation/ Original code by Geoff Steele. Alicia Gibb altered the code by commenting out the fade functions so the building blinks LEDs on and off rather than fade LEDs on and off.

The original code is still all there if others wanted to keep playing with it; just take out the duty cycle comments.

The delays have also been changed, but can easily be reinstated by looking at the original code:

*/

I explained each of my code alterations by commenting that I altered the code from the original. I used a charlieOFF command rather than the original charlieON command in line 121.

```
Figure 6.2. Schematic by Davy Uittenbogerd drawn in Fritzing.
(Source: Image CC-BY-SA Davy Uittenbogerd)
I explained in another portion of the code that I commented out the code dealing with the time delays for fading an LED so it blinks rather than fades:

```c
    current++;  
    if(current==23) { ///// start over
        //Alicia commented out the below code to make the LEDs blink on and off rather than fade out.
        ///// now fade out the snowflake in that final position #19
        for(int dutyCycle = 3; dutyCycle <= 15; dutyCycle += 3) {
            //loopCount = 0;
            //timeNow = millis();
            //while(millis() - timeNow < (displayTime+current*2)) { ///// fade out as slow as animation has achieved by now
                loopCount++;
                if(!(loopCount % dutyCycle)) charlieON(19);
                else charlieOFF(19);
            // }
        }
    }
```

Once I had the code working on a bread-boarded prototype, I drew the schematic in Eagle following the Fritzing diagram. My schematic in Figure 6.3 is licensed as Creative Commons—By—Share Alike (CC-BY-SA), because the original schematic was licensed as CC-BY-SA. Due to the share-alike license, I must share it the same way. Any derivatives of this kit must also be shared alike as well, with the same Creative Commons license (CC-BY-SA) attached to the source files.

From the schematic, I created a board layout in Eagle (Figure 6.4) to be shaped like the Empire State Building. For instructions on how to give PCB boards an interesting shape, read Chapter 7. This board file is covered by a CC-BY-SA license: Because the schematic was posted under a share-alike license, and the board file is generated from the schematic, I must share it the same way. Note, however, that because Davy Uittenbogerd did not create this particular Eagle schematic or board file, it is licensed as CC-BY-SA Alicia Gibb and I will give him attribution in the README file and the product description. Labeling Davy as the creator at this point would cause confusion as to who produced and manufactured this product.

**Bill of Materials**

I am creating a kit for my Blinky Building that users will put together themselves. Since this is a kit, the bill of materials (BOM) will not go to a manufacturer, and it is not as detailed as the examples in Chapter 14. Generally, for a simple do-it-yourself (DIY) kit, the BOM serves the purpose of telling people what is in each kit. If the parts are standard, general parts that you could find at any hackerspace, there is no need to go into greater detail than the information shown in Table 6.1. In other documentation, it is advisable to include the data sheet of your chip as well.
Figure 6.3  Schematic drawn in Eagle.
(Source: Image CC-BY-SA Alicia Gibb, derived from Davy Uittenbogerd’s Fritzing diagram)
Figure 6.4  Blinky Buildings Board file in Eagle.
(Source: Image CC-BY-SA Alicia Gibb)

Table 6.1  BOM List

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Description</th>
<th>Package</th>
<th>Vendor Part Number</th>
<th>Manufacturer Part Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Empire State PCB</td>
<td>Dimensions 1.7” x 2.95”</td>
<td>Golden Phoenix</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Round white diffused LED 8K MCD</td>
<td>3 mm</td>
<td>EBay</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>RES 680 ohm 1/4W 5% carbon film</td>
<td>Axial</td>
<td>Digike CF14JT680RTR-ND</td>
<td>CF14JT680R</td>
</tr>
<tr>
<td>1</td>
<td>8-bit microcontroller: MCU</td>
<td>PDIP-8</td>
<td>Mouser 556-ATTINY85-20PU</td>
<td>Attiny85-20PU</td>
</tr>
<tr>
<td>1</td>
<td>Switch micro-mini slide 30V</td>
<td>Through hole</td>
<td>Digike 679-1854-ND</td>
<td>MMS1208</td>
</tr>
<tr>
<td>1</td>
<td>Holder cell 2032 w/gold pins</td>
<td>Through hole</td>
<td>CTECHi BH32T-C-G-ND</td>
<td>BH32T-C-G</td>
</tr>
<tr>
<td>1</td>
<td>Battery lithium coin 3V 20 mm</td>
<td>CR2023</td>
<td>Digike P189-ND</td>
<td>BH32T-C-G</td>
</tr>
</tbody>
</table>
Chapter 6  Making a Derivative

In addition to my README file in Appendix F, this concludes the source files for this kit. With the schematic, board layout, BOM, and README file, others should be able to reproduce my Blinky Buildings Empire State kit.

