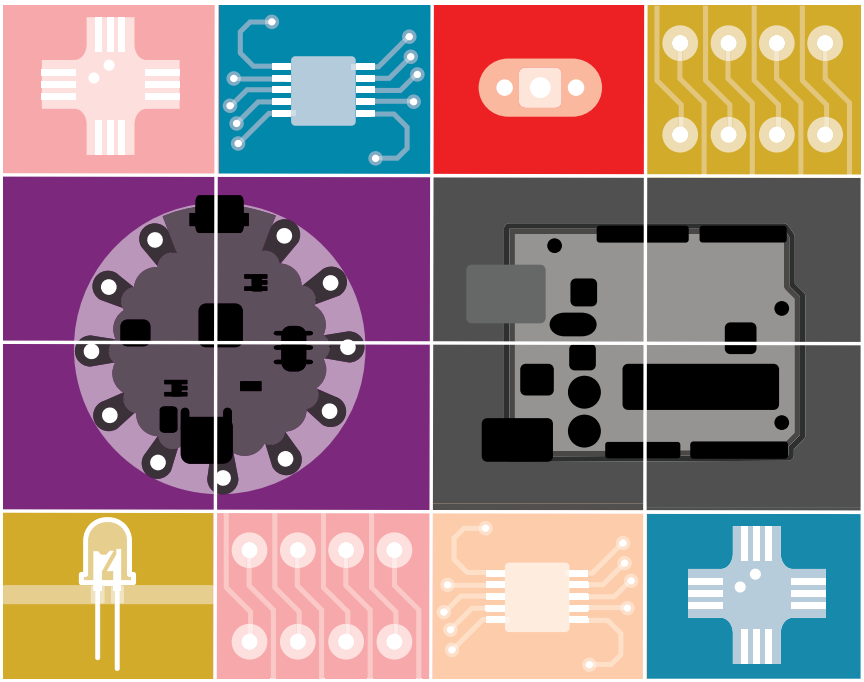
Make:

# Jumpstarting Your Own PCBs



## Learn How to Design and Make Customized Circuit Boards

SHAWN WALLACE

Make:

# JUMPSTARTING Your Own PCBs

## LEARN HOW TO DESIGN AND MAKE CUSTOMIZED CIRCUIT BOARDS

**Shawn Wallace**

**Maker Media, Inc.**
San Francisco

Published by
Maker Media, Inc.
1700 Montgomery Street, Suite 240
San Francisco, CA 94111

Maker Media books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (safaribooksonline.com). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Publisher: Roger Stewart
Editor: Patrick DiJusto
Copy Editor: Elizabeth Welch
Proofreader: Scout Festa
Interior and Cover Designer: Maureen Forys, Happenstance Type-O-Rama

September 2018: First Edition

Revision History for the First Edition

2018-09-24 First Release

See *oreilly.com/catalog/errata.csp?isbn=9781680455076* for release details.



978-1-680-45507-6
PDF: 978-1-680-45510-6
mobi: 978-1-680-45509-0
ePub: 978-1-680-45508-3

**Safari® Books Online**

Safari Books Online is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business. Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training. Safari Books Online offers a range of plans and pricing for enterprise, government, education, and individuals. Members have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and hundreds more. For more information about Safari Books Online, please visit us online.

**How to Contact Us**
Please address comments and questions to the publisher:

Maker Media, Inc.
1700 Montgomery Street, Suite 240
San Francisco, CA 94111

You can send comments and questions to us by email at
*books@makermedia.com*.

Maker Media unites, inspires, informs, and entertains a growing community of resourceful people who undertake amazing projects in their backyards, basements, and garages. Maker Media celebrates your right to tweak, hack, and bend any Technology to your will. The Maker Media audience continues to be a growing culture and community that believes in bettering ourselves, our environment, our educational system—our entire world. This is much more than an audience, it's a worldwide movement that Maker Media is leading. We call it the Maker Movement.

To learn more about Make: visit us at *make.co*.

# CONTENTS

# 1

# Making PCBs with EAGLE

Autodesk EAGLE (Easily Applicable Graphical Layout Editor) is a collection of programs, each serving one part of the design process. We'll focus on the Schematic Editor and the Board Editor; other EAGLE modules include Autorouter (which uses a simple AI to trace circuit routes), the Parts Editor, the CAM Processor (for creating machining-ready files), and a scripting interface for writing user language programs (ULPs). This chapter is a quick overview of the tools we'll be using to create three different electronics projects in later chapters.

Designing a circuit board is not difficult but has some challenges specific to the application area that make having an electronic design automation (EDA) tool particularly useful. EAGLE offers a few advantages over a non-EDA design tool:

* Visualizing connections: Multisided PCBs can have components on different sides and traces that zigzag between layers.
* Hierarchical and parametric design: You should be able to make designs that can inherit properties from modules and be customized using data.

* Libraries: Electronics components come in a dizzying range of packages and variations; tools for managing and organizing devices are critical.
* Design rules: You can specify constraints and test your design against them.

Another important consideration is how the electronics designer fits into a larger product development cycle. Many EDA tools are now offering closer integration with CAD tools—for example, Altium and SolidWorks, or EAGLE and Autodesk Fusion 360.

## GETTING STARTED

Autodesk offers a few different options for licensing your EAGLE software. With EAGLE version 8, Autodesk moved to a subscription model, which gives you a few choices:

* The free version: Autodesk offers a free download aimed at hobbyists and makers, with some limited features (a 12.4 sq. in. size limit, for example).
* EAGLE Standard: This is the version you need to subscribe to if you are doing commercial PCB work. It's $15 a month with some limitations (e.g., an area of 24 sq. in.). Students can get a free standard subscription for three years.
* EAGLE Premium: The premium version is aimed at design teams and provides additional tools for hierarchical design and library management, and it removes all of the limitations of the standard version.

Everything in this book can be done using the free version, which you should download from Autodesk:

*www.autodesk.com/products/eagle/free-download*

We've exported all the library parts used in this book into an easy-to-use custom library. You can grab the JumpstartEAGLE library from *http://github.com/moderndevice/JumpstartEAGLE* and place the LBR file in EAGLE's `lbr` directory (see Chapter 4 for more on creating your own libraries for EAGLE).

## USING EAGLE

EAGLE is a modular tool; you will often have several windows open at once and will be flipping back and forth, so I suggest using a large display. Also, I strongly suggest that you work with a mouse with a scroll wheel if you're using a laptop; the mouse makes it easier to work with EAGLE, compared to the trackpad. The mouse scroll wheel controls the zoom, which you'll find yourself using a lot.

### The Control Panel

The control panel is where you access all the various EAGLE modules and manage libraries, plugins, 3D models, and CAM jobs. There's also documentation bundled with the application. The control panel pops up when you first open EAGLE (Figure 1-1).

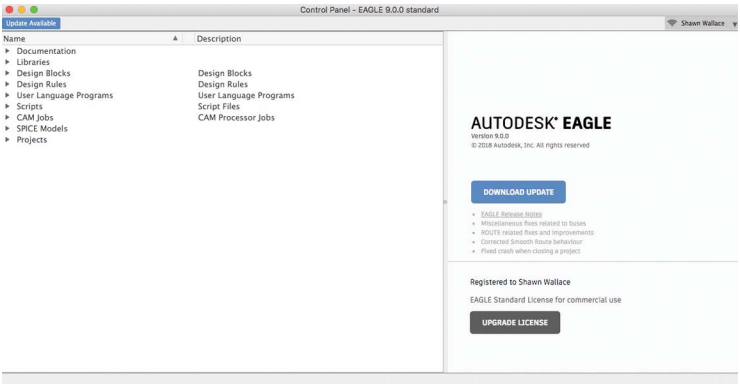



**FIGURE 1-1:** The control panel is a dashboard linking together the various EAGLE modules.

## The Schematic Design Module

The Board Editor (Figure 1-2) provides a set of tools for creating a signal and component diagram that defines all the connections in your circuit. Each module has its own set of tools for its task; the Schematic Editor's tools are aimed at creating networks of signals that connect to the pins of components. Schematics can be organized into sheets to make them more readable and modular.

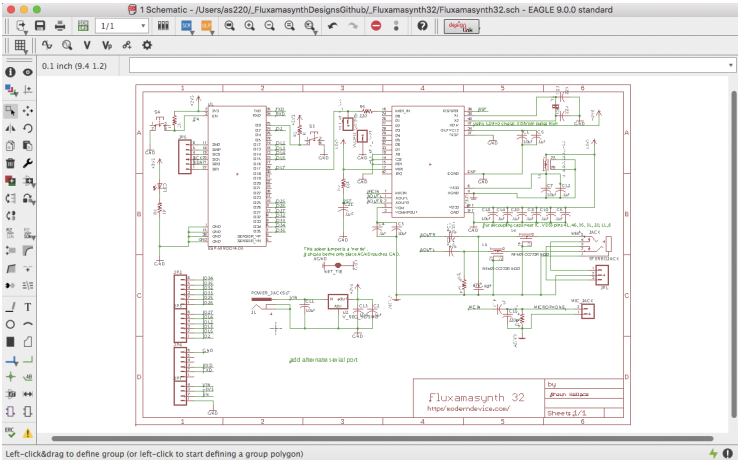



**FIGURE 1-2:** The schematic design tool

## The Board Layout Module

The board design tool (Figure 1-3) is used to lay out pads and traces on the various layers of the circuit board. In this book we will be dealing only with boards that have copper on the top and bottom, but EAGLE is capable of handling up to 16 layers in the Premium version and 4 in the standard version.

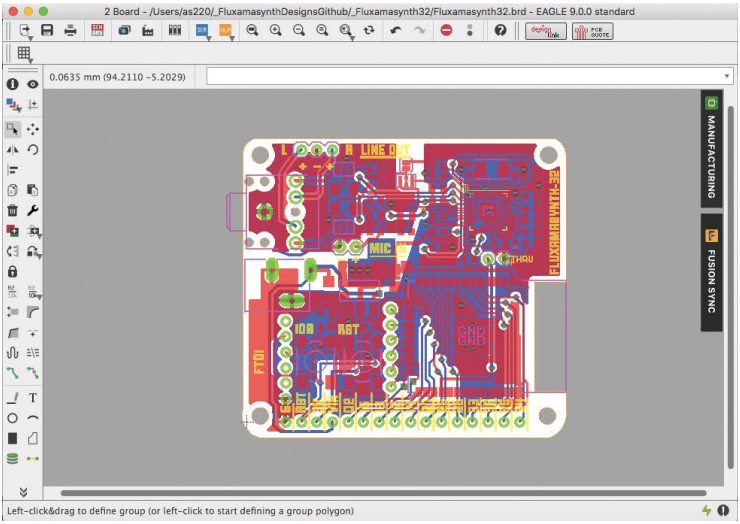



**FIGURE 1-3:** The Board Editor module

Chapter 2 goes into more detail on using the schematic and board tools, and it introduces the layered model of PCB design. The two-sided PCBs we'll be making have eight layers, as shown in Figure 1-4.

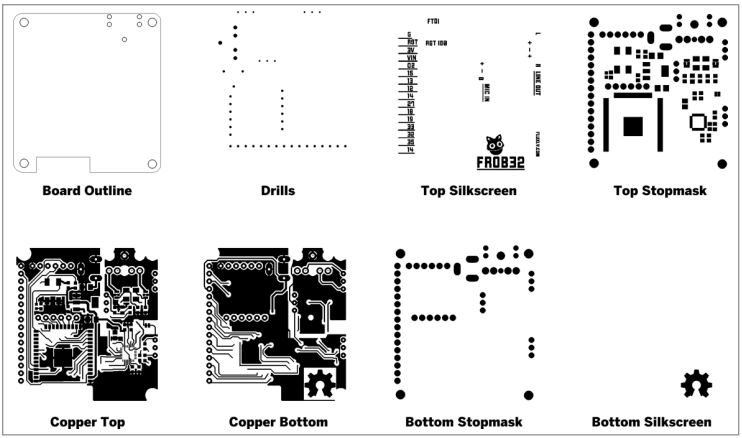



**FIGURE 1-4:** The eight layers of a printed circuit board, from top to bottom: the Board Outline, the drill file, the top silkscreen, the top stopmask, the copper top, the copper bottom, the bottom stopmask, and the bottom silkscreen

## THE WORKFLOW

When designing a circuit board, you'll follow these steps in order:

1. Design and sourcing: Find parts, read the datasheets.
2. Component device design: Find existing device libraries or draw symbols and packages if necessary.
3. Schematic Layout: Connect parts electrically with signals.
4. Electrical Rule Check (ERC): Run an algorithm to identify schematic errors.
5. Board Layout: Place parts on the board and draw the actual traces connecting them.
6. Design Rule Check (DRC): Run an algorithm to identify layout errors.
7. Generate Gerber and drill files: Run the CAM Processor to generate machining files.

## LIBRARIES

An important consideration when choosing a design tool is how it handles reusable libraries of components. In EAGLE, a component or device is a member of a collection of devices called the *library*. A device is composed of four parts: a schematic symbol, a board package, a 3D model, and a device definition file that makes the connections between the two views. EAGLE comes with a large selection of device libraries, and most manufacturers provide EAGLE libraries for their parts. There are a number of repositories for EAGLE libraries:

* Autodesk's library collection: *http://eagle.autodesk.com/eagle/libraries*

* Library.io: A tool for parametrically generating libraries of devices and 3D models that can be synced with Fusion 360
* Element 14's community-contributed libraries: *www.element14.com/community/community/cadsoft_eagle/eagle_cad_libraries*

Autodesk has also introduced a new feature called *managed libraries*. These are libraries of parts that live in the cloud at `Library.io` and are available locally. Managed libraries are created with manufacturers and partners like Adafruit, SparkFun, Seeed, Würth Elektronik, and Nordic. Because they are centrally managed, they'll always be up to date.

Chapter 4 goes deeper into EAGLE's libraries and shows how to design your own library of devices.

**NOTE** **Similar to libraries, design blocks are another new feature that allow you to create larger reusable modules of schematics and board layouts. Design blocks are great for when you have a number of repeating component groups.**

## USER LANGUAGE PROGRAMS

EAGLE has a plugin system, where you can write automation scripts called ULPs. An example of a ULP that became integrated into the main user interface is the Import Bitmap function; there are also ULPs for doing convenient transforms like snapping all parts to a grid or changing properties in bulk. A number of ULPs come with the default distribution, available via the control panel or the File menu.

