Hardware documentation Guide

Julien Colomb

2025-08-15

Table of contents

A guide to open source hardware projects documentation

How to use this guide

- Read this first page to understand the logic of documentation and think a about why and for whom you are documenting.
- Download the template files in the home of this project, choose the incrementingtemplate or the Fulltemplate version depending on the stage of your project and your preferences.
- Fill the template: the checklists in the presentation of the steps (in the chapter Chapter 3) can help you both organize the work and easily access the information this guide gives on each element. You may also go through the template and search the book for information.
- Discuss the content with your team (you may do it in the open on an open git forge). We advice to have such discussions and team agreement at each steps proposed by the book.
- Once you have a release, make sure to get an archived version of the documentation (you may use zenodo for this), which will be available in case the git forge is not there in the future.

Background

After analyzing open source hardware documentation and existing templates (see below and Chapter 12), we are convinced that (1) documentation should grow with the different phases of a project and (2) the documentation should not restrict itself to the technical documentation necessary to reproduce the hardware (hardware replication or "making") and information to facilitates its operation, maintenance, repair, recycling, and disposal (covering the life cycle of the hardware). Hardware documentation should have an even larger scope: the goal is to facilitate collaborative work during the development of the hardware.

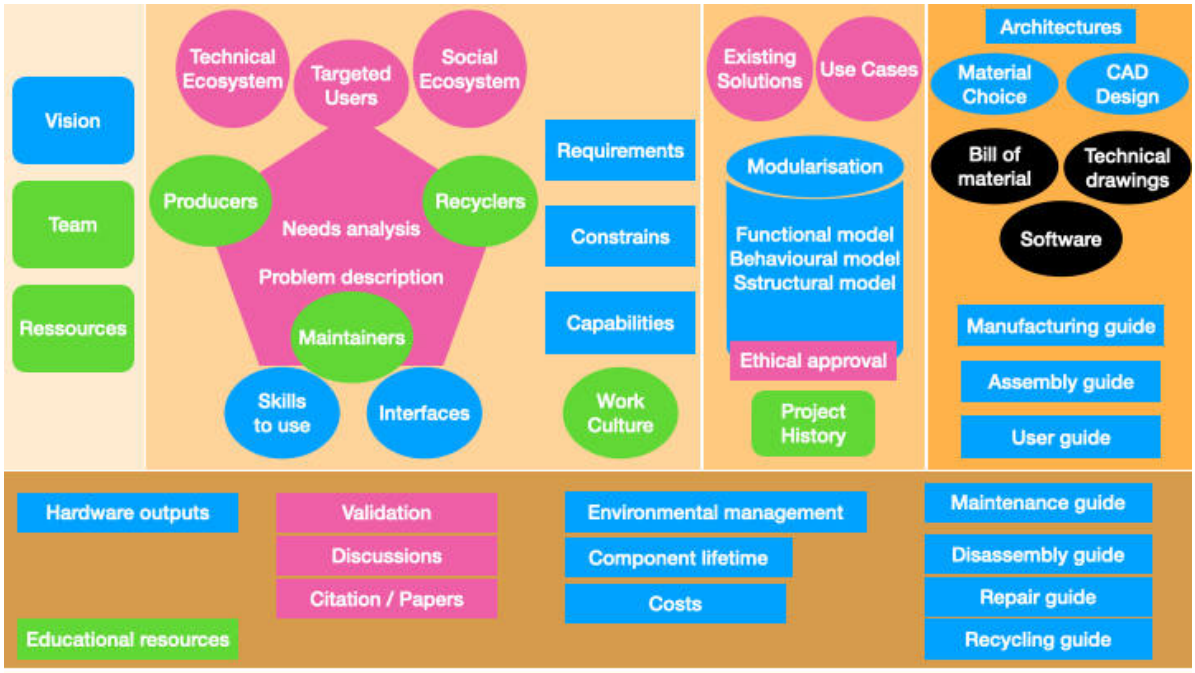


Figure 1: Hardware project documentation is diverse, it grows with project development (orange boxes get darker for each of the 5 stages: ideation, specification, concept development, prototyping, replication), and go beyond the technical documentation (black boxes) with content directed mainly on the hardware design (blue), community development (green), and business development (magenta) aspects of the hardware project. See annimated version of this image on the online and epub version of the book.

Different assets of the documentation can be mainly related to one of four objectives or work direction:

- Hardware design,
- Community building,
- Business development and
- Hardware (re-)production.

, and can be defined at different stage of the project:

1. Ideation
2. Specification
3. Concept development
4. Prototyping
5. Replication

The assets can of course be important for different aspects. For instance, the values of the community will modify the business development research which leads to different requirements that will define the hardware design and its production.

Accordingly, we created both a template for hardware project including different folders and files, and this guide. We hope to facilitate the creation of hardware project documentation with these two tools.

Rational

Open source hardware projects documentation can be divided in at least two activities: documenting the product (manufacturing, assembly, use, maintenance and so on.) and documenting the process (project vision, collaborators, how to collaborate, design process). This distinction can also be linked with the five OSHWA defined freedom for OSH: study, modify being mostly related to the process; make, distribute, and sell being mostly related to the product.

