Best Practices for Open-Source Hardware

As described in the open-source hardware definition and statement principles, the essence of open-source hardware (OSHW) is sharing the design files for a piece of hardware for others to modify or make hardware from (including for commercial purposes). There are, in addition, many other things you can do to encourage the development of a vibrant community of people who use and improve your open-source hardware project. This document discusses these best practices.

Elements of an Open-Source Hardware Project

Here are some files that you should consider sharing when publishing your open source hardware project. You are not required to post them all, but the more you share the more the community benefits and the higher the likelihood the community will pick up your project.

Overview / Introduction

Your open-source hardware project should include a general description of the hardware's identity and purpose, written as much as possible for a general audience. That is, explain what the project is and what it's for before you get into the technical details. A good photo or rendering can help a lot here.

Original Design Files

These are the original source files that you would use to make modifications to the hardware's design. The act of sharing these files is the core practice of open-source hardware.

Ideally, your open-source hardware project would be designed using a free and open-source software application, to maximize the ability of others to view and edit it. For better or worse however, hardware design files are often created in proprietary programs and stored in proprietary formats. It is still essential to share these original design files; they constitute the original "source code" for the hardware. They are the very files that someone will need in order to contribute changes to a given design.

Try to make your design files easy for someone else to understand. Organize them in a logical way; comment complex aspects; note any unusual manufacturing procedures; etc.

Examples of Original Design Files include:

- 2D drawings or computer-aided design (CAD) files, such as those used to describe two-dimensional laser cut, vinyl cut, or water-jet cut part, in their original format.
  Example formats: Native 2D design files saved by Corel Draw (.cdr), Inkscape (.svg), Adobe Illustrator (.ai), AutoCAD, etc.
- 3D designs that can be 3D printed, forged, injection molded, extruded, machined, etc.
  Example formats: Native files saved by SolidWorks (.sldprt, .sldasm), Rhino, etc.
- Circuit board CAD files such as capture files (schematics) and printed-circuit board (layout) design files.
  Example formats: Native files saved by Eagle, Altium, KiCad, gEDA, etc.
- Component libraries (symbol, footprint, fastener, etc.) necessary for native modification of CAD files.
- Additional technical drawings in their original design formats, if required for fabrication of the device.
- Additional artwork that may be used on the device and is included as part of the OSHW release, such as an emblem, or cosmetic overlay in the original design format.

In the event that a design was originally created in an alternative format, even one that might normally be considered as an auxiliary design file (as discussed in the following section), that original design in the original format could be considered the “original design files”.

Examples of alternative formats that could constitute original design files under special circumstances include:

- Hand-coded G-code for a machined part. (G-code)
- Scans of hand-drawn blueprints. (JPEG)
- Detailed 3D scans of a hand-carved resin-casting mold. (STL)
- Mask pattern for etching a single-side circuit board, as drawn in MS Paint. (PNG)

Auxiliary Design Files

Beyond the original design files, it is often helpful to share your design in additional, more accessible formats. For example, best practice open-sourcing a CAD design is to share the design not just in its native file format, but also in a range of interchange and export formats that can be opened or imported by other CAD programs.

It is also helpful to provide ready-to-view outputs that can easily be viewed by end users who wish to understand (but not necessarily modify) the design. For example, a PDF of a circuit board schematic, or an STL of a 3D design. These auxiliary design files allow people to study the design of the hardware, and sometimes even fabricate it, even without access to particular proprietary software packages. However, note that auxiliary design files are never allowed *as substitutes* for original design files.

Examples of auxiliary design files include:

- 2D drawings or CAD files, in a 2D export or interchange format.
  Example formats: DXF, SVG
- 2D drawings or CAD files, in an easily viewable 2D export format.
  Example formats: PDF, JPEG, PNG, etc. (Where possible, vector formats are preferred over bitmap formats.)
- 3D designs or CAD files, in a 3D export or interchange format.
  Example formats: STEP, IGES
- 2D or 3D designs in manufacturing-ready export formats
  Example formats: G-code, STEP-NC, STL, AMF
- Circuit board design files in export or interchange formats.
  Example formats: EDIF, Open JSON
- Circuit board designs in manufacturing-ready formats
  Example formats: Gerber RS-274X, Excellon
- Additional technical drawings in their original formats, if required for fabrication of the device, in a commonly-readable format such as PDF.
- Additional artwork, for example different colored skins for an instrument panel.

Bill Of Materials

While it might be possible to infer from the design files which parts make up a piece of hardware, it is important to provide a separate bill of materials. This can be a spreadsheet (e.g. CSV, XLS, Google Doc) or simply a text file with one part per line. If your CAD package has integrated or add-on BOM management tools, those are also a good option. (Examples include the built-in tools in SolidWorks and bom-ex for Eagle.) Useful things to include in the bill of materials are part numbers, suppliers, costs, and a short description of each part. Make it easy to tell which item in the bill of materials corresponds to which component in your design files: use matching reference designators in both places, provide a diagram indicating which part goes where, or otherwise explain the correspondence.