ULPs are written in a custom scripting language that is C-like, with all of the data structures of the schematic and board

documents exposed via a relatively simple API. A trivial ULP looks something like this:

```
int returnValue = dlgDialog("HW") {
    dlgLabel("Hello world");
    dlgPushButton("OK") {
      dlgAccept();
    }
};
```

## GETTING PCBS MANUFACTURED

There are a few processes for making circuit boards from the designs that we'll be creating in this book. The process you choose will depend on the application, safety, and the availability of tools. Here are a few processes:

* Etching: This is how all production run boards are made; a stencil is applied to a copper clad board and corrosive chemicals etch away the negative space to create copper traces and pads. Common corrosives are ferric/cupric chloride or ammonium/sodium persulfate. Each has its own safety concerns and hazards.

* Machining: This is an increasingly accessible way of making circuit boards and involves using a precision milling machine to cut away the negative space around your traces and pads. Popularized in the fab labs, now there are many desktop milling machines that are up to the task. Great for prototyping a couple of boards but too time-consuming for production work.

* Sewing, printing, cutting: As soft electronics materials become more commonplace, there are corresponding processes for making circuits with conductive inks, thread,

or foils. Most of these crafty processes introduce design challenges of increased capacitance and resistance but are totally viable.

Most people will probably not be manufacturing their own boards in their own workshop and there are plenty of good reasons why, not the least of which is that these days it is incredibly affordable to get a small run of boards produced by a number of vendors who specialize in short runs and prototype editions. Two of the most popular are OSHPark (purple boards; *http://oshpark.com*) and Screaming Circuits (*www.screamingcircuits.com/*). These services are great for getting three boards for a few dollars per square inch. If you need 100 or more, expect to pay around $1.50 to $2 per board to start (some of that is startup tooling cost), depending on the size of your PCB.

# 2

# Your First Board: A "Learn to Solder" Badge

For our first PCB project, we'll start with a staple of many Maker Faires: a "Learn to Solder" badge. This is the simplest of circuits: two cycling RGB LEDs and a battery clip (see Table 2-1). Your goal of this project is to get comfortable navigating EAGLE's toolbox and layers. You'll also see how to create custom silk-screen graphics and nonrectangular outlines.

**TABLE 2-1:** Blinky badge bill of materials

| PART | SOURCE | PRICE ($) |
|---|---|---|
| (2) Cycling RGB LEDs | Microtivity IL603 | .20 |
| Through-hole CR1220 battery clip | Digi-Key 3000k-nd | .42 |

To make this kit in a workshop setting, you'll also need some tie tack pins.

Figure 2-1 shows what the finished product will look like.

**FIGURE 2-1:** A successfully soldered blinky badge board. These RGB LEDs have a forward voltage of about 3V and slowly cycle through different colors.

## START WITH THE SCHEMATIC

The Schematic Editor is where we connect all the components in the circuit with networks of signals. Chapter 3 has a more robust example of creating a schematic; this first project will have a very simple schematic. You'll be using many of the tools in the schematic toolbar shown in Figure 2-2.

EAGLE separates CAD drawings into color-coded (and numbered) functional layers. In the Schematic Editor, part outlines are on the Symbols (94) layer, labels on the Names/Values (95/96) layers, and nets or signals on the Nets (91) layer. You can show or hide the layers using the View → Layer Settings menu. The current layer will show in the toolbar, so check that often to make sure you're drawing into the right layer.

To start, open the Schematic Editor by selecting File → New → Schematic. The first tool you'll use is the Add tool, which allows you to add a symbol to the schematic area from one of EAGLE's device libraries. Chapter 4 shows how to create your

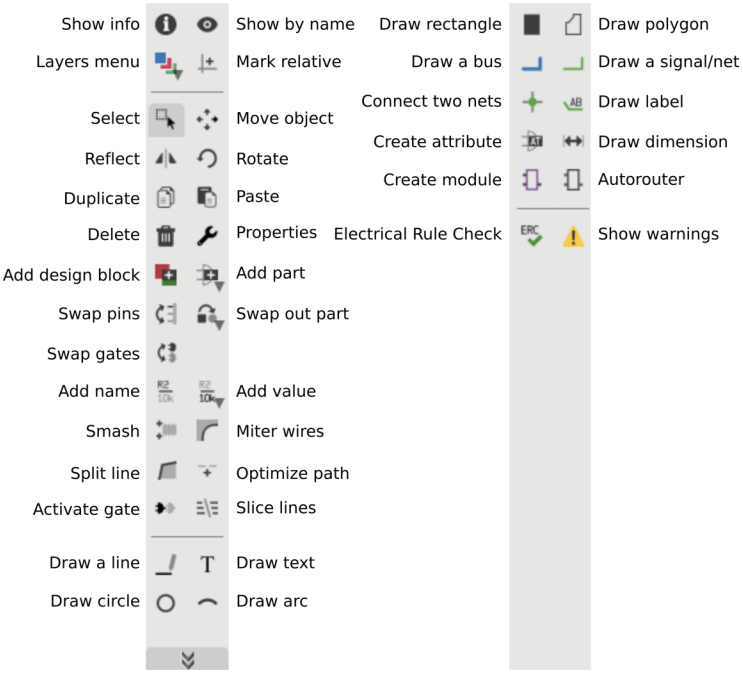



**FIGURE 2-2:** The Schematic Editor and the Board Editor share many of the same tools, with a few differences.

own device libraries, but for this first project you will work with the Jumpstart library that you installed in Chapter 1.

## Keeping Schematics and Board Layouts in Sync

**When you create a new schematic, you'll get a warning that "no forward/ backward annotation will be performed." Normally you'll have both the Schematic Editor and the Board Editor open at once so EAGLE can keep them synced; this message just says that there's no board file open yet. Once you've created a board from the schematic, the two files must be open at all times or they will get out of sync (which is difficult to recover from).**

Use the Add tool to select the“CR1220TH” and“LED5MM” parts, which you can place anywhere on the schematic, as shown in Figure 2-3. One thing you'll notice is that the naming of devices and packages is nonstandardized, so you may have to pick your way through a bunch of jargon when trying to find parts by different manufacturers. With EAGLE 8.7, Autodesk has put a lot of effort into library management (see “Libraries” in Chapter 1), but you may still want to spend some time looking through the libraries to get your bearings.

In EAGLE, parts are connected using the Net tool, the green drawing tool near the bottom of the toolbox. Each signal net that you draw has a unique name, and nets have the special property that anything with the same name will be considered connected to the net whether or not they are visually connected by a line. This allows for much cleaner schematics, but also means that you need to be disciplined and organized about naming your nets

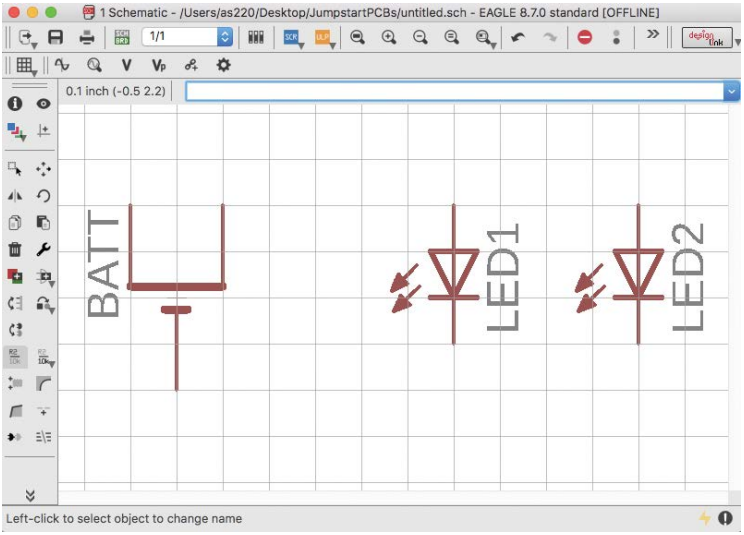



**FIGURE 2-3:** Place the three components somewhere on the schematic layer. Use the Zoom To Fit tool to change your workspace to match your part layout.

(see Chapter 3). Start connecting the battery and the LEDs using the Net tool as shown in Figure 2-3.

In Chapter 3 you'll make a much more involved schematic and be more explicit about labeling voltages and ground signals, as well as use the built-in Electrical Rule Check to proof your schematic. For this first project, you're all done with the schematic if it looks like the one in Figure 2-3; time to move on to laying out the board!

## LAYING OUT THE BOARD

Switch to the Board Layout tool using File ➞ Switch To Board. Your parts will initially appear to the side of your work area. Select the Move tool and grab each part by its origin crosshair to move it onto the white work area. The default white work area is about 6″×4″ in the educational version; resize it using the Move tool to shrink the boundary lines to about 1.5″ square (Figure 2-4), and then move the parts to the workspace (Figure 2-5).

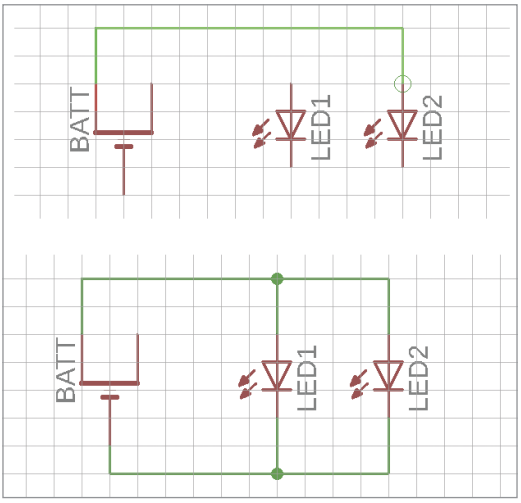



**FIGURE 2-4:** Connect the positive side of the battery clip to the anode (positive) side of the LEDs (top), and then the negative sides to the cathodes (bottom).

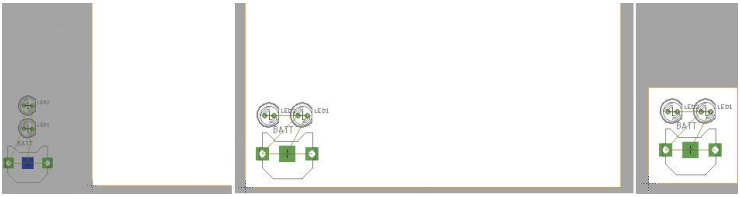


**FIGURE 2-5:** Move the parts to the workspace. Note that in the educational version of EAGLE you cannot place objects outside the work area once you have picked them up.

## Things Won't Move Without TOrigins

**You will need to have the TOrigins layer active for the part selection crosshair to appear. If that layer is not active, you won't be able to move parts around since the crosshair will not be visible. You can also lock parts in place so they can't be moved by accident using the Lock tool.**

You'll notice the yellow lines connecting the various pads of the circuit; these are *airwires* that show the connections that were made in the schematic tool. These airwires are a map for routing the connections between the various parts.

One of the first things you should do when routing a board is check the grid. Anything you draw or place will snap to the grid; the grid can be your best friend, or it can be frustrating if parts get "off grid." The default grid is 100 mils (100 thousandths of an inch = .1″), which is standard spacing on wireless breadboards or headers but is too coarse for most routing. Select View ➞ Grid and set the grid to 50 mils (.05″) to start.

You can use the Move tool to manually position parts, or you can use the Info tool to set a part's position by typing in a value. For this project our parts need to match up with a silkscreen

layer perfectly, so we will use the Info tool to type in exact x- and y-coordinates. For each part, type in the coordinates shown in Table 2-2 and click OK.

**TABLE 2-2:** Coordinates for the parts in the project

| PART | POSITION X (IN.) | POSITION Y (IN.) |
|---|---|---|
| LED1 | .32 | .78 |
| LED2 | .92 | .80 |
| BATT | .68 | .29 |

There are many ways of routing traces in a circuit, and EAGLE also has an Autorouter tool that applies some machine learning to more complicated routing problems. For the circuits in this book, we will manually route the traces.

Select the Route tool from the bottom of the first section of the toolbox. A toolbar will appear that allows you to control various routing behaviors, as shown in Figure 2-6.

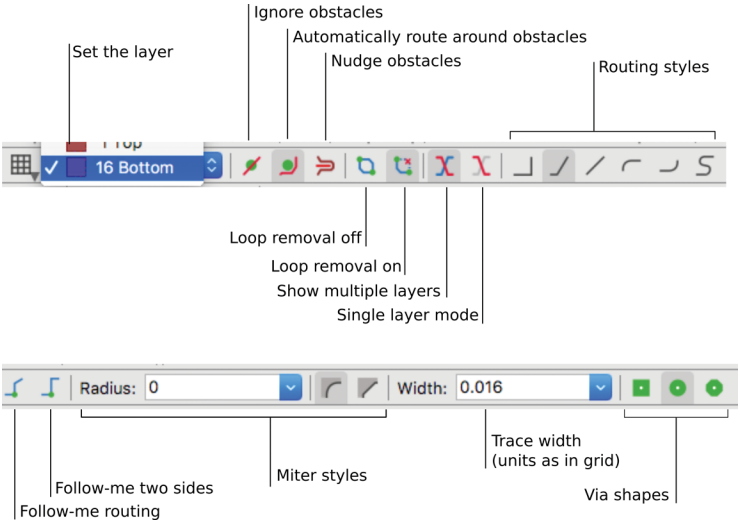



**FIGURE 2-6:** The Routing toolbar selects the layer and trace width, and also provides various constraints and helper functions for making routes that follow your design rules.

Start by connecting the LED cathodes and the negative pad of the battery on the bottom layer (blue traces), with a 16 mil (.016") trace (Figure 2-7). Note that the pad/negative side of the battery clip is on the bottom (blue) layer in the library. The green pads are plated-through so they appear on both the top and bottom layers.