On the other hand, an hardware project goes through several development steps, each of them deserving some specific documentation. In the ideation phase, one should define the project vision and find collaborators. In the needs analysis phase, the team is documenting their research on the definition of requirements and on the related existing projects. In the concept phase, the team is describing some implementation possibilities. In the prototyping phase, the team is iteratively designing a prototype, the product documentation is starting now, while the process documentation may be expanded as new collaborators comes in. In the replicator phase, the prototype is moved to a market-ready product, and the documentation aimed at users and producers of hardware will become more important.

Guide organisation

The guide starts with a description of the steps, with a checklist of information to add at each step. This corresponds to the elements of the template in its "incremental template" form. In the checklist, each element is linked to the corresponding sub-chapters of the guide that explain what that element is.

The book chapters and subchapters follows the order of the "full template", which correspond to all elements one may include in an advanced project: readme file,community folder, project history folder, conceptualisation and specification folder, hardware design folder, and user guides folder). Some elements of the guide are not in the template, because the elements should best be organised via a different repository and linked in the readme (software, data, educational resources).

Because we expect most project to see this template when already having a prototype, we added a special chapter on starting documentation at this stage (Section 1.1).

Navigating the book

On the web version, a first table of content of chapters is found on the left bar. Once a chapter is chosen, a table of content of that chapter is available on the right bar. One can also search for specific terms.

Technicality

This is a quarto book, each element has its own quarto/markdown file and a specific code merge them together to create chapters. See the Git repository to modify this book.

Online version of the book:

URL: https://open-make.github.io/Hardware-template-guide/

Cite this book via its zenodo archive: Julien Colomb. (2025). Guide and template for hardware project documentation (v2.0). Zenodo. https://doi.org/10.5281/zenodo.16532362

Guide version: 2.1.1

- 2.0 This is the first stable version of the guide, based mostly on the theoretical framework developed inside the Open.Make project by the authors. It was checked for consistency

between steps and chapters but was not yet tested outside the authors projects and any feedback is very welcome !
- 2.1.1 New sources and new introductory chapter on metadata files were added. Box of info on book use added following community feedback.

FAQ

1. How iterative is the process ?

    - At each step, the whole content may be modified. Especially, the analysis of "Similar projects" coming in step 3, and the prototyping (step 4) are often giving new ideas and refining use cases. The vision may change when new team members enter the project, independently of the development stage.

2. What is the minimal documentation? My project is tiny.

    - The size of the project is not really affecting the number of elements that are important, it will affect the size of each element, though.
    - While you may tend to skip the community aspects of the documentation, we think these are important aspects of the development process, unless you really want to do it alone.

3. Why is the technical documentation description so small?

    - Technical documentation will be very different depending on the hardware created, we only gave general hints in this guide. As a rule of thumb, developing a hardware collaboratively with at least one other human will help you define how to organise the technical documentation.

4. When should I start documenting?

    - As soon as possible. This template allows you to start documenting your project at the ideation phase. While it might seem too early, it is useful when you want to present your ideas to collect feedback or even find collaborators. This allows to make clearer what are the important aspects of your idea, and can start interesting discussions.

1 Development stages

The documentation will grow with your project. Usually, an hardware project follows different stages of development, which may be seen as particular milestones where the team check that they gathered enough information and documented their process, before going to the next stage.

The guide propose 5 stages of developement: - ideation - specification (needs analysis) - concept development - protoyping - replication phase

1.1 From prototype, to demonstrator and market ready product

When some ideas to address a particular need are tested, the process is called prototyping. After many iterations and testing, a prototype will be selected for further development. The design that solves the needs but is not yet complete nor ready to be replicated (usually because some parts are not well documented or requires a lot of manual adjustments) is called a demonstrator. When the design and documentation are polished and are ready to be used by hardware producers, the hardware is usually labeled as a market-ready product.

Document when you already have a prototype

We expect some (or even most) readers will come to this documentation template when they actually already have a prototype, and are mostly interested in the documentation of the technical parts of the project (mostly what will be in the 04_hardware folder). Most of the advantages of documenting the project exhaustively will then be obsolete, and authors may want to restrict the document to its essentials.

As explained in the FAQ though, all elements may be important and we encourage everyone to go through each steps (even rapidely), making short description of what they had in mind when developing the project so far. These descriptions may be shorter than what you would have written if done earlier, but it may be important to ask these different questions, even when the project seems quite advanced. So, our advice is to look at the checklists of each step also when you have a prototype. You may want to use the provided template “Full_Project_prototype”, going rapidly through it to get to some documentation done fast.

There are different degrees for how open an OSH project can be, but every step in this direction is welcome and an opportunity to contribute to better research and open new career paths.

The process may help you to refine the scope of your project, potentially help you find your users and facilitate discussions with other contributors or investors.

2 Metadata

In order to facilitate discovery, your documentation may have a OpenKnowHow manifest file, where information is provided in a computer readable form. Please refer to the online documentation and tools: https://okh.makernet.org/form. Open Know-How Specification. (2022). Internet of Production Alliance. Retrieved from https://standards.internetofproduction.org/pub/okh

3 Steps

3.1 Step 1 Ideation

Ideation

At this step, you want to share your idea, usually with a small number of people, and want to "test the water".