Cost–Benefit Analysis of Suppliers

For complete transparency, the following lists include the suppliers from which I received quotes for each item listed on the BOM. Listing the results of the cost–benefit analysis of supplies is not required for open source hardware, but in teaching people how to make a derivative, it is important to know some economics behind what is created. As stated in Chapter 15: Business, typical mark-up on hardware is between 2.6 to 4 times your BOM

<table>
<thead>
<tr>
<th>Board Manufacturer</th>
<th>Price/Unit</th>
<th>100 Pieces</th>
<th>Timeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold Phoenix PCB</td>
<td>$3.11</td>
<td>$311</td>
<td>5 day turn + 8 day shipping</td>
</tr>
<tr>
<td>OHSPark</td>
<td>$5.00</td>
<td>$500</td>
<td>4 week turn + shipping</td>
</tr>
<tr>
<td>Advanced Circuits</td>
<td>$4.33</td>
<td>$433</td>
<td>4 week turn + shipping</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parts: ATtiny85</th>
<th>Price/Unit</th>
<th>100 Pieces</th>
<th>Timeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digikey</td>
<td>$1.95</td>
<td>$195.00</td>
<td>3–4 day shipping</td>
</tr>
<tr>
<td>Mouser</td>
<td>$0.75</td>
<td>$75.00</td>
<td>3–4 day shipping</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parts: Resistors</th>
<th>Price/Unit</th>
<th>500 Pieces</th>
<th>Timeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digikey</td>
<td>$0.008</td>
<td>$8.00</td>
<td>3–4 day shipping</td>
</tr>
<tr>
<td>Mouser</td>
<td>$0.33</td>
<td>$165.00</td>
<td>3–4 day shipping</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parts: LEDs</th>
<th>Price/Unit</th>
<th>2000 Pieces</th>
<th>Timeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digikey</td>
<td>$0.209</td>
<td>$418.50</td>
<td>3–4 day shipping</td>
</tr>
<tr>
<td>Evil Mad Scientist</td>
<td>$0.20</td>
<td>$400.00</td>
<td>3–4 day shipping</td>
</tr>
<tr>
<td>EBay: LED shop 2010</td>
<td>$0.02</td>
<td>$40.00</td>
<td>2–4 weeks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parts: Batteries Holders</th>
<th>Price/Unit</th>
<th>100 Pieces</th>
<th>Timeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digikey</td>
<td>$0.60</td>
<td>$60.00</td>
<td>3–4 day shipping</td>
</tr>
<tr>
<td>CTECHi</td>
<td>$0.35</td>
<td>$35.00</td>
<td>5–7 day shipping</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parts: Batteries</th>
<th>Price/Unit</th>
<th>100 Pieces</th>
<th>Timeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digikey</td>
<td>$.28</td>
<td>$28.00</td>
<td>3–4 day shipping</td>
</tr>
<tr>
<td>CTECHi</td>
<td>$.35</td>
<td>$35.00</td>
<td>5–7 day shipping</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parts: Switch</th>
<th>Price/Unit</th>
<th>100 Pieces</th>
<th>Timeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digikey</td>
<td>$.96</td>
<td>$96.00</td>
<td>3–4 day shipping</td>
</tr>
<tr>
<td>SparkFun</td>
<td>$1.20</td>
<td>$120.00</td>
<td>Pick up in CO or 3–4 day shipping</td>
</tr>
</tbody>
</table>
costs. I went with the cheapest possible BOM and marked up the Blinky Buildings kit 2.6 times the BOM.

### Note

Pricing per unit were calculated based on the number of pieces being ordered. For example, a single switch at Digikey was $1.20, but buying 100 units brought the price down to $0.96.