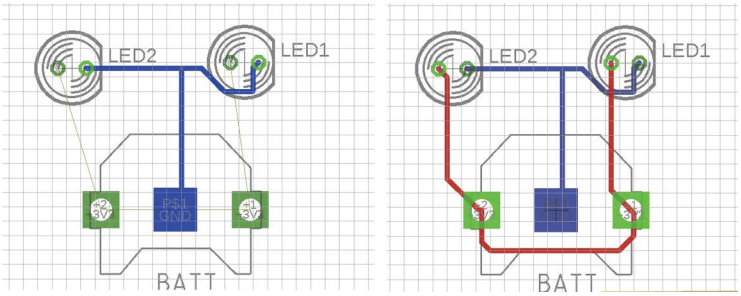



**FIGURE 2-7:** Routing the bottom copper traces (left) and top copper (right)

Sixteen mils is a good starting point for general-purpose traces; you'll usually be able to fit in between pads without changing layers, though sometimes you may need to go down to 10 mils or so. Most general-purpose PCB houses guarantee a lower limit of 6 mil traces for bulk PCB etching.

## ADDING A CUSTOM SILKSCREEN AND OUTLINE

If you don't change any of the default settings, the Dimension, tPlace, tNames, and tValues layers will be included on the silkscreen layer. To see what the silkscreen layer looks like, go to the Layers menu and deselect all layers except those four. You'll also be able to preview the silkscreen layer when you go to make your final manufacturing files.

When adding text to the board, you should keep some general guidelines in mind. PCB manufacturers use different resolution

meshes in their silkscreens and generally use a screen that's just fine enough to get the job done. If you have very fine lines they may not reproduce well; a general rule of thumb for text is that the smallest size should be 32 mil or so, and the thinnest line of a letterform should be around 5 mil. To achieve this, zoom in on your finest letterforms and use the Ratio property to adjust the text to a proper weight. A 15 percent ratio at 32 mil is a good minimum guideline.

In this first project, we'll use the Text tool only to add some polarity indicators on the LEDs; everything else will be a custom image. You can use this technique to put a logo on your board, or you can design your entire silkscreen layer in a graphics program to use whatever typefaces and graphics you want.

To make a custom silkscreen bitmap, get a PNG version of your board layout to draw on top of. The following steps show one way to accomplish this, which assumes you're using the Inkscape vector drawing tool. The important part is to get a PNG image of your board that is the same dimensions as the EAGLE representation:

1. Select only the Pads, Vias, Dimension, and tPlace layers from the Layers menu.
2. Choose File → Export → Image.
3. Fill in a resolution of 920 pixels per inch. Inkscape imports images at 92dpi by default. We're going to be scaling it down to 10 percent of the original size to get a nice, sharp image.
4. Open your vector drawing program (Inkscape in this case) and import the board PNG.
5. In Inkscape, select Object → Transform → Scale and scale the imported PNG to 10 percent of its size.

Now that you have a nice, clean image of the board as a background, draw the silkscreen image using the vector tools. For this project you can use the files that are included in the GitHub repository if you want to skip the drawing step. In Inkscape, select just the black vector objects (not the original board PNG) and select Export. Export the PNG at a high resolution (400dpi is good for most PCB manufacturers).

Next, convert that PNG to a black-and-white BMP image. If you exported it from Inkscape, you'll need to flatten the image as well to remove the transparent background. In a raster image editing program like the GIMP, open the file and then perform these steps:

1. Select Image ➞ Flatten Image.
2. Select Image ➞ Mode ➞ Indexed ➞ Use Black & White (1-Bit) Palette.
3. Select File ➞ Export ➞ Windows BMP Image (.bmp) (Figure 2-8).

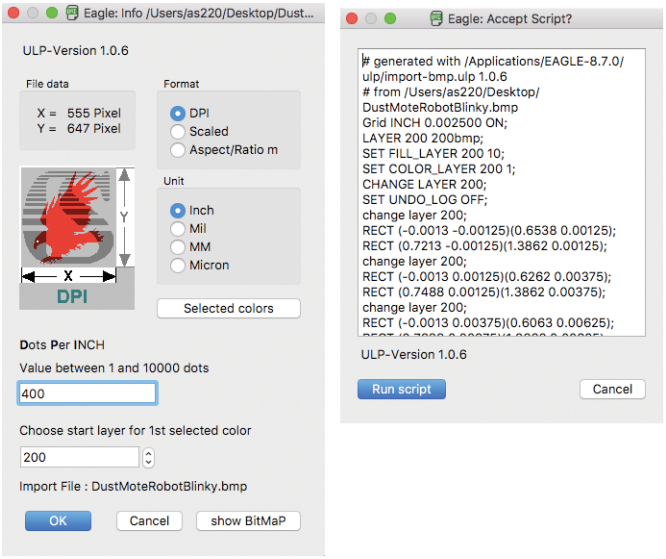



**FIGURE 2-8:** The dialog box for the Import Bitmap command

Next, return to EAGLE and import the bitmap onto a layer of your board. Select File ➞ Import ➞ Bitmap. This runs an EAGLE plugin (or user-defined program [UDP]) that is also available under the UDP menu.

Specify the color you wish to import (black), select DPI as the format, and enter **400** under Dots Per Inch. Use the default layer 200, though you can import bitmaps to any layer you want, even the top and bottom copper or stopmask layers.

The script will import the bitmap onto the layer you specified, and it will translate the image into hundreds of horizontal lines. You won't be able to edit it directly, and all the lines must be moved as a group. It's a little clunky, but here's how to move the whole silkscreen image:

1. In the Layers menu, select only the 200 bmp layer.
2. Use the Selection tool to grab the whole image.
3. Select the Move tool.
4. Right-click on one of the lines in the image and select Move Group.
5. Turn the other layers back on now that the image is selected.

**NOTE** **The ULP script used by the Import Bitmap function will change your grid along the way, so be sure to set it back to something sane.**

If you placed the LEDs and the battery clip in the exact positions mentioned earlier, the silkscreen should import exactly aligned with the parts. When you go to make the final Gerber files in the next section, be sure you assign layer 200 to the silkscreen section of the CAM job.

Finally, to make a custom outline for the PCB, you can draw into the Dimension layer. The outline must be a watertight rectangle or polygon; EAGLE 8.7 introduced a special layer that is generated during the CAM process called the Board Outline. The outline is calculated based on the shapes on the Dimension layer. Any outline will define the shape of the board, and any closed polygons inside the bounds of the shape will be interpreted as cutouts.

## MAKING GERBER FILES FOR MANUFACTURING

The files you send to the PCB manufacturer are in the Gerber file format, a standard CAM file format for PCBs. Gerber files are created using the CAM Processor, which is available by selecting the Manufacturing tab to the right side of the Board Layout tool window. When you first open the Manufacturing tool, you'll see a rough preview of what your board will look like if you run the default template for the CAM tools, as in Figure 2-9.

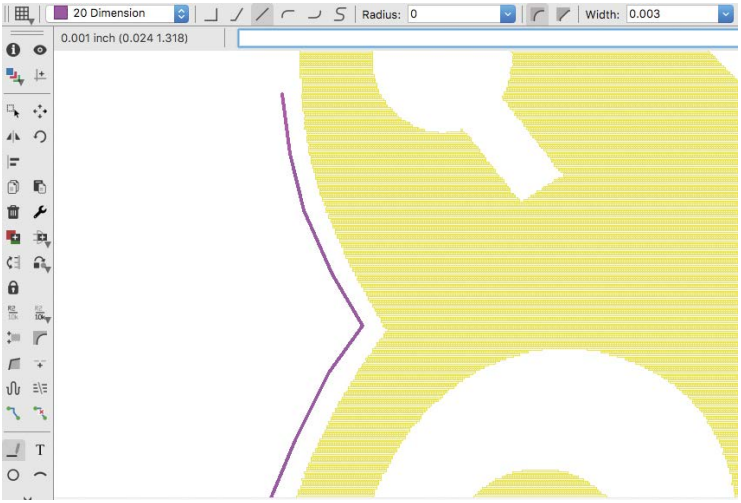



**FIGURE 2-9:** Drawing an outline into the Dimension layer with the Polygon tool. Note that the outline needs to be watertight, so you can't import a bitmap to define the outline.

**NOTE** Many services allow silkscreen on the top and the bottom of the board. To add a silkscreen design to the bottom, you'll have to add a new Gerber file to the CAM job with a .pls extension.

To create the CAM files, open the CAM Processor from the File menu. Note that this tool is context-sensitive whether you're in the schematic or board view, so make sure you're in the board view. Open the excellon.cam job to create the drill files (with extensions .dri and .drd). You only need the DRD file; the DRI is just metadata.

Next, generate the Gerber files by opening the template_2layer.cam job as in Figure 2-10. The only section you should need to double-check at this point is the Silkscreen tab (Figure 2-11); if you've added a bitmap layer (or don't want a layer on the silkscreen), be sure to assign the correct layers on that tab.

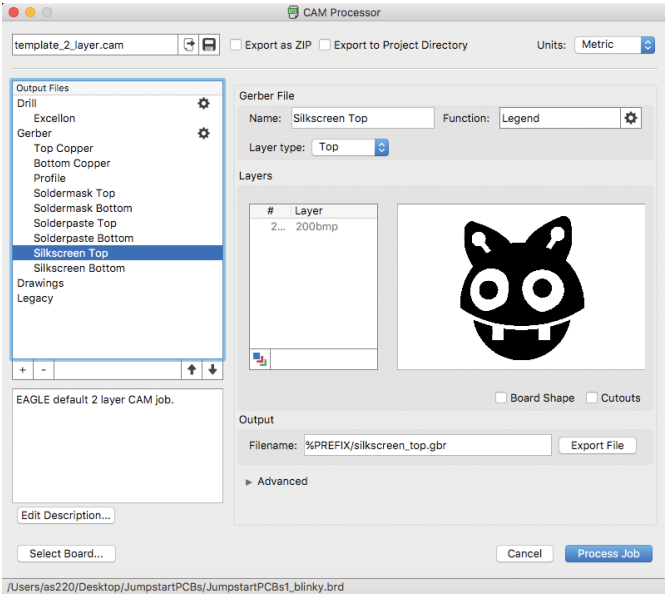



**FIGURE 2-10:** The CAM Processor

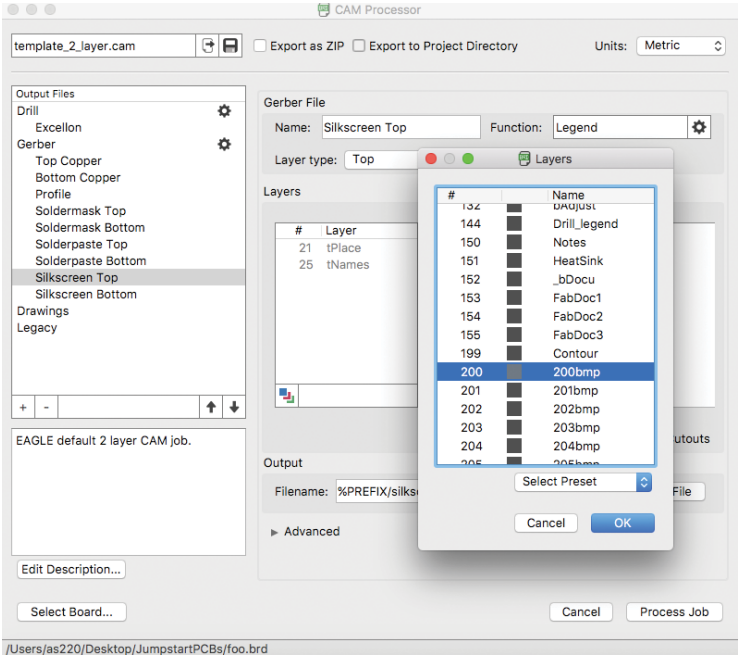



**FIGURE 2-11:** Removing tPlace and tNames from the Silkscreen Top and adding 200bmp, the custom bitmap layer

To preview your files you'll need a Gerber viewer tool. Gerbv (Figure 2-12) is a good open source option if you're on a Unix-based system, or you can use ViewMate on Windows. If you're on a Mac, Gerbv is a little tricky to get up and running, so you should try to use a package manager like MacPorts to install it. The easiest solution to previewing Gerber files is to use the free online tool from CircuitPeople.

Import the layers into your Gerber viewer (Gerbv is shown here). I usually reorganize them from top to bottom to make it easier to proof:

| | | |
|---|---|---|
| .drd | .stc | .sts |
| .plc | .cmp | .sol |

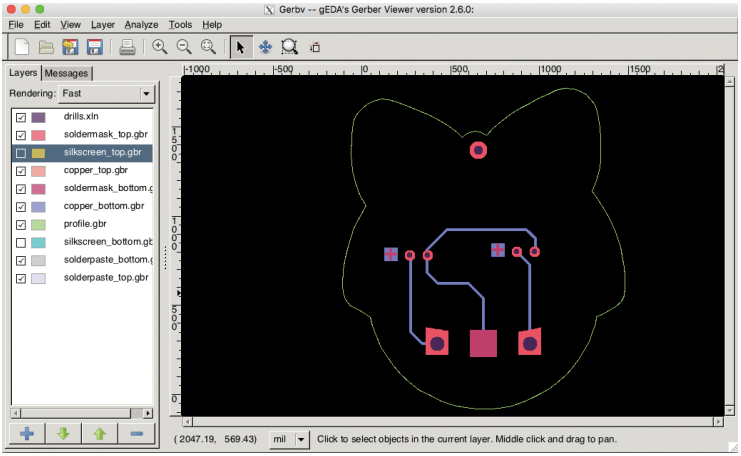



**FIGURE 2-12:** The Gerbv tool, part of the open source gEDA suite of tools

Turn each layer on and off and inspect; if you tweak the layout, rerun the CAM Processor.

> **NOTE** If you move a part, remember to run the Excellon drill job as well. Otherwise, your holes won't match up with your pads!

Bundle them up and send them out. A great service for small runs is OSH Park, which grew out of the DorkbotPDX service and merged with SparkFun's BatchPCB service. The interface is well designed and service is quick and consistent, with a free shipping option. Your board will be bundled with dozens of others and will come back with a distinctive purple soldermask. If you need more than a dozen boards, it is more economical to go with another service. There are many competitive options (and more popping up in the United States); I've had good luck with the affordable PCBCart service.

## GOING FURTHER

A great way to extend this project would be to add a switch. The battery lasts about four days, but it would be nice if you could turn it off. If the switch is a through-hole part, you'll have to think about how it affects the front-side design. If it is surface mount, you'll have to think through how you'll teach kids to solder it easily.