Checklist ideation

This should all be included in the readme file:

- ☐ General information Chapter 5
  - ☐ name of the project
  - ☐ development stage: idea generation
  - ☐ *type of hardware, subject area* Section 5.1
  - ☐ For whom is this documentation made
  - ☐ **License(s)** Section 5.19
- ☐ vision and motivations Section 5.2
- ☐ short problem description Section 5.10
- ☐ Contributions
  - ☐ List of team members / contributors Section 5.16
  - ☐ skills required, who could contribute (at this point) Section 5.17
  - ☐ contact point information / communication channel and tools used for communication (this can also be one email address) Section 5.18
- ☐ Funding information
  - ☐ List of Sponsors and funding Section 5.20
  - ☐ List of putative funding opportunities Section 5.21

3.2 Step 2 Specification, Needs analysis

Needs and ecosystem analysis

Using a open source hardware canva to analyse the project may be useful at this point (defining users, contributors, communication channels, resources required).

A lot of the user analysis and the problem description part aims at the definition of the constrains and requirements for the hardware which is included in the product development part of the documentation.

It may also be time to work on community engagement.

Checklist specifications

- ☐ Complete the readme file
  - ☐ development stage: needs analysis
  - ☐ *ethics statement (human/animal use or Informed Consent Statement)* Section 5.23
  - ☐ *competing interest* Section 5.24
  - ☐ future work (Section 5.15), roadmap (Section 5.13)
  - ☐ *Project history summary* Section 5.14
  - ☐ longer problem description Section 5.10
  - ☐ Documentation structure Section 5.26
- ☐ Contributions
  - ☐ Contribution guidelines Section 6.2
  - ☐ work culture that you want to promote Section 6.1
  - ☐ Code of conduct Section 6.3
  - ☐ Governance Section 6.4
- ☐ User analysis (this can be a personas analysis)
  - ☐ Ecosystem analysis (`stakeholder`) Section 7.8
  - ☐ target groups (who will use the product) Section 7.9
  - ☐ External interfaces (how will they use the product) Section 7.10
  - ☐ skills needed to use Section 7.11
- ☐ Product development
  - ☐ requirements Section 7.1
  - ☐ constrains Section 7.2
  - ☐ capability Section 7.3
- ☐ History
  - ☐ *changes log* Section 10.1

3.3 Step 3: Concept development

Concept development

After having an idea and defining the problems, now is time to look at putative solution. This step aims at researching the technology that will be best adapted to fulfill the requirement and constrains.

If possible, one should try to define the **Modular architecture** of the hardware, which describes **architecture of functions and instructions of the product**.

An important part of this step is the research of similar project. You may end up joining an existing community and extending (adding a new module) or adapting an (or combining several) existing open hardware solutions.

Usually, this step ends with a redefinition of the needs and vision, and the three first steps often are iteratively determined until a concept is chosen for the first prototype.

> Durability and repairability constrains should be included at this point of the design. While this will be mostly documented in step 5 with repair and disassembly instructions, these concepts should be incorporated early in the design.

FBS design methodology

The concept phase is the main design phase of the hardware. While in practice, it is often made in parallel to the prototyping, larger project should invest some time at this step, and the use of the Function-Behaviour-Structure (FBS) design approach will facilitate future co-design (see Chapter 8):

- **Function (F)** stands for **"what the object is for".**
- **Behaviour (B)** stands for **"what the object does".**
- **Structure (S)** stands for **"what the object consists of".**

Checklist concept development

- ☐ Complete the readme file
  - ☐ development stage: concept development
  - ☐ dependencies Section 5.11
  - ☐ *conclusions* Section 5.27
  - ☐ software used for development Section 5.12
  - ☐ harware summary overview Section 5.3
- ☐ History

- ☐ release note Section 10.2
- ☐ design_choices Section 10.3
- ☐ update change log Section 10.1

☐ Product development

- ☐ update hardwareoverview
  - ☐ application, use cases Section 7.6
  - ☐ *reuse potential @*sec-reuse-possibilities
  - ☐ main architectural structure @sec-structural-model
  - ☐ *foreseen cost + time cost @*sec-foreseen-cost
- ☐ functional model Section 8.3
- ☐ Behavioral model Section 8.4
- ☐ Similar projects Section 7.5

3.4 Step 4: product development and prototyping

Prototyping

Here the work on the design starts! Continuous documentation of choice made, successes and failures are welcome, so this step has an iterative components: with every development can come specific documentation.

In addition, the documentation may need to be performed for different parts (or modules) of the hardware.

Importantly, the Product design, manufacturing and assembling instruction may be organised using different strategies. Some projects may write simple text files like the rest of the documentation; other projects may using Gitbuilding to write the assembly instruction, and deriving the bill of material from it; other projects may derive assembly instructions from their CAD files.

This step is also divided in two: a preparatory phase defining the main components and an iterative phase which can change with the different version of the hardware.