Software and Firmware

You should share any code or firmware required to operate your hardware. This will allow others to use it with their hardware or modify it along with their modifications to your hardware. Document the process required to build your software, including links to any dependencies (e.g. third-party libraries or tools). In addition, it's helpful to provide an overview of the state of the software (e.g. "stable" or "beta" or "barely-working hack").

Photos

Photos help people understand what your project is and how to put it together. It's good to publish photographs from multiple viewpoints and at various stages of assembly. If you don't have photos, posting 3D renderings of your design is a good alternative. Either way, it's good to provide captions or text that explain what's shown in each image and why's it's useful.

Instructions and Other Explanations

In addition to the design files themselves, there are a variety of explanations that are invaluable in helping others to make or modify your hardware:

Making the hardware. To help others make and modify your hardware design, you should provide instructions for going from your design files to the working physical hardware. As part of the instructions, it's helpful to link to datasheets for the components / parts of your hardware and to list the tools required to assemble it. If the design requires specialized tools, tell people where to get them.

Using the hardware. Once someone has made the hardware, they need to know how to use it. Provide instructions that explain what it does, how to set it up, and how to interact with it.

Design rationale. If someone wants to modify your design, they'll want to know why it is the way it is. Explain the overall plan of the hardware's design and why you made the specific choices you did.

Keep in mind that these instructions may be read by someone whose expertise or training is different from yours. As much as possible, try to write to a general audience, and check your instructions for industry jargon, be explicit about what you assume the user knows, etc.

The instructions could be in a variety of formats, like a wiki, text file, Google Doc, or PDF. Remember, though, that others might want to modify your instructions as they modify your hardware design, so it's good to provide the original editable files for your documentation, not just output formats like PDF.

Open-Source Hardware Processes and Practices

Designing your Hardware

If you're planning to open-source a particular piece of hardware, following certain best practices in its design will make it easier for others to make and modify the hardware:

- Use free and open-source software design (CAD) tools where possible. If that's not feasible, try to use low-cost and/or widely-used software packages.
- Use standard and widely-available components, materials, and production processes. Try to avoid parts that aren't available to individual customers or processes that require expensive setup costs.

Hosting your Design Files

A basic way of sharing your files is with a zip file on your website. While this is a great start, it makes it difficult for others to follow your progress or to contribute improvements.

We recommend using an online source-code repository (like GitHub, Gitorious, or Google Code) to store your open-source hardware projects. All files (design, bill-of-materials, assembly instructions, code, etc) should be version controlled where possible. If you want to develop your hardware publicly, online repositories make it easy to publish changes to your files as you make them. Or, you might publish updates in conjunction with releases of the hardware.

Most online repositories also include issue trackers, which are good way to keep track of the bugs in and future enhancements planned for your software in a way that others can view and comment on. Some include wikis, which can be good places to document your project.

As an alternative to an online repository, you might develop your project in an online CAD tool (like Upverter). Or, you could share your files on a site like Thingiverse.

Licensing your Designs

While licensing is a complex subject, use of licenses is an important way of signalling how others can and should use your work. By explicitly applying an open-source license to your hardware design files and other documentation, you make it clear that others can copy and modify them. When licensing your project, keep in mind that someone who makes a derivative of your hardware will probably also want to build on your software, instructions, and other documentation; you should license not just the hardware design files but also these other elements of your project.

Note that copyright (on which most licenses are based) doesn't apply to hardware itself, only to the design files for it – and, then, only to the elements which constitute "original works of authorship" (in U.S. law) and not the underlying functionality or ideas. Therefore, it's not entirely clear exactly which legal protections are or aren't afforded by the use of a copyright-based license for hardware design files – but they're still important as a way of making clear the ways in which you want others to use your designs.

There are two main classes of open-source or free-software licenses: copyleft (or viral) licenses which require that derivatives be licensed under the same terms; and permissive licenses, which allow others to make modifications without releasing them as open-source hardware. Note that the definition of open-source hardware specifies that you must allow modification and commercial re-use of your design, so avoid licenses with a no-derivatives or non-commercial clause.

Popular copyleft licenses include:

- Creative Commons Attribution, Share-Alike (BY-SA)
- GNU General Public License (GPL)

- Hardware-Specific Licenses: TAPR OHL, CERN OHL

Permissive licenses include:

- FreeBSD license
- MIT license
- Creative Commons Attribution (BY)

It is good practice to include a copy of the license in the version control repository, and a statement in every file or at least the README specifying the author(s) and year(s) of non-trivial modifications, and the license.

Distributing Open-Source Hardware

- Provide links to the source (original design files) for your hardware on the product itself, its packaging, or its documentation.
- Make it easy to find the source (original design files) from the website for a product.
- Label the hardware with a version number or release date so that people can match the physical object with the corresponding version of its design files.
- Use the open-source hardware logo on your hardware. Do so in a way that makes it clear which parts of the hardware the logo applies to (i.e. which parts are open-source).
- In general, clearly indicate which parts of a product are open-source (and which aren't).
- Don't refer to hardware as open-source until the design files are available. If you plan on open-sourcing the product in the future, say that instead.

Building on Open-Source Hardware

- Respect the trademarks of others.
- Make useful improvements to a piece of hardware rather than simply selling copies of it.
- Share your changes and improvements with the creator of the original hardware.