**GOSH 2017:**

Hardware documentation   
and sharing workshop

***Tobey Wenzel & Luis Felipe R. Murillo*  
(Journal of Open Hardware)**

### Introduction

This document was prepared for the “hardware documentation and sharing workshop” at GOSH 2017 in Santiago, Chile. It is based on Open Hardware guidelines, such as the “Best Practices for Open-Source Hardware[1]” from the Open Hardware Association (OSHWA), the “Guide for Open Source Hardware[2]” from “OPEN! Methods and tools for community-based product development project, and the Docubricks’ “Quality sharing best practices for Open Science Hardware[3]”. We included here a summary of these documents to help designers and engineers prepare Open Hardware projects. It follows the general orientation of the community-vetted Open Source Hardware definition [x] and the GOSH manifesto for Open Science Hardware [x].

#### **What is “hardware documentation”?**

Hardware projects are developed through a series of steps including design and prototyping, fabrication, testing, and distribution. In this workflow, “hardware documentation” describes each step to make the project accessible and reproducible. In general terms, documentation includes: rationale for the project, bill of materials, design files, electronics schematics, circuit board layouts, mechanical drawings, software source code for hardware dependencies, images, videos, and other explanatory material with (software and hardware) build instructions.

#### **How to make hardware open?**

Hardware documentation is “open” when its wide accessibility and unrestricted circulation for protected by an Open Hardware license. The “Open Source Hardware” (OSHW) definition was vetted by the Open Hardware community and describes the basic conditions for a hardware project to be considered “open:”

Open Hardware licenses are based on the OSHW definition. There are several options for different needs of a particular project: Ballon Open Hardware license, TAPR OHL, CERN OHL, and Solderpad Hardware License. Some projects adopt Creative Commons licenses for design files, but they are not considered useful

Hardware documentation needs to be:

**- Accessible:** it has to written in a simple and clear language without sacrificing technical detail. Be generous in the use of media such as photos and diagrams, these are key helpers to communicate all aspects of functionality and handling. Accessibility also means having the freedom to study, modify, use, and redistribute a particular project for whatever need you might have.

**- Sufficient:** it must be complete in the sense of including all files and instructions that are needed for a project to be replicated, ideally in a modular structure.

**- Targeted:** documentations must specify their target group. The instructions that need to be provided heavily depend on the knowledge that can be taken for granted in that target group.

**- Findable:** projects must be documented in locations that a user would expect (in the code, in a public repositories linked to from community webpages, and in open access journals).

**- Open from the start:** openness starts with the design process. Keep in mind what resources the target group has access to and explain your design rationale in the documentation. Open hardware only becomes really useful when others can build on the design.

#### **Key Elements of Hardware Documentation**

Documentation is prepared at several phases of a project: design/prototype, fabrication, testing, build instructions, and publication. These elements are discussed in detail below:

1. **Description:** a quick description of what your hardware is, what it does, and what it is originally intended for is necessary to get you started. If it derives from another project, say so in order to help people who might be familiar with the project you used to bootstrap your derivative work. In the process of describing the project, select an Open Hardware license and attach the text of the license to the body of our file repository. Below we describe different licenses and where to get help to define what license you might need for the future of your project.
2. **Audience:** skill levels and pre-requisites for building your hardware must be described upfront. If you use surface mount components, for instance, you might want to warn people who do not have experience with the technique to seek help or train themselves up before starting the build process. The less frustration you generate through the build process, the better itself for your project and the community as a whole.
3. **Accessibility:** to ensure “openness” on various levels, you need to select an Open Hardware license for your project. Your choice depends on the nature of our project. The most recommended path is for you to use a strong-copyleft license in order to ensure that derivative projects return to the common pool of Open Hardware documentation they draw from. This is generally the best approach because it fosters a circuit of contributions to keep projects alive (with the best documentation and community support). Copyleft licenses include hardware-specific ones from [CERN](http://www.ohwr.org/cernohl) and [TAPR](https://www.tapr.org/ohl.html). Permissive licenses include the Solderpad license, which is inspired by the Apache license.You should apply one of these licenses, or another one to meet the [OSHWA definition](http://www.oshwa.org/definition/) (which is not a license, but a community-vetted document, think of it as a form of social contract) to your project documentation. **IMPROVE HERE the description of the licenses.**
4. **Parts list:** bill of materials (BOM) and special tools needed must be listed in this section. The best approach here is to describe the part, its part number, the distributor you have used, and the code for the distributor (if it has an SKU, you can use it or any other unique identifier which makes it easier for adopters to purchase the part list at once). There are interesting solutions developed by the community in which you can convert your BOM to a particular format and then automagically fill an e-cart at a particular electronics distributor: URL for the TOOL here. SHOW EXAMPLE of a table describing BOM.
5. **Software Dependencies and Required Tools:** your project may require software and hardware tools, so you must describe them. Open Hardware projects cannot be truly “open” if the tools to test, study, modify, fabricate, adapt, and redistribute are not free (as in *freedom* not *price*). Instead of assuming people know, it is a good idea to explain what is in your toolchain (according to the Wikipedia, a toolchain is a “set programming tools that are used to create a software product” which, in the case of hardware, might be a firmware file for instance). It is also a good idea to describe which programmer you are using (if needed), what revision number it has, its platform and software requirements, etc. EXAMPLE.
6. **Build instructions:** step-by-step demonstration on how to build a particular piece of hardware (with all its software dependencies). This is a tricky part of your documentation. You must include, if you can, images, videos, and diagrams to facilitate the work for newcomers. You must also attempt to describe procedures in a clear without sacrificing technical detail. This is a complicated balance, so it is a good idea to co-write the documentation with other people who are as involved with the technical aspects of the hardware. Best examples of documentation we have include easy-to-follow explanations of the techniques that were used to accomplish a certain task (i.e. to fit more data into a limited space of a flash memory of an microcontroller through compression). EXAMPLE.
7. **Fabrication instructions:** this is a crucial part of the process, if your hardware depends on manufacturing (for small or large batches, or for simple or more complicated electronics projects which require automation). Most manufacturers will not be able to manipulate your KiCAD files, so it is a good idea to stick with Open Hardware distribution companies offering PCB and related fabrication services. In the fabrication instructions, you might want to specify XXXXX YYYYYYYYYYY ZZZZZZZZ.
8. **Calibration and testing procedures:** after the first batch is fabricated, how do you easily test and calibrate your hardware? This step is important to ensure quality before a hardware is fabricated in large quantities and distributed widely. Technical specs for calibration must be given (example here) and testing rigs can also be shared (example here). This facilitates communication with potential users, contributors, and technicians on the manufacturing end.
9. **Applications:** if you have different applications for the your piece of hardware, list them here. If you imagine potential applications, you can also list them here as a hint for potential contributors to pick your project where you left off or to take your project into an area you and your group do not have experience to venture into. In this section, you can also list all the derivative versions you project have.
10. **History:** list the changes and bug fixes for each version of the project. This is section resembles the CHANGELOG file of Free Software projects (it is very useful for users and developers to quickly get up to speed with the past and present status of your project by reading this section).
11. **Roadmap:** if you have a list of features you want to implement, list them here. You might find people willing to help you improve and extend the project! This section can also be called WISHLIST or TODO if these terms are more familiar in your community.
12. **Known Issues:** if you have come across a non-intuitive limitation of your hardware, it is a good idea to describe it here. This section will help users and contributors to have a clear idea of the limitations of your project, so they can either contribute directly to solve an issue or understand its limitations before getting started on a derivative version (which might lead to a non-obvious roadblock).
13. **Versioning and Distribution:** Neatly separating versions (public releases) of your hardware project is very useful for potential collaborators. It is a big waste of time when people have to chase down files and try out different ones to figure out which one is supposed to work on a particular set-up. Make sure you make separate versions which refer to one and only particular state of the whole documentation (which includes software source code and all the hardware design files). Distribution can be done through your favorite source code management repository, but we recommend using a repository for long-term digital preservation such as Zenodo, linking a particular public release with your project with a Digital Object Identifier (this is very useful for indexing your project, claiming prior art in patent disputes, and helping your project get published academically in case you want to also publish it as a paper). For integrating Zenodo and Github, for instance, you can follow this guide: <https://guides.github.com/activities/citable-code/> (please observe that you do not need to use Github or Zenodo, you can use other solutions to store, manage, and preserve the documentation files for your project).