It is adviced to read the introductory chapters on hardware production Chapter 9, including information on modularisation of the hardware Section 9.4.

Checklist 4a: preparations

☐ Complete the readme file

- ☐ development stage: prototyping

- ☐ Standard compliance Section 5.4
- ☐ Product outputs (if relevant: data outputs) Section 5.5
- ☐ Citing information Section 5.8

- ☐ Product development
  - ☐ Structural architecture
    - ☐ Mechanical architecture Section 8.6
    - ☐ Software/firmware architecture Section 8.7
    - ☐ Electrical design architecture Section 8.8

Checklist 4b: iteration of design

- ☐ Complete the readme file
  - ☐ Update Documentation structure
  - ☐ Dependencies
- ☐ Product design (for each part/module)
  - ☐ Bill of material Section 9.7
  - ☐ material characteristics Section 9.8
  - ☐ electrical design Section 9.9
  - ☐ Software: Link to softwareve documentation, see Section 9.10
  - ☐ Product design and CAD files Section 8.9
- ☐ Software: add readme and code Section 9.10
- ☐ Manufacturing instructions
  - ☐ Manufacturing safety, skills and tools Section 9.11
  - ☐ Manufacturing sequences and instruction Section 9.12
- ☐ Assembly instructions
  - ☐ assembly skills and tools Section 9.13
  - ☐ Safety information Section 9.13
  - ☐ Assembly sequence and instruction Section 9.14
- ☐ User guide
  - ☐ Safety information Section 11.4
  - ☐ overview of the hardware Section 11.3
  - ☐ Operation instructions Section 11.5
    - ☐ Calibration instructions Section 11.5

- ☐ Testing instructions and troubleshooting Section 11.7
- ☐ basic maintenance + schedule Section 11.8

☐ History

- ☐ update changelog/release note
- ☐ update design choice history

3.5 Step 5: replicator step

Replication and maturation

Here the prototype is mature enough that it should be replicated in different places. While most of the work was already present at step 4, here we go into more quality and details.

Checklist replication

☐ Complete the readme file

- ☐ development stage: replication ready
- ☐ scientific publication Section 5.9
- ☐ education resources Section 5.7
- ☐ Institutional Review Board Statement Section 5.25
- ☐ discussions Section 5.28
- ☐ validation Section 5.6
- ☐ cost

☐ Assembly instructions

- ☐ disassembly instructions Section 9.15

☐ Product design

- ☐ component lifespan Section 11.8

☐ User guide

- ☐ Environmental management Section 11.6
- ☐ Maintenance Section 11.8
  - ☐ Instructions Section 11.8
  - ☐ Schedules Section 11.8
  - ☐ Summary component lifetime or links Section 11.8
- ☐ Repair Section 11.9
  - ☐ Identifying the defective components Section 11.9

- ☐ Repairing the defective components Section 11.9
- ☐ Replacing equipment components Section 11.9

☐ Disposal instructions

4 Metadata

Different metadata format were created. The original OKH, was transformed into the LOSH version. Unfortunately, the two metadata formats are not compatible and both versions are not used at scale.

In these conditions, it might be best to create these files on request. Giving your project an OSHWA certification (if it is an open source project) might still be the best way to make your project more discoverable.

Sources

5 Readme as entry door

Your Open Source Hardware project should include a general description of the hardware's identity and purpose (@sec-vision-and-motivation), written as much as possible for a general audience. A good photo or rendering can help a lot here (see Section 5.3).

Think about your audience when writing the OSH documentation, and tailor the documentation (and design) to them. You may have different entry points for educators, DIY makes, peer scientists, professional engineers and so on. You can start your documentation with a description of some pathways and links tailored to specific users. You should indicate at least an absolute URL pointing to a webpage or to the home repository of the documentation or its archive (in full text, not as a hidden link, such that a printed version of the documentation would link to the documentation).

Your project might be reused by people with different skills, roles, objectives, and socio-economic and cultural environments. Because of this it can be useful to create a list of skills that are required to build (manufacture Section 9.11, and assemble Section 9.13) or use the hardware (Section 7.11). Someone trying to build it from scratch, for example, will require specific set of prior knowledge, skills, and tools. A different set is needed to perform maintenance tasks. An end user operating the assembled project might require an entirely distinct skillsets (and documentation).

The readme usually starts with some main information about the hardware (name, development stage, subject area, keywords, license, languages available, latest change, date of creation). It may be good to have that information both in the readme and in a computer readable format, like a OKH metadata file.

5.1 *Type of hardware, subject area*

On top of keywords mentioned previously, one can report the Cpc (**Cooperative Patent Classification)** class (https://worldwide.espacenet.com/classification) of the hardware.

You also may indicate the purpose and/or maturity of the hardware and its documentation to manage expectations.

> Examples: proof of principle, easy to make in educational workshop

---

Sources

Section 12.1, Section 12.5, Section 12.6, Section 12.7, Section 12.8, Section 12.10

5.2 Vision and motivation

The vision provides details about the project ultimate goal, its specificity and main objectives, it answers the question:

> What, for whom and why do we have this project ?