# 3

# Make Your Own Minimal Arduino

The Really Bare Bones Board (RBBB) and its schematic were originally designed by Paul Badger of Modern Device. It's pretty much the bare minimum needed for a useful Arduino-type development board. The goal of this chapter is to delve into the details of building a more complicated schematic and board layout, and to learn the workflow of the Electrical Rule Check and Design Rule Check. We'll use through-hole parts, but you can easily substitute surface mount parts. A surface mount version of the project is provided with the support files to this book.

Although this is a minimalist project, there are still a dozen or so parts needed to make a microcontroller board (see Table 3-1).

**TABLE 3-1:** Minimal Arduino BOM

| PART | SOURCE | PRICE ($) |
|---|---|---|
| (1) Resistor 10K 1/8W | Digi-Key 10KEBK-ND | .03 |
| (2) Electrolytic capacitor, 47µf | Digi-Key P834-ND | .10 |
| (1) Resistor 1K 1/8W | Digi-Key 1.0KEBK-ND | .03 |

| PART | SOURCE | PRICE ($) |
|---|---|---|
| (1) Switch, pushbutton | Digi-Key CKN9098-ND | .17 |
| (1) Ceramic resonator, 16MHz | Digi-Key 535-9355-ND | .20 |
| (1) ATmega328P | Digi-Key ATMEGA328-PU-ND | 1.96 |
| (1) IC socket, 28-pin (optional) | 4UCON 00820 | .10 |
| (1) Diode, 4005 | Digi-Key 1N4007GOS-N | .15 |
| (1) Voltage regulator, L4931 | Digi-Key 497-5838-1-N | .50 |
| (1) Power jack | 4UCON 18742 | .55 |
| (1) Pin header, 1×6 | 4UCON 00872 | .03 |
| (2) Ceramic capacitor, 0.1µf | Digi-Key 478-3187-ND | .10 |

You'll find that once you build this particular schematic in EAGLE it's easy to incorporate in other designs, this is the core of many Arduino-based designs, like the JeeNode in Figure 3-1.

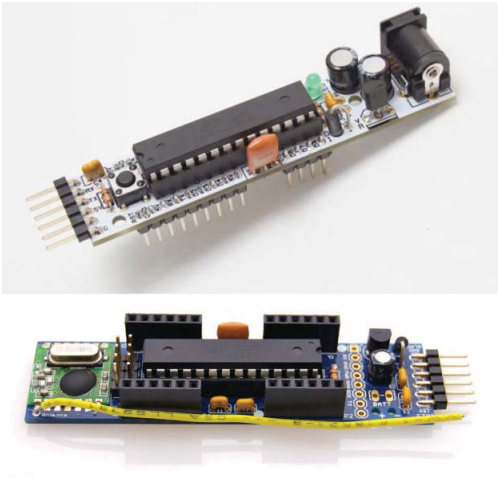


**FIGURE 3-1:** The finished product (top). The JeeNode (bottom) is also based on this foundation design, which has rearranged headers and a radio module.

## SCHEMATIC DESIGN

To start designing the RBBB, choose the Add tool in EAGLE, grab the Frame part from the RBBB library, and place it on the canvas. It's not essential, but it's handy so you don't have to keep resizing your window as the schematic grows.

### The Power Circuit

The five parts of the power circuit allow you to hook up a DC input power supply or battery in the 6V to 12V range and provide a clean 5V supply to the rest of the circuit.

The main component of the power circuit is the voltage regulator (see Figure 3-2). The key specs for a regulator are output voltage, maximum output current, and dropout voltage (which is the minimum difference needed between input and output). The L4931 has a dropout of just 0.4V, so it can reliably output 5V given 6V (e.g., from four AA batteries). It provides current up to 250mA, which is plenty for most microcontroller applications.

The two electrolytic capacitors filter noise on the input supply, pick up slack when batteries fade, and handle momentary power demand spikes. The regulator's datasheet recommends 2.2µF filter caps minimum; we'll use 47µF. That should be stiff enough for even the noisiest battery-powered breadboard experiments.

The power jack may seem big, but this is one of those human interface decisions that really matter—this jack easily accommodates a "wall wart" power supply. We'll position it on the board where it can be snipped off if it's not needed, and we'll provide a two-pin auxiliary power header for optional wiring.

Almost any power diode will serve for short-circuit protection; this one is a common 4005 rated 1A, wired in parallel to prevent damage if power is accidentally connected backward.

Some designs opt for a series protection diode, but that would introduce a 0.7V drop we can't afford if we want to run from 6V.

Some rules of thumb:

* A dot indicates that intersecting signal lines are connected; lines that cross without a dot are just overlapping.
* Stay on the grid. The most common mistakes come from parts not lining up, looking like they're connected when they aren't. Set it to 0.1 (100 mil) to start.
* Each line (or net, or signal) has a unique name, and all signals with the same name are automatically connected, even if there's not a line between them. This is handy if you have so many parts that drawing every line would make a pile of spaghetti.
* Each part should have a visible name and a value. You'll need to move the labels around to make them readable; use the Smash tool to separate labels from parts, and then use Move and Rotate to position them.
* Drawing a schematic is about communicating a circuit assembly to others. Think about whether you're providing everything they will need.

**NOTE** **Some of these diagrams may look different from your screen at first. You can use the Smash tool to separate the labels, and then use Move so they're readable.**

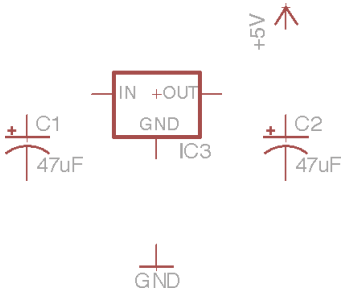



**FIGURE 3-2:** The schematic symbol for a voltage regulator and two capacitors. The +5V and GND symbols are special parts in the "supply" library.

Select the Add tool, and choose the Regulator part from the Jumpstart PCBs GitHub library you downloaded in Chapter 1. Place the regulator somewhere in the upper-left quadrant. Use the Add tool to place the two electrolytic caps, plus the GND and +5V supply signals (under Supply), as shown in Figure 3-3.

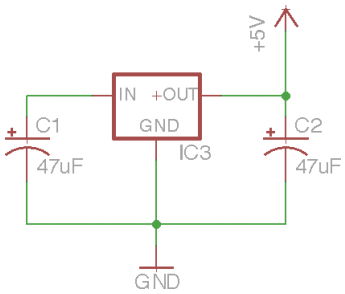



**FIGURE 3-3:** Connecting the parts with nets and labeling the components

Use the Net tool to connect the caps' positive sides to the regulator's input and output and to connect their negative sides to the regulator's ground lead. Connect the regulator's ground lead to GND and its output to +5V.

Use the Value tool to assign each capacitor a value of 47µF.

Add the Power_Jack part beside the regulator input (Figure 3-4). This jack is center-positive, which (outside of musical electronics) is pretty standard. Connect its center pin to the regulator input and its sleeve to ground. If you get a Connect Net Segments? dialog box, click Yes.

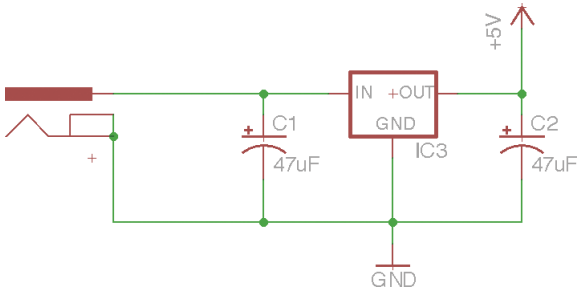



**FIGURE 3-4:** Adding a power jack

Add the diode (Figure 3-5). It will appear horizontally. Use the Rotate tool to turn it cathode/negative side up. Then use the Net tool to connect it across the regulator input and ground.

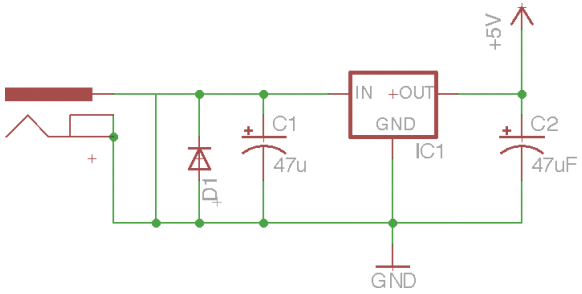



**FIGURE 3-5:** Adding a diode, which will protect the regulator if power is applied in reverse polarity

Finally, drop in the 1× header (1×_Pinhead) for the optional power input. Position it with Rotate, and then connect one pin to power and one to ground (Figure 3-6).

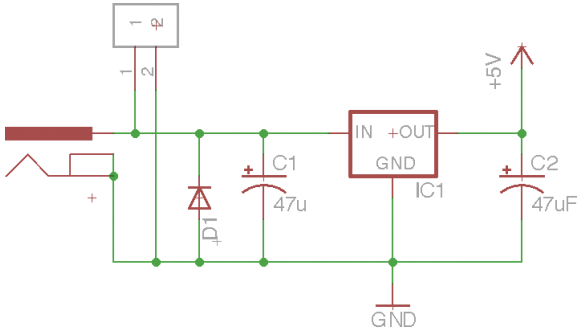



**FIGURE 3-6:** Pulling out the power input to a couple of header pins

## The Microcontroller and Headers

The headers connect to the microcontroller's general-purpose inputs and outputs, and they provide points where you can solder wires or headers for your particular project.

Add the microcontroller chip (ATmega) somewhere in the middle of the schematic, and connect its two ground pins to the GND supply signal as in Figure 3-7.

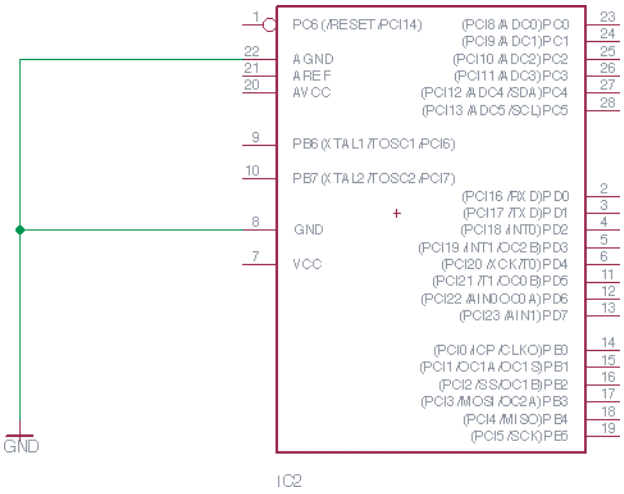



**FIGURE 3-7:** The ATmega328P is the microcontroller at the heart of your typical Arduino.

Place a 0.1µF capacitor near the ATmega's power supply pin (7), and connect it to power, ground, and pin 7 (see Figure 3-8).

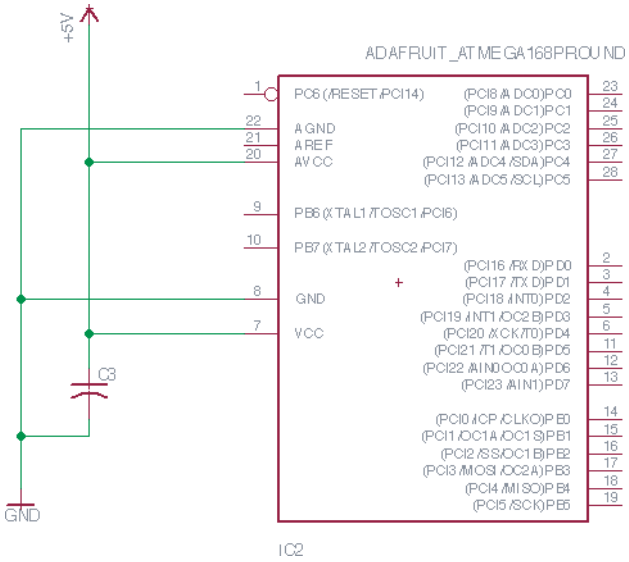



**FIGURE 3-8:** At a minimum you should have a .1µF filter capacitor close to the VCC power pin.

Place the resonator near clock pins 9 and 10, and connect its three pins as shown. Make sure the center pin goes to ground (see Figure 3-9). Create a signal line for all the other pins. Don't forget analog reference (AREF) on the left. Use Net to connect a short signal to each pin, then use Label (right underneath) to label each one.

**NOTE** **EAGLE's "group move" function does not work like most modern drawing software. If you need to move several parts at once, first use Zoom to zoom out, and then choose Select and click-drag around the parts to be moved. Choose Move, left-click on the selected objects, Ctrl-right-click to get a menu, and choose Move Group.**

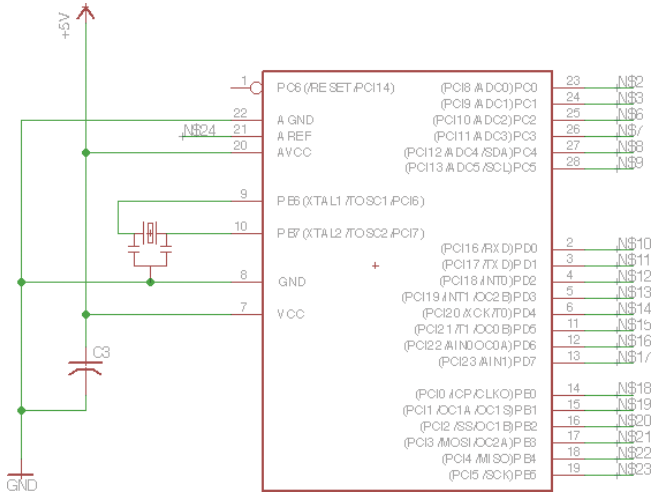



**FIGURE 3-9:** The 16MHz resonator attaches between the XTAL pins on the microcontroller. This resonator is not quite as accurate as a crystal, but it has internal capacitors so it simplifies the design and lowers the part count. You can see the internal capacitors represented in the schematic symbol.

You'll see EAGLE gives a default label to each signal, something like N$2. Now use the Name tool to rename each signal to match its ATmega pin name (as in Figure 3-10).