---

**Anecdote: Enclosure Case for Blinky Buildings**

**Jason Brownstein**

An enclosure was made for the supplied Blinky Buildings board (Figure 6.5), in an effort to further illustrate the process of making a derivative including a 3D printed piece. This anecdote describes the design process and materials used. The enclosure described here was made using 3D modeling software, slicing software was used to produce G-code, and finally the enclosure was printed on an open source 3D printer. Chapter 8 provides links to the open source 3D modeling and slicing software packages.

![3D printing the enclosure.](Image CC-BY-SA Jason Brownstein)
When stepping through the design process, many iterations were made to the enclosure to improve its functionality, rather than the overall aesthetic. Beginning with the documents and Blinky Buildings board dimensions, constraints were established on the design of the enclosure. The purpose of this enclosure is to extend the current shape profile and complement the blinkiness. The dimension tool in the board’s Eagle file can be used to obtain the exact dimensions of the board, which becomes the minimum dimension of the enclosure and forms the cavity. Keeping with the building theme, the enclosure profile was chosen to match the board dimensions, with a wall thickness of 2 mm or roughly 1/16 inch. This wall thickness allows for a rigid part, but permits some minor flexibility that will be utilized later in the design process.

Several iterations were made to enable the removal of the board from the enclosure, while still accessing the switch. The first iteration of the design had two full-length tabs on the top and bottom of the enclosure to hold the circuit board. However, with a full tab, the switch on the board would not allow the board to tilt into place. A modified route was then taken in the second iteration in which snap-fit features were used on the inner side walls of the enclosure. These lofted extrusions on each wall of the enclosure allow the board to slip over it and be retained. However, the snap-fit features were still too tight to get the circuit board in and out. Thus part of the bottom tab was removed so the board could slide into place.

Figure 6.6 contains three images of the iteration process, moving from the initial model to the final model. The files to print this enclosure for the Blinky Buildings kit are provided in Appendix F as well as at www.bit.ly/blinkybuildings. The print files are licensed as CC-BY-SA Jason Brownstein.

Figure 6.6  (a) First version. (b) Second version. (c) Third version.  
(Source: Images CC-BY-SA Jason Brownstein)
To complement the blinkiness of the board, materials selection was considered as part of the design process. Clear plastic filament is available for use on 3D printers but it differs in transparency. A trial print was completed with standard clear PLA available from Lulzbot, and the result was considered acceptable. In the second iteration, T-Glase Nylon clear filament was used. This filament was chosen because of its optical properties, which enable it to carry light in much the same manner as fiber-optic cable. The T-Glase made the PLA look slightly dull in appearance. The appearance of the T-Glase diffused the light much better and provided an almost shiny finish to the enclosure. But with any change in the design process comes consequences, which may cause you to rethink another aspect of the design or manufacture. When using PLA, the settings are relatively standard for the Lulzbot, and the printing operation was executed with a high success rate. By comparison, the T-Glase nylon required reworking the settings, including increasing the extruded temperature, slowing the print head movements down by more than half the PLA speed, increasing additional cooling with use of the onboard fan, and adjusting the extrusion layer height. Each of these settings ensured the print layers fused together well, while minimizing shrinkage and unwanted slanted enclosure walls. Print one off for yourself or make a derivative!

Making a Derivative of This Kit

All of the files to re-create this kit or make a derivative of this kit live at www.bit.ly/linkybuildings. These files are licensed as CC-BY-SA Alicia Gibb. (Remember, if you make a derivative of this project, you cannot include a noncommercial or no-derivatives clause to the source files or hardware.) Whether you change functions in the schematic, change the board form factor, or change the economics of the project, feel free to make derivatives! Turn your board into another building in your city or a landmark that is near and dear to your heart. Make the building roll away on wheels or create a glow-in-the-dark version by 3D printing a new case. Use it as a nightlight, a flashy model train landscape, or just a means to impress your friends!

Giving Correct Attribution: An Example

Earlier in this chapter, the “Giving Correct Attribution” section discussed what giving correct attribution means. This section shows you what correct attribution looks like.

Blinky Buildings is a communal descriptive title of this product that you can use—but be aware that having a communal project name is not always the norm. However, the same is not true of my name, Alicia Gibb; my company, Lunchbox Electronics; or my logo. Only I can use my name, along with other people named Alicia Gibb. But I’m the only person who can use my company name or my logo; even other people named Alicia Gibb cannot use my company name or logo. This is why the board contains the following text: By Alicia Gibb ← remove my name for derivatives and put your name, logo, or trademark in its place. Follow these instructions and place your own name or logo here and move my name (and the names of the other original creators) to an attribution section in the README.txt for your derivative.