It serves to give meaning to the whole endeavor and is a representation of what we want to achieve. It may also present the problem the project aims at solving.

It addresses the question: Why are you starting this project?

Examples:

> OpenFlexure (https://openflexure.org)
>
> The OpenFlexure project makes high precision mechanical positioning available to anyone with a 3D printer - for use in microscopes, micromanipulators, and more.
>
> Pedal-powered toolkit for fablab (https://codeberg.org/openmakeXlow-tech/PedalPoweredMachine-4fablab)
>
> This projects aims at providing fablabs and makerspaces with pedal powered toolkit, in order to open discussions around the principles of low technologies: especially questioning needs (do we need the object), and designing while recognizing the ecological impact (choice of material, improving durability and repairability).
>
> One single pedal "motor" will be connected with several tools usually requiring a rotating motor (sewing machine, saw, …). The main goal is to question the use of electrical power and show the simplicity of the tool in fablabs. The multi-functionality is important for the concept of sufficiency (less resources for a similar output). An additional goal may be to enhance collaborative work (one need two people to use the tools).
>
> We think this may also help solves the problem of "building fancy but useless objects" we sometimes see in fablabs, when objects are build to show one's skill and the possibilities of the machines, but they do not answer any needs.
>
> BCN3D
>
> The project BCN3D Moveo is motivated by the high cost of the materials that undergraduate students must use for learning how to engineer mechatronics systems.

Sources

Section 12.1, Section 12.5

5.3 Hardware summary overview

This part is meant to give an overview of the hardware, more detailed description should be given in the 03_product_dvt/hardware_overview.md file.

Give an overview of the hardware, what it does (intended use), how it was produced, and, if relevant, the research for which it has been used. Write as much as possible for a general audience. That is, explain what the project is and what it is for, before you get into the technical details. Describe how the hardware was implemented/created, with relevant details of the architecture and design, including general materials.

You may also describe any variants and associated implementation differences, as well as give numbers about how many hardware were made and by whom.

A schema, a picture or a video may be added here: 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.

> Example:
>
> The project BCN3D Moveo is an open source robotic arm that everyone should be able to replicate - with or without modification - at home without the need for highly technical knowledge and expensive materials. The robotic arm will support several of the existing training itineraries: mechanical design, automation, industrial programming, etc.
>
> Pedal powered machine: https://codeberg.org/openmakeXlowtech/PedalPoweredMachine-4fablab : photo + "size: about 700x1000 mm, table is 1100 mm high."

Sources

Section 12.1, Section 12.3, Section 12.5, Section 12.7, Section 12.10

5.4 Standard compliance

Please indicate if the hardware is compliant with standards (DIN, ISO, …).

You may also indicate why it is not, if relevant.

You may indicate:

- Standard title
- publisher
- reference
- certification status: certifier, date, link

  example

  The stairs are compliant with DIN 18065:2011-06

Sources

Section 12.7

5.5 Outputs: Products and data

This section define the product or data produced by the hardware.

It may describes examples of applications of the hardware. This should include some evidence of output, like data produced by the use of the device or a picture of other types of results.

It may also present or link to a standard data structure used, or involve the explanation of the data structure used.

One may link to other repositories or add data directly in the hardware documentation, as an extra folder.

> example
>
> Data obtained with the plastic scanner can be found at https://github.com/Plastic-Scanner/Data
>
> https://docs.smartcitizen.me: hardware documentation and data obtained are linked at the same level in the main documentation homepage.

Sources

Section 12.3

5.6 Validation

How do you know the hardware works ?

Elaborate how the hardware technically/methodologically advances the state-of-the-art, including references to relevant research articles and online references.

This may be important for future development, as new validation cycles would probably build on this documentation. Indicate any sofware/code used for validation of the design.

You can indicate whether the whether the thing has been made (and by whom) and verifies that it is makeable. This section should indicate whether the thing has been made independently using the documentation by someone who was not a contributor to the design or documentation and, therefore, verifies that the documentation is sufficient to make the thing.

Sources

Section 12.3, Section 12.7

5.7 Education resources

Indicate and link to any resources for educating the users on why/how to use the hardware. The form of these resources may be online courses, videos, or in presence workshop, to name a few.

example:

https://docs.smartcitizen.me: hardware documentation and education resources are linked at the same level in the main documentation homepage.

5.8 Cite this project

It is good practice to treat hardware citation similarly to other research outputs. The use of archives that can assign persistent identifiers (like a DOI) can help to guarantee specific versions/releases of the hardware project available over the long term. Though within the open hardware community this is not the practice, it would be beneficial to adopt going forward. For research to be reproducible, long term archiving through a platform that is dedicated to it would be necessary.