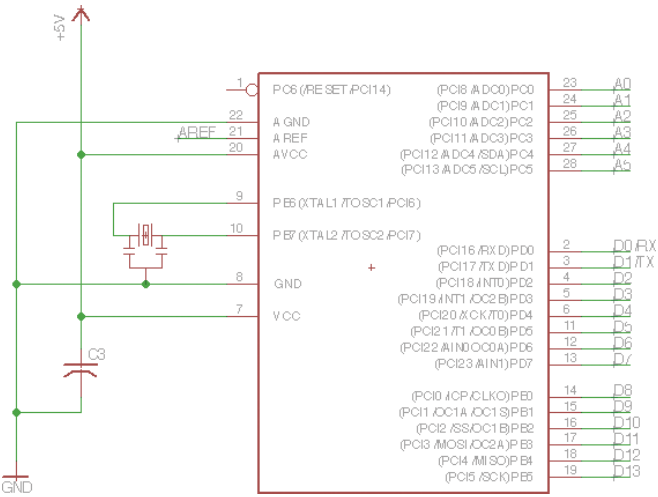
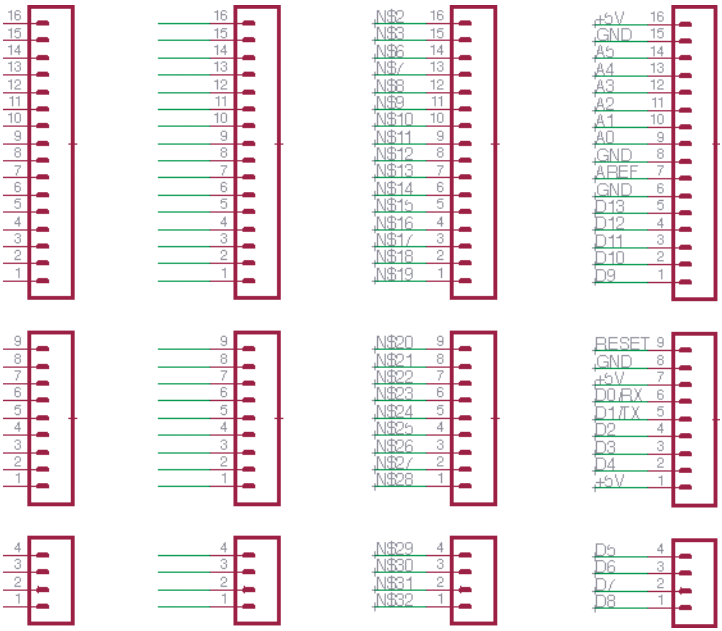



**FIGURE 3-10:** Using the Name tool to rename all the nets attached to the microcontroller pins

Add the 1×16, 1×9, and 1×4 header blocks and repeat the process of adding signals, adding labels, and changing names (as in Figure 3-11).




**OFIGURE 3-11:** The three steps of adding the header blocks: placing the parts, adding a net to each pin (EAGLE automatically gives you a label), and renaming them to match the numbering scheme in Figure 3-10. The order here was chosen to make routing the connections a bit easier.

## The FTDI Serial Communications Header

The data interface is a 6-pin header connected to the ATmega's serial port (see Figure 3-12). The original Arduino had a USB port and a Future Technology Devices International (FTDI) chip to translate from USB to TTL Asynchronous Receiving and Transmitting Protocol (UART). Moving this chip off the board and into

the cable is an easy way to pare down the design. FTDI sells a cable with a 6-pin connector, or you can use a USB-BUB or FTDI Friend.

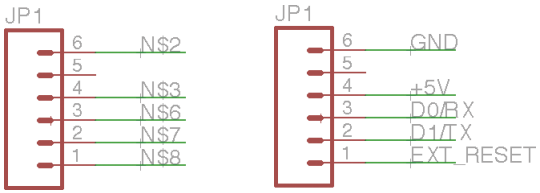



**FIGURE 3-12:** Place the 6-pin header and connect the RX (receive), TX (transmit), GND, +5V, and EXT_RESET pins.

## Some User Amenities

Many designs skimp on human factors, with tiny buttons, inconvenient part arrangements, and indecipherable labels. Let's not do this. Our design includes a generously sized reset button and an LED "pilot light" to clearly show when the board is powered (Figures 3-13 and 3-14).

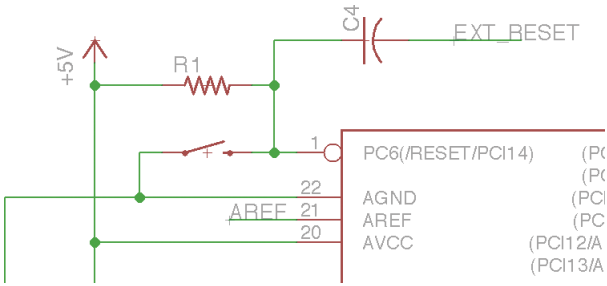



**FIGURE 3-13:** Three things have been added here: first, a pull-up resistor on the reset line so the pin doesn't "float" and randomly reset itself; second, a capacitor in line with the RESET pin on the header so that the Arduino software can automatically reset the board when programming; and third, a decent-sized reset switch that will connect RESET to ground when pushed.


**FIGURE 3-14:** A pilot light wastes a little bit of current but is a handy feature.

Position the switch, the 10K resistor, and the remaining 0.1µF capacitor as shown. Connect one side of the switch to ground.

The switch is normally open but pulls RESET low when pressed. The 10K resistor is connected between RESET and +5V so that the pin is not "floating" (which can cause erratic behavior) when the switch is open. During programming, the host computer pulls EXT_RESET low to reset the chip before the bootloader starts loading code. The small capacitor keeps the timing of this pulse within the limits expected by the system.

Finally, add an LED in series with a 1K current-limiting resistor.

## Check the Circuit: Electrical Rule Check

Run the Electrical Rule Check (ERC) by selecting ERC from the Tools menu. You'll get a list of errors and warnings; click on them to make any needed fixes. A common mistake is for lines to be very close without actually connecting.

Once your schematic has passed the ERC, you're ready to start routing the board. EAGLE has a tool that will automatically

route all the traces on the board following rules you specify in Design Rule Check (DRC).

## Route the Board

When you're ready, select File → Switch To Board and you'll be asked if you want to create a new board from the current schematic using the Board Editor. From now on you'll always want to have both schematic and board windows open whenever you're doing work. EAGLE will keep the two in sync, but only if they're both open.

When you first open the Board Editor, you'll see a jumble of parts to the left and a rectangular work area to the right (Figure 3-15). The free version of EAGLE is limited to 4″ × 3.2″, and you'll get a warning message if you try to place a part outside this boundary.

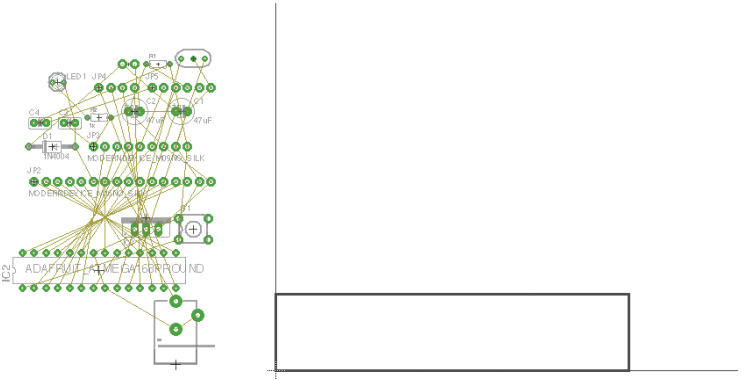


**FIGURE 3-15:** When you first create the board, all your parts will be in a jumble off to the side.

The black lines defining the boundary of the board layout are drawn on the Dimension layer. Because we're working from an existing design, the first thing to do is to draw the actual boundaries of the board; the RBBB is 3.0″ wide by 0.6″ tall. This is pretty much the smallest footprint in which you can fit all of the parts while still leaving room for some labels.

Choose the Line tool (see Figure 3-16) and select the Dimension layer from the top toolbar. Draw the Board Outline, as shown in Figure 3-15. This defines the routed edge of the board; you can draw complicated shapes as long as you keep in mind the limitations of a router bit.

**NOTE** **As with the schematic, it is very important to keep a consistent grid. The general rule of thumb is to keep the grid as coarse as you can until you need to make it finer. When routing traces, you'll need to set the grid to 0.025″ (25 mils). Do that now under View → Grid.**

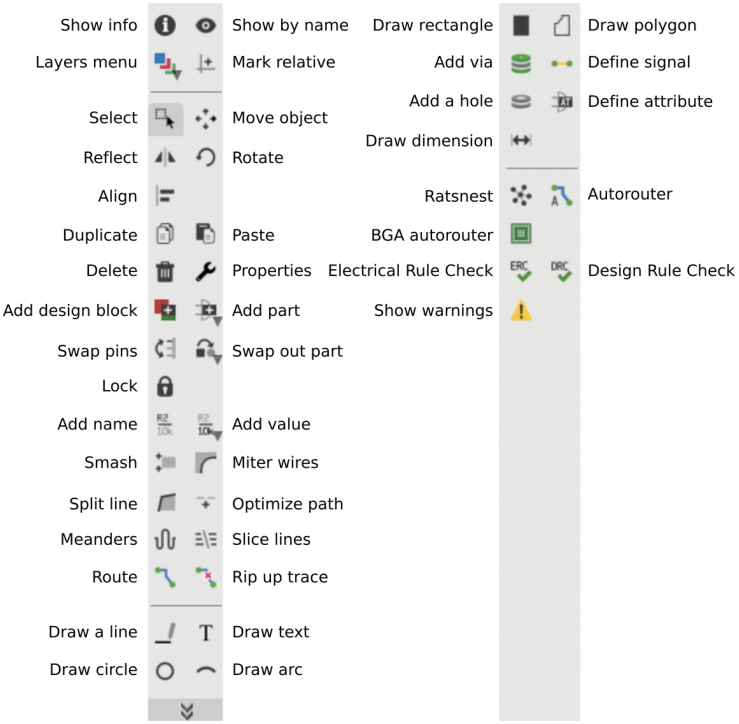



**FIGURE 3-16:** This toolbox for the board design tool has some of the same tools as in the Schematic Editor, with a few additional tools.

## THE POWER CIRCUIT

Now let's move some parts onto the board (Figure 3-17). Grab a part by clicking its crosshair, which is the part's "origin." These origins are on their own layers (tOrigins and bOrigins); if these layers aren't visible, you won't be able to grab the part to move it.

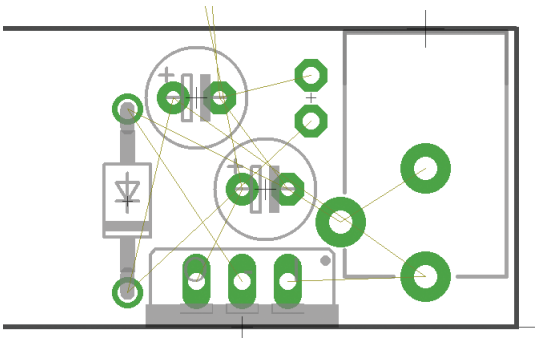


**FIGURE 3-17:** Placing the regulator, capacitors, diode, and power jack. As you move parts into place, you can use the Ratsnest tool to recompute the airwires to their closest connection point.

As before, start with the power circuit. Move the power jack, the regulator, the diode, the two power capacitors, and the auxiliary input header JP1 into place as shown. The exact placement doesn't matter too much, as long as you keep them in the rightmost 0.9″ (or so) of the Board Outline.

You'll notice the yellow "airwires" (unrouted connections) that run between the various parts, which correspond to the way the parts are connected in the schematic. As you move the parts, the airwires will follow; keep an eye on how they're connected when you're placing them and use the Rotate tool to turn parts around to make connected points closer and easier to route.

It's generally a good idea to place all the components first and route them after, but in this case we can route as we go. Select the Route Manually tool and use the default trace width of 16 mils. Select the bottom layer, and then click on the topmost

pad of the power jack; this is the Ground pad. You'll see all the other nearby ground pads become highlighted.

Route traces between all of the Ground pads, as shown in Figure 3-18. It's good practice to avoid sharp corners, the behavior of which you can change in the Wire Bend section of the toolbar.

Next, route traces to connect the positive side of the input power pads as shown in Figure 3-19. Move the 0.1µf capacitor (C3), the 1K resistor, and the "pilot light" LED onto the board (Figure 3-20). Use the Rotate tool to position them correctly, and then route the traces (see Figures 3-21 and 3-22).

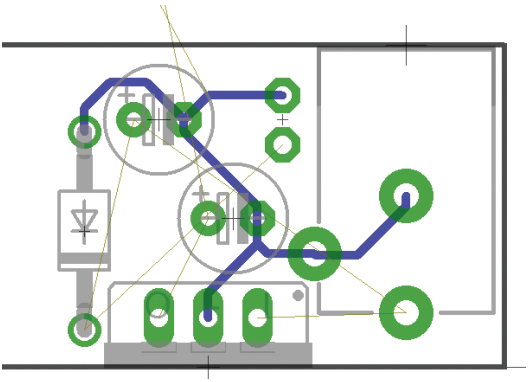


**FIGURE 3-18:** Routing all of the ground connections for the power circuit on the Bottom (blue) layer

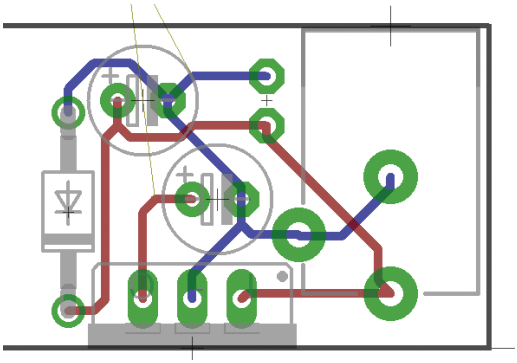


**FIGURE 3-19:** Connecting all of the positive input power pads on the Top (red) side of the board

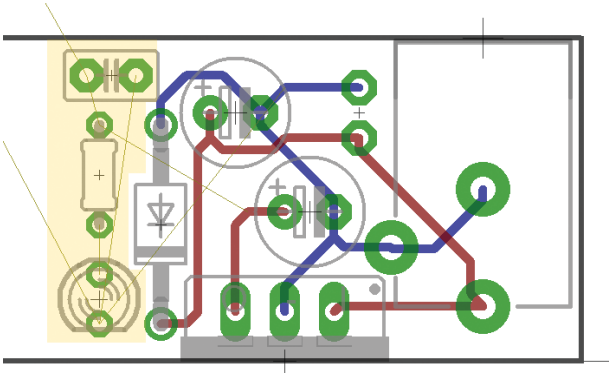


**FIGURE 3-20:** Moving the filtering capacitor and the pilot light onto the board

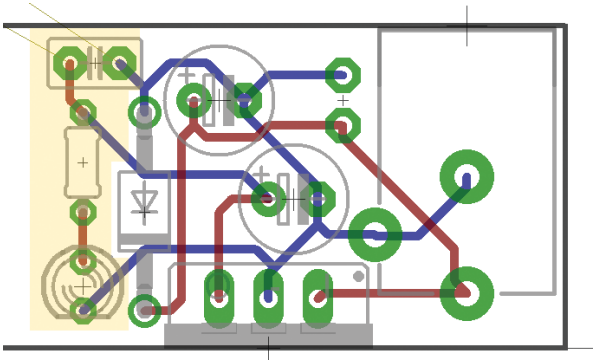


**FIGURE 3-21:** Routing the three components

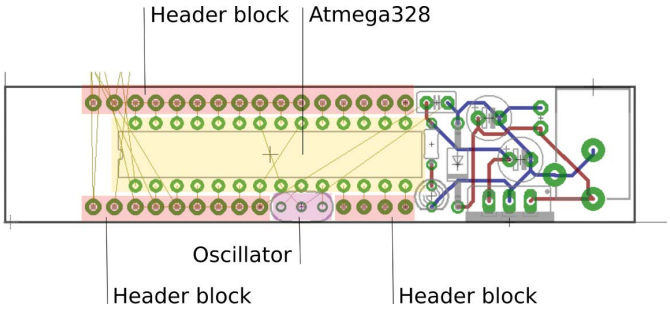



**FIGURE 3-22:** Placing the microcontroller chip and the header blocks. The oscillator should go as close to the XTAL pins as possible. Note that all of the airwires between the ATmega and the headers should be connected in a straight line, as in the figure.