### Summary

**Open Hardware documentation must include:**

* Description of the project’s functionality with indication of intended audience
* Source files (e.g. schematics, modifiable CAD files, board designs, etc.)
* Inclusion of an Open Hardware license (such as CERN OHL, TAPR, and Solderpad)
* Any useful intermediate files (e.g. STL files, print-ready PCB files)
* List of parts with supplier details, list of Free Software tools which are necessary
* Complete (software and hardware) build instructions with photos, diagrams, etc.
* Calibration and testing instructions (if needed)
* Sufficient information to recreate any custom parts required

**When documenting your project, bear in mind the following:**

* Who’s your audience? Tailor your documentation accordingly
* Indicate purpose of documentation and skill levels which are necessary to manage expectations: “proof of principle” vs. “easy to make in educational workshop”
* Include meta-information about design decisions, so your project can be extended
* Privilege Free Software tools: access to tools is a fundamental part of OSHW
* Instruction steps should be minute and explicit (w/ obvious filenames)
* Test your instructions without giving additional information - submit for peer-review
* Bonus to let it go viral: include thoughts on possible improvements and modification

**Prepare design files to ensure usefulness and extensibility:**

* Provide design files in an open format wherever possible, see: [open source hardware definition](http://www.oshwa.org/definition/)
* Share modifiable design files in addition to build files (provide more than only STL, PDF, DXF files or similar)
* Design the project to be easily adaptable for common differences (metric vs imperial) and adaptations, ideally make parametric
* Utilise asymmetric design to make parts fit together uniquely and avoid miss-assemblies
* Name files with a clear name, material specs and number of pieces

#### **REFERENCES**

**1.** <https://www.oshwa.org/sharing-best-practices/>

**2.** <http://opensourcedesign.cc/wiki/index.php/A_guide_to_Open_Source_Hardware>

**3.** Docubricks documentation: <http://docubricks.com/best-practise-guide.jsp>

**4.** DOIs for repositories: <https://guides.github.com/activities/citable-code/>

**5.** FST-01 hardware documentation: <http://wiki.seeed.cc/FST-01/>

… start… the journal documents can be found here: <https://docs.google.com/document/d/10CATc4hyk_mD_4-PUNmUAp9lCVx65xjyGj-1PS4t10Q/edit> and here: <http://openhardware.metajnl.com/>

Here is also a bullet-point guide from last year:

We feel that the available tools were not sufficient to enable sharing useful open source hardware projects that integrate different types of components. These projects should be documented in a **modular** fashion where it is easy to assess **how** the project solves a problem, whether the information is **complete** and if **calibration** strategies etc. are given. We also want to provide a **database for scientists** who would like to **cite** the documentation and files in a publication as well as demonstrate the **community impact** of each module (= important bricks) of their work (these features are coming soon). Is it not a shame for the openness of science, that experiment setups are rarely published? At the same time, scientists miss out on demonstrating the impact of these aspects of their work. Companies are also welcome to use the database and **increase the visibility** of their open source products in comparison to hosting the instructions on their own page. We would like to see more projects and companies providing open source hardware solutions!