Zenodo is a good example of the type of archive that can issue a persistent identifier and provide a good citation metadata, if authors are set correctly. The use of a Citation.cff file may be recommended. You may refer to the Turing way book for more information: https://book.the-turing-way.org/communication/citable

> Example (fictive)
>
> Please cite this project by citing the article: Ho, I., Kumar, A. H., & Harris, D. M. 2022. Reconfigurable Mechanical Vibrations Laboratory Kit. Journal of Open Hardware, 6(1): 4, pp. 1–11. DOI: https:// doi.org/10.5334/joh.40
>
> Please cite this hardware as TUM Magnetic fields, & Bernhard Gleich. (2023). TUMMFE/Hardware: Hardware Release (Version 1). Zenodo. https://doi.org/10.5281/zenodo.10006096
>
> Citing White Rabbit J. Serrano, P. Alvarez, M. Cattin, E. G. Cota, P. M. J. H. Lewis, T. Włostowski et al., The White Rabbit Project in Proceedings of ICALEPCS TUC004, Kobe, Japan, 2009.

Sources

Section 12.5

5.9 scientific publication

Include experiment results or the reference to a publication (published or to be published) where these results are detailed.

You may also point to ongoing work. This may be merge with the “how to cite section”.

Example

This project lead to the publications: - Ho, I., Kumar, A. H., & Harris, D. M. 2022. Reconfigurable Mechanical Vibrations Laboratory Kit. Journal of Open Hardware, 6(1): 4, pp. 1–11. DOI: https:// doi.org/10.5334/joh.40

Sources

Section 12.3

5.10 Problem description

Describe the problem in this section and how the hardware addresses the problem.

example

As you might know, plastic recycling is important, but putting this into practice can be quite challenging. One of the big things is identifying and sorting plastic.

Discrete near-infrared spectroscopy makes it possible to identify over 75% of all plastic used in everyday life. Therefore, it became my mission to make this technology accessible to recyclers in low and middle-income countries.

Sources

Section 12.3

5.11 Dependencies

Here it is welcome to acknowledge the existing sources that have been used in this project with locations, and name their initiators. At best, present dependencies following what these projects provide as citation information. But at least:

- Initiators of the original project
- URL of the original project

You may also cite projects that project is citing as dependencies or source, with the URL of other related projects, and link to projects that are using this project as a dependency. This correponds to the OKH metadata entries variants_of, drivation_of, and sub-thing_of.

These dependencies can be hardware or software projects, modular components, libraries, or frameworks. You may indicate information on version compatibility. You should explicitly state if dependencies are proprietary / closed source.

This may also be used to present other OSH projects

Sources

Section 12.1, Section 12.3, Section 12.7

5.12 Software used for development

Indicate what software is/was used in the development, this is particularly important for (expensive) CAD software, in order for collaborators to know if they have the right software and skills to help. Also indicate what software are used for the behavioral modelling, if there is any. Always indicate software name and version, possibly adding a link to the software online home.

> Example
>
> The CAD files were created using autodesk fusion 360 (version 2601.0.90)
>
> We use the following things:
>
> - Speech - That is by far the easiest, it make communication quick and fun, ideally in person but it can also be done digitally. For super important agreements we mail.
> - Discord, for finding others interested in development of the Plastic Scanner!
> - KiCAD 6.0 - We use KiCAD as a development tool for making our PCB's.
> - Fusion360 - We use Fusion360 as a tool to do 3D modelling.
> - Wordpress, The main website runs on wordpress, easy to add content!
> - Docusaurus, for all the documentation you are reading right now!
> - GitHub, for version control of software, firmware, hardware and documentation
> - Google Drive, for internal exchange of files.
> - Youtube, for update videos
> - Notion, project management
> - PlatformIO, for firmware development

5.13 Roadmap

Provide overarching as well as short-term goals and describe expected outcomes to help contributors move away from focusing on a single idea of the feature. Describe the possible expansion of features in pre-determined and agreed on ways at stages beyond the initial implementation.

Provide sufficient information for what the expected outcomes and deliverables are.

> Example
>
> https://plasticscanner.com/future-plans/ : With a focus on improving spectroscopy, writing coding, and continuous integration, we aim to create a practical and effective solution. Be a part of the progress and join us in making a difference.
>
> In order to place the endeavors in the greater scheme of things, we have specific milestones we want to reach. We want to achieve these milestones sequentially, this makes it a plan, not a planning. We do not make any promises about when these milestones will be reached, it needs to stay fun for us to develop.

Sources

Section 12.5

5.14 Project history summary

Indicate main information about the history of the project, as well as the last updates in the project and in the documentation (especially if the documentation is not up to date).

> example:
>
> The White Rabbit project was **started in 2008** by Javier Serrano and his colleagues. From the start it has been a collaborative effort where everyone works together to make a working solution. By **summer 2012** we had built 18-port switches and used the SPEC hardware as end nodes and had made a demonstration program where one could see that the WR protocol lived up to its promises. https://gitlab.com/ohwr/project/white-rabbit/-/wikis/Status

5.15 Future work

Further work pursued by the authors or collaborators; known issues; suggestions for others to improve on the hardware design or testing, given what you have learned from your design iterations.

Sources

Section 12.3

5.16 Community, List of team members and contributors

Describe here who are the maintainers and the main contributors of the project, indicate their name, role in the project and link to further information (for instance their orcid number and/or affiliation for academic staff).