## THE GPIO PINS

Place the ATmega microcontroller, the three GPIO headers (JP2, JP3, JP4), and the resonator, as shown in Figure 3-23.

Some notes on laying out the header pins:

* Place the microcontroller with the semicircle notch facing left. This indicates the side of the chip with pin 1.
* Make sure that the headers are exactly 0.5″ apart (on center) on the y-axis. That way they'll fit perfectly across the "trench" in the center of a standard breadboard.
* Make sure that the headers are oriented correctly. Depending on how you connected them in the schematic, you may need to use the Rotate tool. JP2 should have +5V as the leftmost pin, JP3 should have RESET on the left, and JP4 should have D4 on the left.
* The resonator goes right next to ATmega pins 8, 9, and 10. The high frequency signal lines between the resonator and the chip should be as short as possible, and other signals should be kept away from the area around and beneath the resonator, to prevent unwanted radio frequency (RF) interference. Make all of these connections on the Top (red) layer.

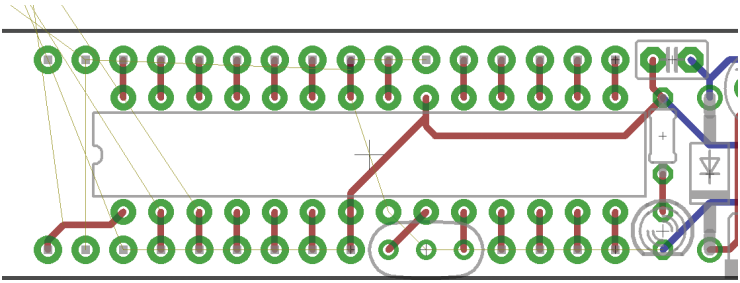


**FIGURE 3-23:** Routing the connections between the ATmega pins and the header blocks

Start routing with the signal and power lines on the top layer of the board (Figures 3-24 and 3-25). Another good rule of thumb is to keep your ground connections all on the bottom layer if possible. That way, they can all be connected together in the largest possible ground plane.

Connect all the ground pads on the bottom layer.

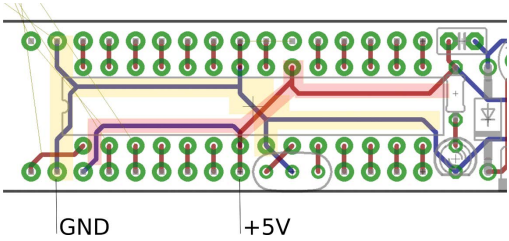



**FIGURE 3-24:** Routing the +5V (pink highlight) and GND (yellow highlight) signals

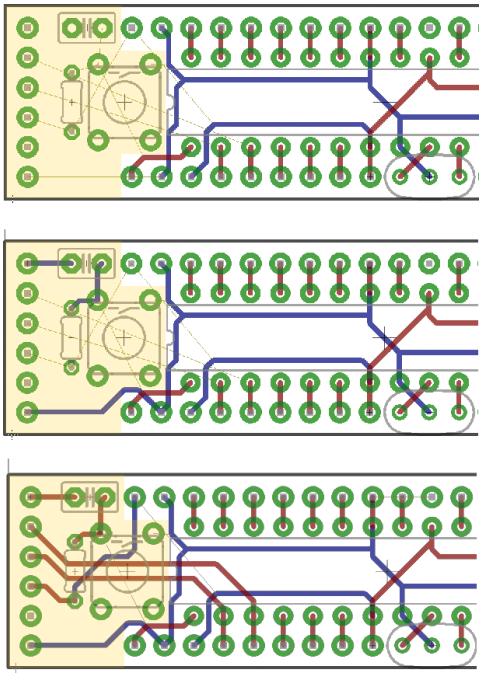


**FIGURE 3-25:** Move the remaining parts onto the board (top) and route the traces on the bottom (middle). Finally, route the remaining three signals on the top layer.

## THE RESET SWITCH AND SERIAL HEADER

At this point you should only have four parts left: the reset switch, the 10K reset pull-up resistor, the reset capacitor, and the 1×6 serial header.

Now only two unrouted signals should remain as yellow "air-wires" in your window. The +5V signal can be easily routed on the top side to other traces on the board. The reset signal—connected to the left top pad of the reset switch—will need to be routed between the top and bottom using a *via*, which is a plated hole that connects traces between two sides and/or layers of a PCB.

The solution is to add a via to jump over the trace by going to the other side of the board.

To make a via, start a new trace by clicking on the first pad at the reset switch. Route a path, as shown in Figure 3-26, until you get to the open area just before your trace will hit other traces on the bottom layer.

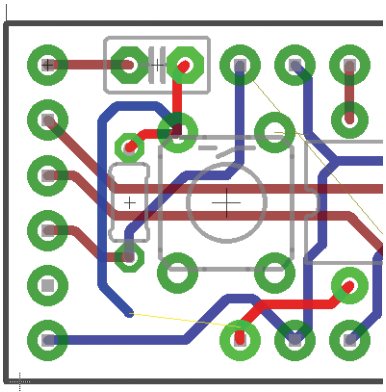


**FIGURE 3-26:** A typical routing problem: how to get from point A to point B without crossing traces on the same side of the board

With the Routing tool still active, go to the toolbar and select the top layer. Finish the route to the connecting pad on the top

layer. When you finish the path, you'll see the small plated via pad appear, as shown in Figure 3-27.

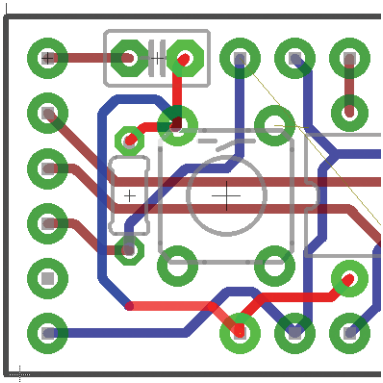


**FIGURE 3-27:** EAGLE will add a plated through-hole via at the point that the trace switches from the bottom to the top.

**NOTE** **You can use the Change (wrench) tool to change many attributes of a part after it has been placed or drawn. Change the shape of your via from square to round by first selecting Shape → Round from the pulldown menu and then clicking the via.**

## THE DESIGN RULE CHECK

The Design Rule Check (DRC) is the board layout equivalent of the ERC; select it from the Tools menu when you're done with the board. That will bring up a tabbed panel where you can set a number of design constraints to test your board against.

For example, if your PCB fabricator tells you that traces should be no closer than 6 mils, you can set that in the DRC and

see if any traces come too close. Or if you're making a beginner's kit and you want your pads to all be a little bit larger, you can set this under the Restring tab. The default values are all pretty good, though, so you don't have to change anything here and it should still turn out fine.

When you run the rule check, you'll get a window with errors and warnings (Figure 3-28). In some cases you may choose to ignore or clear warnings if you know what you're doing, but unfixed errors will almost certainly result in a nonworking board.

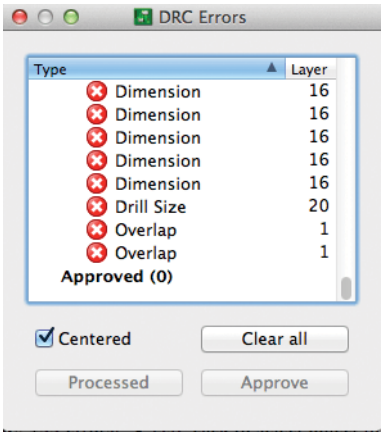



**FIGURE 3-28:** Errors caught by the Design Rule Check

## GOING FURTHER

That's all there is to routing; the only steps left are to create the silkscreen layer and the Gerber and drill files to be sent to the PCB house.

# 4

# Add a LoRa Radio to a Raspberry Pi Zero

A recent phenomenon in the Internet of Things area is the proliferation of new, long-reaching radio modules, in particular LoRa radios that can transmit data more than 10km in the open ISM radio bands (434 and 912MHz in the United States, 868MHz elsewhere). Also, the $10 Raspberry Pi Zero is a great option when you need to embed more computing power than a microcontroller in a project. In this chapter we will use surface-mount parts to design an EAGLE library for a LoRa radio module, and then connect it to a Raspberry Pi Zero (see Table 4-1).

**NOTE** This project is not technically a Raspberry Pi HAT (Hardware Attached to the Top), which is a detailed spec for Pi add-ons. It follows the physical form factor but is lacking a couple of components, like an "auto identify" EEPROM (which could be easily added). The GitHub repository for this book contains designs for a second board that has the EEPROM and 16 channels of analog inputs.

**TABLE 4-1:** LoRa for Pi bill of materials

| PART | SOURCE | PRICE ($) |
|---|---|---|
| (1) RFM95 LoRa radio module | Digi-Key 1597-1488-ND | 7 |
| (1) 10k SMT 1206 resistor | Digi-Key 311-10.0KFRCT-ND | .01 |
| (1) 10uF SMT 1206 capacitor | Digi-Key 587-1352-1-ND | .10 |
| (1) .1uF SMT 1206 capacitor | Digi-Key 399-5615-1-ND | .05 |
| (1) SMA antenna edge connector | Digi-Key CON-SMA-EDGE-S-ND | 1.77 |
| (1) 2×20 female header | 4UCON | .35 |

The finished product is shown in Figure 4-1.

## DESIGNING AN EAGLE LIBRARY

An EAGLE library is a collection of devices, so it may be more accurate to say that we're designing an EAGLE *device* in this

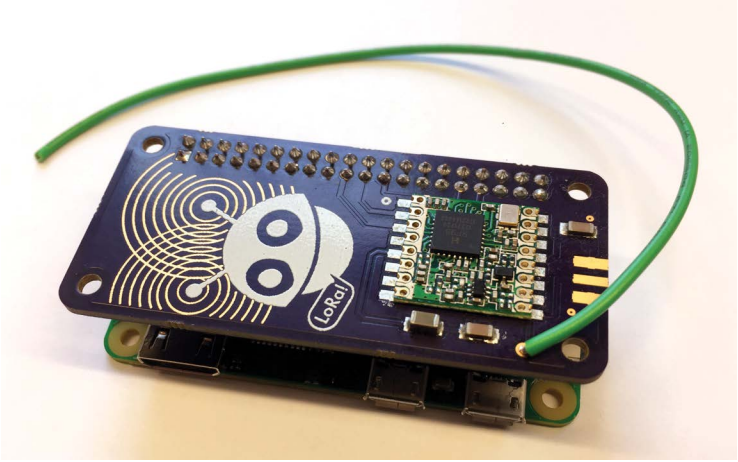



**FIGURE 4-1:** The finished board, with custom artwork on the silkscreen layer and the top solder mask layer, as described in this chapter

chapter—a device that will reside in its own library. In EAGLE, a device has three components:

* Symbol: This is the representation of the device that appears in the schematic view. It describes the interfaces and signals of the part, and it provides pins to hook it up to other parts.
* Package: This how the part appears in the board design tool, and it corresponds to the physical form factor of the part. The package provides holes or pads for connecting the part by traces to other parts. A device can come in several different packages.
* 3D package: EAGLE version 8 is much more integrated with Autodesk's Fusion 360; each library package part can have a 3D model attached to it. The 3D model can be created in Fusion 360 or in EAGLE's built-in tool, or it can be imported from 3D tools like SketchUp.

**NOTE** **For some of the library functions to work, you'll need to be online and connected to Autodesk. Use the link under your profile in the control panel to connect online. We'll be creating a local library here; see Chapter 1 for more on EAGLE's managed library feature, which requires you to be online.**

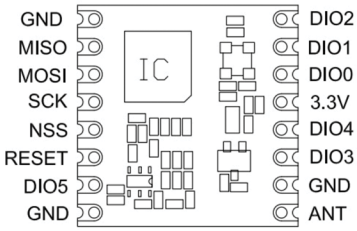
Start EAGLE on your computer. Select File ➞ New ➞ Library to open a dialog box showing the various components of the library part. We'll start with the symbol. The Add symbol will bring up a layout tool with a similar but slightly different toolbox than the Schematic Editor.

When creating a device, you'll need to have access to the datasheet for the part in question, and you'll have to do a little

hunting to find the information necessary to interpret the part as a device. At a minimum, you'll be looking for a table showing the pin numbering and functions, and a mechanical drawing showing the size of the part and pads. You may also find suggested application notes that will provide minimal hookup circuits and recommendations for filters and such.