When working on my Blinky Buildings projects, I leaned heavily on what has already been created within the open source hardware community. I have no idea how to figure out the code and schematics for charlieplexing, but I wanted to drive 20 LEDs off a small, low-cost chip. When I discovered a project close to my needs, I looked at the licensing to ensure I could use it openly, and I looked at who was behind the project to give correct attribution. I had to do a little Internet digging, but found both of the respective creators on Twitter and contacted them to ask permission to use their work (even though it was already licensed as CC-BY-SA, I wanted to ensure it would be okay to write about in a book) and for their preferred name/handle. When you contact someone to tell them you’re using their open source hardware, you’ll probably make their day. One reason to open source your hardware is to allow it to grow and change in ways you never expected.

Here is an example of how to correctly give another creator attribution on your project. This is the attribution section of the README.txt for my project:

Attribution –

The code for this project was downloaded from http://code.google.com/p/avr-hardware-random-number-generation/ Original code by Geoff Steele, altered by commenting some code out to blink LEDs on and off by Alicia Gibb.

The original Fritzing design of charlieplexing 20 LEDs was downloaded from http://fritzing.org/projects/charlieplex-snowfallshooting-star-20-leds by Davy Uittenbogerd. Alicia Gibb drew this schematic in Eagle and altered it into the Empire State Blinky Building form factor.

The original code and the hardware files are both under a CC-BY-SA creative commons license.

Code: CC-BY-SA: Geoff Steele
Fritzing layout: CC-BY-SA: Davy Uittenbogerd
Blinky Building schematic, board file, and BOM: CC-BY-SA: Alicia Gibb

I didn’t take either source’s names, logos, or trademarks and pass them off as my own. Instead, I gave credit for their work by acknowledging the work they created. The standard procedure for open source hardware is to include attribution both in the README.txt and in the software comment block. (For more information on README files, refer to Chapter 14, Taxonomy of Hardware Documentation.)

**Onboard Byline**

The physical board carries only my name, because putting the entire README file, or two other names, on the board file would take up too much space. Physical objects have a footprint with limited space for attribution, a fact that is well understood by the hardware community. There is usually not enough room on the piece of hardware itself to write the names of everyone who worked on it, including the original creators.

Figure 6.7 shows the open source hardware logo front and center on my board, so everyone sees it will know it follows the Open Source Hardware Definition and can be copied under those terms. My board has the attribution text reflected in Figure 6.7: “By Alicia Gibb ← remove my name for derivatives and put your name, logo, or trademark
in its place.” These are instructions on what to do when creating your derivative. In other words, the attribution on the board you create should read: “By: [Insert Your name here].” But unless you are Alicia Gibb, do not use that name!

Notice that on the board I used the open source hardware logo and the word “By” instead of CC-BY-SA. This usage is meant to alleviate any confusion about copyright claims. The physical hardware is not under copyright, so a CC license would not be applicable. CC licenses can be applied only to the source files and documentation, so I did not reflect the CC-BY-SA terms on my board for clarity of separating the IP that protects written documents and hardware. The term “By” does not close down my board, but patenting it would. Generally, creators do not bother to include “By” on their boards, but for the exercise of creating derivatives I wanted to spell out all the elements associated with attribution.

The open source hardware community does not yet have a standard for applying an attribution icon with the open source hardware logo. Over time, the community will most likely come to some sort of consensus as to how the open source hardware logo and other terms should be displayed, along with other conditions such as attribution.

**Summary**

By now, you should understand how to make a derivative of open source hardware, and be aware of the issues and benefits surrounding derivatives. You can create your own derivatives of the Blinky Buildings kit using the source files highlighted in this chapter and available at www.bit.ly/blinkybuildings. If you create a Blinky Buildings derivative, please email me at amgibb@gmail.com so that I can link to your building as well!

Remember to read the licenses of the original creators when making derivatives and follow the license terms. The most important issue that open source hardware faces with respect to derivatives is giving correct attribution without copying a trademarked logo or name. You can help open source hardware become a stronger brand by taking care to give attribution without infringing on another person’s trademark.