- This section may be split in different categories of contributors. For example, one can separate authors, contributors and acknowledged people. There is presently no definition of these categories or standard way to report contributions.
- For each contributor, you may indicate tasks performed (design, assembly, use cases contribution, documentation, paper writing,...)
- Avoid giving personal information (like emails) in the documentation itself. One non-personal email (or not recognisable email) can be given as a communication tool.

Sources

Section 12.1, Section 12.3, Section 12.6

5.17 Who could contribute

Mention the specific knowledge a contributor shall own to contribute to the project, as a maker or as a different role, indicate what kind of skills you are looking for.

You may describe here briefly how people can contribute to your project (link the more detailed guidelines if existant Section 6.2) and what they can expect to be rewarded. Mention what is the agreement between contributors and maintainers of the project.

> We are always looking for bright minds that want to help out making plastic identification possible. Especially data science, spectroscopy, and electronics. If you want to chip in with your skills please read the page here:

5.18 Communication channel, how to contribute

Indicate how discussion are run in the community (email to maintainer, email list, forum, social media, direct communication,...)

In first step, this can be restricted to give an email where newcomers can ask for further information. If you are using a Git Forge, the issue system of the forge may be linked here.

In developed project, a forum page or the use of a community communication tool like mattermost or matrix (to give two open source examples) is often better, as the community can work decentralized.

5.19 License and rights

Under what license is this open-source hardware documentation provided ? Specify when different parts of the documentation have different licenses

Without an open license, others cannot legally use, copy, distribute, or modify that project. The situation of hardware licensing is a bit more complex than for research outputs like software, as there are some cases where patent law and not copyright law will apply. Also note that you may use different licenses for different part of the project.

- Comparison of free and open-source software licences
- license of open hardware projects

Suggested license:

- Texts and guides: CC-BY 4.0. See: https://creativecommons.org/share-your-work/cclicenses/

- Hardware: CERN-OHL-S Strongly reciprocal (most restrictive); CERN-OHL-W Weakly reciprocal; CERN-OHL-P Permissive (less restrictive). See: https://cern-ohl.web.cern.ch
- Software: Any of the Open Source Initiative approved license (https://opensource.org/licenses)

At least one license should be indicated, use a SPDX identifier: https://spdx.org/licenses/

In general, clearly indicate which parts of a product are open-source (and which aren't).

> Example:
>
> This readme file, the 01_community, 02_project_history, and 05_user_guides folders are shared under a CC-BY-4.0 license, the 03_specification_concept and 04_hardware folders are shared under a CERN-OHL-P-2.0 license. The 11_software folder is shared under a MIT license.

Sources

Section 12.1, Section 12.3, Section 12.5, Section 12.7

5.20 Sponsors and funding

Who is sponsoring or funding your project? If funded by research grant, indicate the funder and grant number.

- URL:
- Name:
- E-mail address:
- grant number:

Sources

Section 12.1, Section 12.3

5.21 Future funding opportunities

It is often a good idea to list putative funding opportunities when the project has no long term financing. An indication of the funding strategy followed by your community is also a sign of how open the project is and will be in the future.

5.22 Administrative information

5.23 Ethics statement

State any ethics issue you project may have, and whether an ethics committee has been reviewing the project.

5.24 Competing interest

If any of the authors have any competing interests then these must be declared. The authors' initials should be used to denote differing competing interests. For example: "BH has minority shares in [company name], which part funded the research grant for this project. All other authors have no competing interests." Or "BH is selling kits and parts connected to the here presented hardware via platform XX. A fundraising via Crowdfunding platform YY is planned to start commercialisation." If there are no competing interests, please add the statement: "The authors declare that they have no competing interests."

Sources

Section 12.3

5.25 Institutional Review Board Statement

5.26 Documentation structure

How is your documentation organized?

These guidelines will provide you with a standard structure that is mainly following the product life cycle and the technological decomposition. It is implemented in the documentation template available in this project. Think about updating this aspect if necessary.

Sources

Section 12.1

5.27 Conclusions

This may include conclusions, learned lessons from design iterations, learned lessons from use cases, and/or a summary of results.

Sources

Section 12.3

5.28 Discussions

You may discuss the current state of the project. Explain how the hardware differs from other hardware, and the advantages it offers over pre-existing methods. Discuss the capabilities and limitations of the hardware. For example, load and operation time, spin speed, coefficient of variation, accuracy, precision.

Sources

Section 12.4 (coming from another section of template)

6 Community building

While scoping your project, it is well advised to think about the different people who may engage with your OSH (see Section 7.8). In the past, different OSH projects have included different partners at varying stages of their development. On top of user and contributor roles that OSH have in common with open source software, local or global hardware manufacturers may become important partners of your project. You may also think early about the people who will eventually have to maintain and repair (and recycle) the hardware.

It also helps to make your hardware designs modular (splitting your hardware in modules which may have alternative designs). Another specificity of hardware may be the importance of the creation of replication tutorials, workshops, seminars, or training materials (Section 5.7), which can impact the adoption of hardware designs. This is particularly relevant if the OSH is meant to be produced in Do-It-Yourself environments or as a teaching opportunity.

It is important to decide whether, when and where you want to engage with, or build a community. Most OSH communities are local in comparison to open source software project. You may not have the time or skills to build a community, and your project may not need a community to flourish.