For example, the HopeRF LoRa RFM95 module we will be using has a 121-page datasheet that can be downloaded from Digi-Key or the manufacturer (it's also in the GitHub repo): *www.hoperf.com/upload/rf/RFM95_96_97_98W.pdf*.

Figure 4-2 shows the module and the pin assignments, which appear on page 10 of the datasheet and are described in more detail on page 11.




| Number | Name | Type | Description<br>Description Stand Alone Mode |
|---|---|---|---|
| 1 | GND | - | Ground |
| 2 | MISO | I | SPI Data output |
| 3 | MOSI | O | SPI Data input |
| 4 | SCK | I | SPI Clock input |
| 5 | NSS | I | SPI Chip select input |
| 6 | RESET | I/O | Reset trigger input |
| 7 | DIO5 | I/O | Digital I/O, software configured |
| 8 | GND | - | Ground |
| 9 | ANT | - | RF signal output/input. |
| 10 | GND | - | Ground |
| 11 | DIO3 | I/O | Digital I/O, software configured |
| 12 | DIO4 | I/O | Digital I/O, software configured |
| 13 | 3.3V | - | Supply voltage |
| 14 | DIO0 | I/O | Digital I/O, software configured |
| 15 | DIO1 | I/O | Digital I/O, software configured |
| 16 | DIO2 | I/O | Digital I/O, software configured |

**FIGURE 4-2:** The pin descriptions from the module's datasheet. Note that pin numbering is counterclockwise from the upper-left pin 1.

## CREATING THE SYMBOL

With this diagram in hand, we can start laying out the schematic symbol for the part. Start with the Pin tool. Pins are connection points for signals; they have one side with a "hot spot" for making connections that should be facing outward from the part.

**NOTE** EAGLE 8.7 has a "smart paste" feature that allows you to create pin layouts by cutting and pasting tables of data from a spreadsheet using the Paste tool. This can be handy if you have a whole lot of pins and have a well-formatted datasheet.

You have a lot of leeway in how you lay out your symbol; it doesn't have to correspond to the physical layout and, in fact, usually shouldn't.

**NOTE** Don't change your default grid for the symbol; you want to make sure all of your pins are on grid!

Note that pins have default names numbered by the order they were placed. Use the Name tool to change these names to the physical pin that the signal corresponds to. Doing so will help later when connecting the symbol to the package, and the names will also appear on the schematic next to the pin to indicate the mapping (see Figure 4-3).


**FIGURE 4-3:** Adding pins. Group pins functionally in the schematic symbol. In this case, all the module's IO pins are together on the right, the SPI communication interface is on the left, and power and the antenna are at the bottom.

Use the Line tool to draw a box around the pin labels. Add a label with the Text tool.

Each of the pins can be given additional attributes that help EAGLE apply the design rules when checking your circuit for correctness (see Figure 4-4). For example, each of the pins in the symbol can have their directionality marked as inputs, outputs (or both), or power pins. Use the Info tool to change the direction attribute of the 3.3V and GND pins to PWR. This enables additional design checks; you could also change the direction of the other pins to match the table on page 11 of the datasheet.

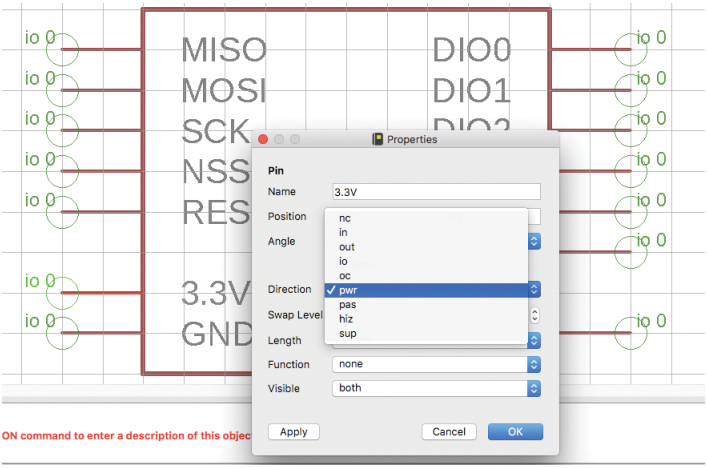



**FIGURE 4-4:** When you set the direction parameter, you indicate that the power pins are in a special class. This allows EAGLE to make additional checks with the design rules.

Save the library; by default it will be saved in the lbr directory of your EAGLE installation. EAGLE libraries are XML files with an eagle.dtd doctype and an .lbr extension. EAGLE also periodically autosaves your work in a file in the same directory with a .l#1, .l#2, etc. extension.

## CREATING THE PACKAGE

Now let's move on to the package, which is what will appear in the board view. Some parts can have the same symbol but come in various standard packages; you may see package names like SOT23, DIP, DPACK, and QFN. The RFM module is in its own custom package footprint, so call it something like **RFM95_MODULE** in the Library Manager.

For the package, we need to find the mechanical drawing of the part in the datasheet. Usually it is toward the end of the document; for the RFM95 module, refer to the package drawing on page 120 of the datasheet (see Figure 4-5).

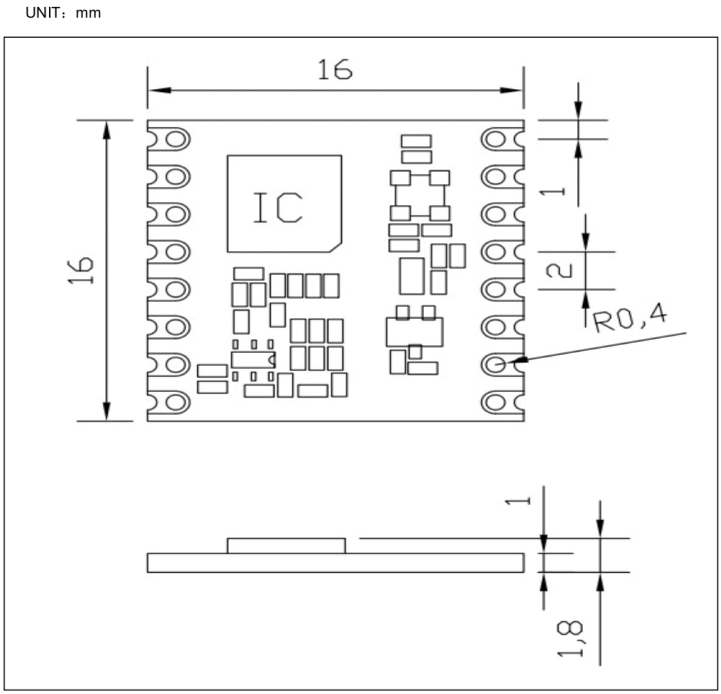



**FIGURE 4-5:** Mechanical drawings for parts can range from the verbose to the cryptic. This datasheet sketch is somewhere in between, with a couple of ambiguous dimensions, like the pad size on the module and tolerances for variations in parts.

Some notes on interpreting mechanical drawings:

* They are almost always in millimeters (mm) unless otherwise specified. Sometimes Imperial dimensions aren't labeled as such but appear in parentheses.
* Sometimes there is a suggested land pattern, and sometimes you need to calculate some dimensions.
* In some cases, there may be additional application notes for ground planes for heat dissipation.
* When in doubt, get out some calipers and measure the actual part.
* After you draw a land pattern, print it out to scale and place the actual part on the paper as a sanity check.

The RFM95 module is 16mm square, and there are eight pads per side. By measuring the pads on the actual module, you'll find they are just about 1.5mm wide and a little more than 1.5mm long. The datasheet says the pitch of the pads is 2mm on center, which leaves .5mm between each pad. We will make pads that the module can sit on, with a little bit of pad sticking out the side so the module can be soldered by hand if needed. The pads will be 1.5mm×3mm so that they can be easily centered on the module outline.

Start by setting the grid to something that matches the pad pitch. In this case we can set a 1mm grid. Use the default pad size to lay out 16 pads, as in Figure 4-6.

A few notes on pad placement:

* Numbering starts with 1 in the upper left, then incrementing counterclockwise.

- Use the Pad tool and set the dimension to the default (we'll change the pad size in a minute).
- Start at coordinate (–8, 15), and then place the pads 2mm apart on the y-axis. Note that you should start at 15 because the center of the pad is 1mm from the edge, as shown in the datasheet.
- As you place the pads, move counterclockwise.

When you're done, pin 1 should be in the upper left, pin 9 in the lower right, and pin 16 in the upper right. We'll use these pin numbers later to hook up to the symbol, so they have to be correct. Confirm with the Info tool if you need to.

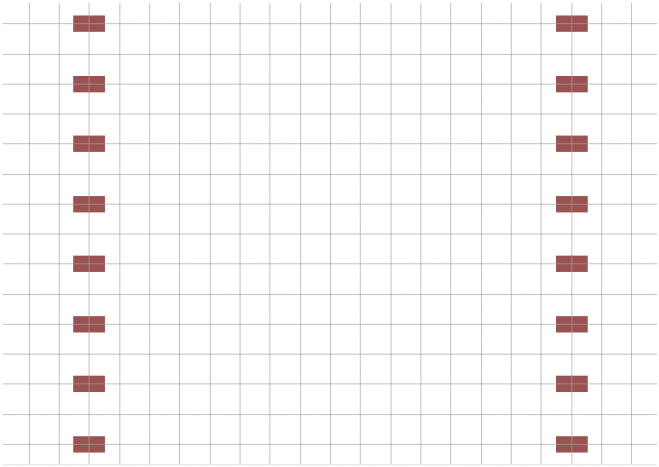


**FIGURE 4-6:** Placing the default-sized pads

Now we'll resize the pads using the Change Object Properties tool. Go to the SMD submenu (for surface-mount pads) and select the ellipsis (...), which allows you to type in a user-defined option. In the dialog box that appears, type **3 x 1.25** (that's width × height

using the units set by the grid). After you click OK, you won't get much feedback that the tool is active, but move your crosshair over the origin of each of the pads (note, as mentioned in Chapter 2, that the TOrigins layer must be active) and click on a pad to change its dimensions. It should look like Figure 4-7. Note that the pad names will be easier to read when the pads are larger; double-check them now.

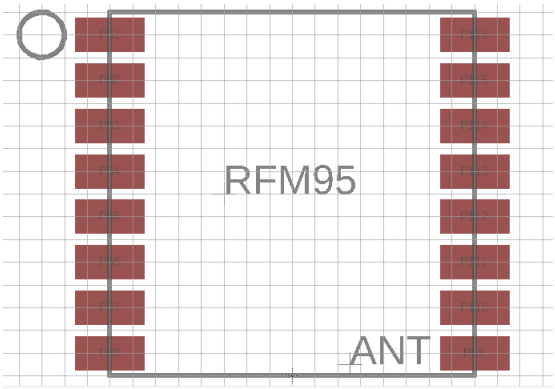
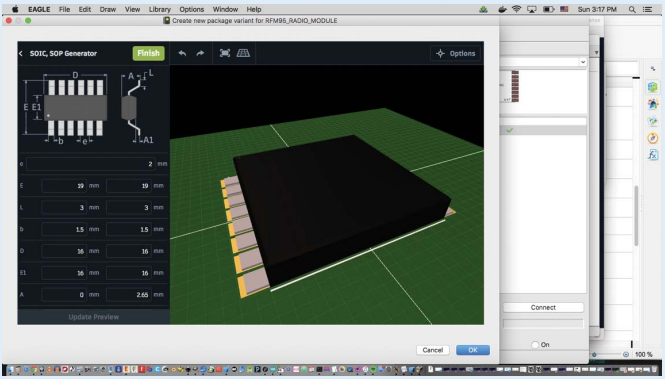



**FIGURE 4-7:** Resizing the pads with the Change Object Properties tool

Pads will have default names like P$1; you can rename them to match the mapping of the schematic pins to the pads on the board layout.

Change the layer to tPlace and draw in the outline of the module; this layer shows the dimensions of components for placement. Add a dot on pin 1 and some descriptive text for landmarks on the board (like the antenna). Confirm that the bounding box is 16mm×16mm as specified in the datasheet.

**NOTE** You can also use EAGLE 8.7's package wizard to parametrically generate the footprint of your package. This works great if your part is a variation of one of the standard packages; the wizard even automatically creates the 3D model for you. To use the wizard, skip the Add Package button and go directly to the Device area, where you'll select Create With Package Generator. An example is shown here.




## CONNECTING THE SYMBOL AND PACKAGE IN A DEVICE

Next we will create a device definition, which makes connections between the symbols and packages of the library. In the Library Manager, select Add Device and name it something like **RFM95_RADIO_MODULE**. You'll see the device connection panel, as shown in Figure 4-8.

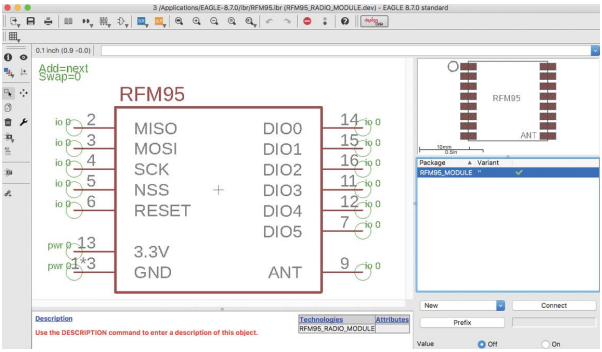



**FIGURE 4-8:** The device connection panel

First, add the RFM95 symbol to the left pane using the Add Symbol tool. Add the RFM95_MODULE to the right pane by selecting New ➞ Add Local Package. Then connect all of the signal pins of the symbol to the pads of the package by clicking the Connect button. When everything is connected, the dialog box should look like Figure 4-9.

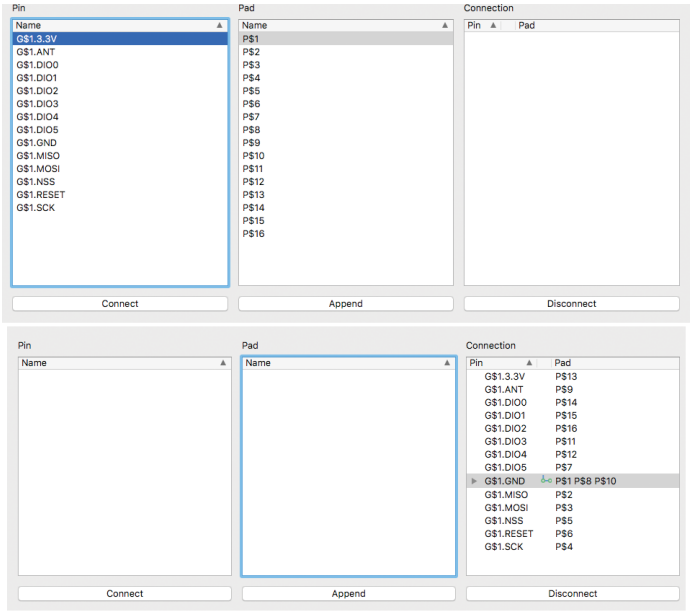



**FIGURE 4-9:** Connecting pins to signals. Use the Append option to connect the multiple grounds (pins 1, 8, and 10) together.

## USING THE NEW DEVICE IN A PCB DESIGN

Now you're ready to use the library. Restart EAGLE to load the library automatically or load the library by choosing Library → Open Library Manager → Use. We'll also be using the Raspberry Pi Zero library included in the GitHub repository, so load that as well if you haven't already.

To create the schematic, you'll follow the process of placing symbols described in Chapter 3. Start with the RFM95 module. The Pi will communicate with the radio module using the SPI serial communications pins. This is a three-wire communications protocol (with a fourth chip select line).

Connect signals to

**MOSI**: Master Out/Slave In

**MISO**: Master In/Slave Out

**SCK**: Data clock

**NSS:** Chip select. This requires a 10k pull-up resistor so it won't be in a floating state.

The datasheet recommends a .1µF filter capacitor close to the power pin on the RFM module, so add that. These will all be surface-mount parts in the large, easy-to-hand-solder 1206 format (0603 would be more common in production to save space). Add a 10µF capacitor between power and ground as well to act as a charge store for when the radio transmits. The radio part of the schematic should look like Figure 4-10.

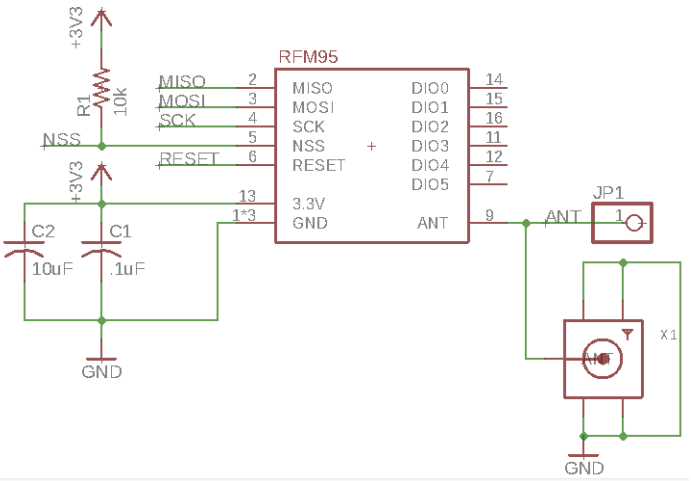



**FIGURE 4-10:** Using the RFM95 radio module in a schematic. Connecting the signals (top), then adding a pull-up resistor, a couple of capacitors to power, and a coax antenna connector.

Next, add the Raspberry Pi part from the library. You'll notice that the schematic labels all the pins on the 40-pin Raspberry Pi header. Find the SPI pins on the Pi (19, 21, and 23) and connect them by drawing a signal and labeling it the same as on the radio module. Connect the NSS select line to pin 31 (GPIO6) and the radio reset to 33 (GPIO13). These will match the radio control code that can be used later. Next, hook up power and ground and you're finished with the schematic (see Figure 4-11).

**NOTE** **The RFM95 module operates at 3.3V and its logic pins are not 5V tolerant. This is fine with the Pi, which is all 3.3V logic levels, but the module will need protection if used with 5V devices.**

Switch to the board view and move the parts onto the work area. The Pi library device has an outline on the dimension layer, so you can delete the default dimension lines and EAGLE will figure out the board shape based on the library part. Place the parts something like in Figure 4-12.

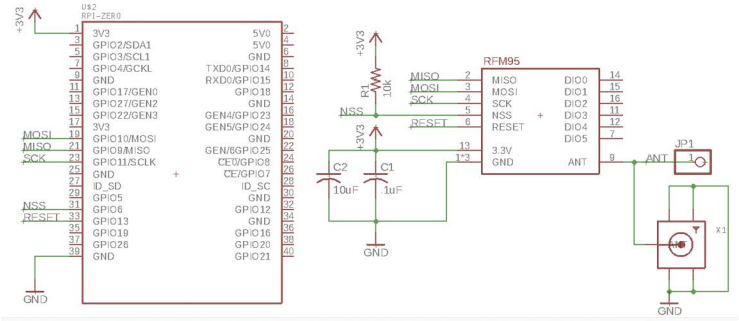



**FIGURE 4-11:** The finished Raspberry Pi Zero and radio module schematic

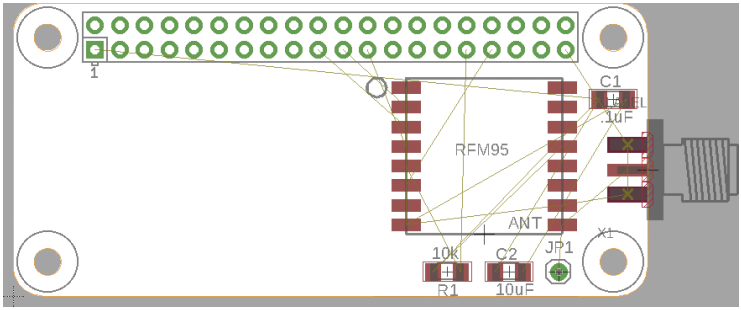



**FIGURE 4-12:** Placing the parts on the board

Route the top signals as in Figure 4-13. We're going to add a ground plane, so don't worry too much if your ground signals are all messy; they'll all be combined into a single plane later.

Next, route the bottom signals (Figure 4-14). In this case, only +3V and GND will be on the bottom.

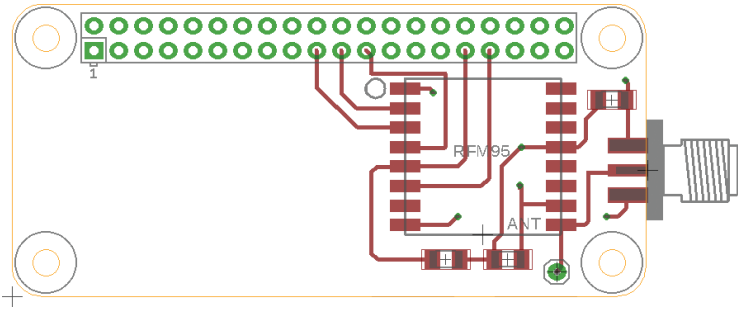



**FIGURE 4-13:** Routing the top signals

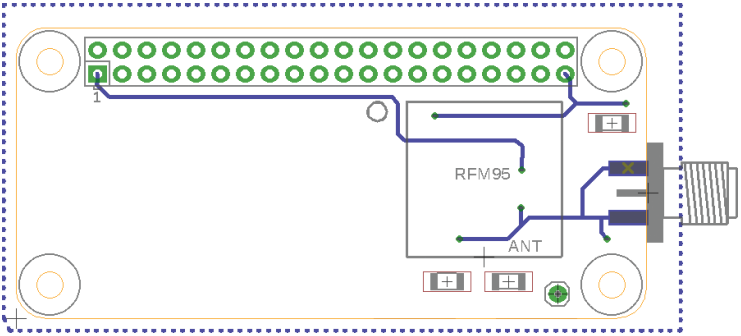



**FIGURE 4-14:** Routing the bottom signals

## ADDING A GROUND PLANE

EAGLE allows you to draw polygons into layers and give them signal names. Using the Ratsnest tool, the named polygon areas will all be unioned together into a signal plane. This is commonly used for turning extra board space into a ground plane, which can help in noise reduction and heat dissipation.

To create a ground plane, first select the Polygon tool and then draw a polygon box around the perimeter of the board. It doesn't matter how close you are to the edge; you'll see in a minute that it helps that the polygons do not overlap so they are easier to select (Figure 4-15).

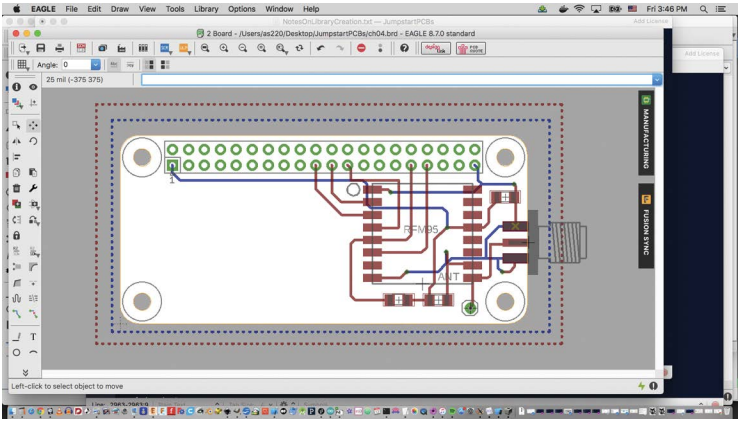



**FIGURE 4-15:** Drawing polygons for a ground plane on the bottom (blue) and top (red) layers

Now select the Name tool and click the polygon. You'll get a dialog box like the one shown in Figure 4-16. Change the name of the polygon to **GND** and it will be automatically connected to all the other GND pads and traces. When you click the Ratsnest button, you'll see the polygon fill all of the empty space on the bottom with a ground plane connected to GND.

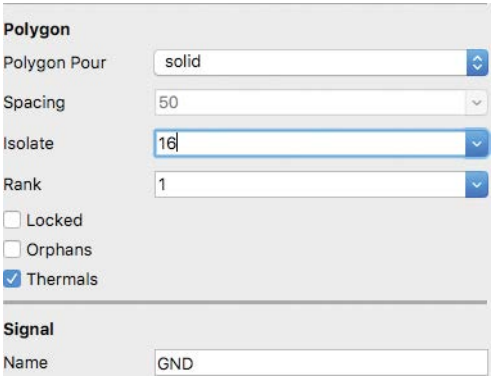



**FIGURE 4-16:** Changing the name of the polygon to GND will result in all of the ground connections contiguous with the polygon being merged into a plane. The Isolate field will make sure that there is at least that much clearance between the ground plane and any traces (16 mils in this case).

## DRAWING A CUSTOM STOPMASK LAYER

In Chapter 2 you saw how to import bitmap images into a custom silkscreen layer. You can also import bitmaps into other layers, like the top or bottom copper or stopmask layers. What is nice about drawing into the stopmask layer is that any copper left exposed by the stopmask will be plated in nickel or gold (depending on your manufacturing process), effectively giving you a second way of marking or decorating your boards. Use the bitmaps included in the GitHub repository to create the graphic shown in Figure 4-17.

Your manufacturer will usually remove any place that the silkscreen ink overlaps with pads or stopmask, so you need to account for this in generating the bitmap files (Figure 4-18).

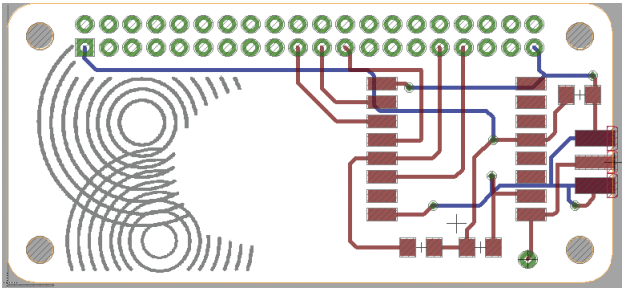


**FIGURE 4-17:** Import a bitmap into the top stopmask layer (gray).

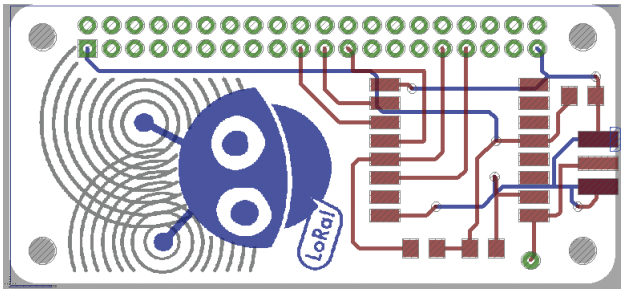



**FIGURE 4-18:** Import a bitmap into the silkscreen layer (blue shaded).

## GOING FURTHER

Once you get your boards back, you have a couple of options for assembling surface-mount parts (see Figure 4-19). This design is easy enough to solder by hand with an iron or a hot-air rework gun. If you're making a bunch of boards, you may want to go the solder paste route, in which case you'll need a stencil to apply the paste before placing the parts and cooking them.

You can order a metal solder stencil from your PCB manufacturer for around $20. Metal requires a laser in the 1000W range, so most people can't make one themselves. If you have access to a school or makerspace with a lower-power laser, you can make a perfectly fine stencil using Mylar or acetate. First export the tcream layer as a black-and-white bitmap; that's the paste layer.

The trick to creating a Mylar or acetate stencil is to use the etching settings on your laser, rather than cutting the outlines as vectors. If you're using a 35W laser, try a speed setting of 15 and a power setting of 40 at 1200dpi (your settings may vary!).

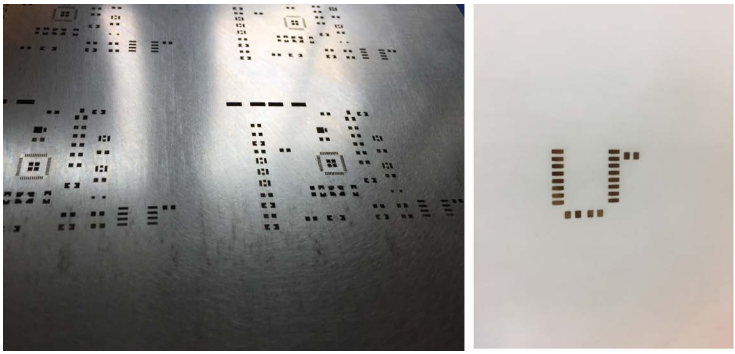


**FIGURE 4-19:** A metal stencil (left) or DIY Mylar (right)