Documentation which may help build a community can range from a simple note on the work culture in the readme, toward a full, implemented code of conduct and community guidelines, as well as a governance structure, on ways to interact and take decisions in the community.

6.1 Community - work culture

Describe what opportunities for collaboration different members will have. Also mentioned what recognition they may receive for their contribution. Communicate the work culture that you want to promote and policies that ensures the safety and security of both your data and people.

> Example
>
> Liquibase Manifesto
>
> The values which guide the Liquibase organization, culture, and remote work.
>
> We uphold our values of integrity, empathy, and accountability.

We encourage teamwork and celebrating achievements.

No punks, no jerks. We are committed to helping each other succeed and grow, and value our diverse backgrounds and experience.

Flexible working hours over set working hours. The results of work over the hours put in.

We empower people to challenge the status quo in surprising ways. This includes questioning meetings, processes, and product direction.

It is the DRI's (Directly Responsible Individual) responsibility to inform and include others.

We write down processes and make them easy to find.

We are transparent, and we value & strive to accept feedback.

Everyone can contribute, but the buck stops with the DRI. After a decision is made, we don't re-hash it and we stand behind it unless significant information presents itself.

We create raving fans.

Sources

Section 12.5

6.2 Community - Guidelines

Describe what opportunities for collaboration different members will have. When possible, such as in an open source project, provide these details for those outside the current group, especially when you want to encourage people outside the project to be involved.

Provide resources on ways of working to ensure fair participation of contributors who collaborate on short- and long-term milestones within the project. It reduces or addresses concerns about the project's progress towards meeting goals and prevents potential fallout between project contributors.

Considering the variety of different backgrounds and skills your members bring, describe how they can participate and start contributing. You should also think about your audience. Your project might be reused by people with different skills, roles, objectives, and socio-economic and cultural environments.

Provide clear opportunities for contributions, review, management, mentoring, and support. Provide an overview of how different contributions or resources are connected and how new contributions will fit into existing materials.

Describe how your research objects are available or will be published and how different contributors will be recognised. It helps when everyone feel appreciated and acknowledged for their contribution to the overall vision.

- A thoughtful guideline helps people decide which pathway they can choose to contribute to your project, or if they want to be in your community at all.
- Make sure that your community interactions and different pathways to contribute are open, inclusive, and clearly stated.
  - If people can't figure out how to contribute they will drop off without helping.
- Value different types of contributions - coding projects are not only about code, therefore list documentation and other management skills as well.

Sources

Section 12.5

6.3 Community – Code of conduct

Add an Open Source Project Codes of Conduct to your project, see https://opensourceconduct.com/ for examples

- This document should not be used as a token, it is very important to put intentional effort into it.
- List the expected and unacceptable behaviors, describe the reporting and enforcement process, explicitly define the scope, and use an inclusive tone.
- Whenever you update your code of conduct, invite comments from your members to ensure that their concerns are addressed.

Sources

Section 12.5: https://book.the-turing-way.org/collaboration/new-community/new-community-guide

6.4 Community - Governance

Provide a decision-making framework to facilitate discussions and reaching a shared conclusion. In the context of software and hardware, open source projects are often as much about communication as they are about coding or building (if not more). Allow informed discussions when a particular project design has reached the end or when it is useful to update it for efficiency and sustainability.

A leadership structure in an open project should aim to empower others and develop agency and accountability in your community. You can start by listing different tasks within your project and inviting your members to lead those tasks. Provide appropriate incentives and acknowledgment for the contributions made by your community members. Create opportunities for members to share some leadership responsibilities with you in the project. When inviting suggestions and ideas from the community, provide the first set of plans where your community can develop from.

More information:

- https://opensource.guide/leadership-and-governance/
- https://book.the-turing-way.org/ethical-research/ethics-open-source-governance

---

Sources

Section 12.5

7 Product development

7.1 Requirements

A requirement is a formal statement that specifies when condition C is true, property P of object O is actu

al and its value shall belong to domain D.

It is usually defined at the end of the ecosystem and user analysis.

- The minimum set of independent requirements can completely characterize the needs of the product in the functional domain.
- Functional requirements describe qualitatively the system functions or tasks to be performed in operation.
- Requirement can state as follows: The [stakeholder] need [Property] [object] [Action verb] at [Condition]

  Example of the functional requirement that ADD-ONS of XYZ cargo provides for the food producers, as a stakeholder, to preserve the quality of food.

  In this example, we assumed a refrigerator on the ADD-ONS could help the food producers to cool down and preserve the temperature of food.

  So, we defined some functional requirements (FR) based on this assumption that consist:

  - FR1: To maintain the quality of food, the food producer needs to main the material at cold temperature (between 3 °C and 10 °C) for short-term preservation (3h) or long-term preservation (24h).
  - FR2: ADD-ONS shall fix the internal ADD-ONS >temperature for 7 °C.
  - FR3: To create a cold ambient in the cooling down system, the ADD-ONS shall compress the low temperature and pressured gas to start the cooling cycle.
  - FR4: the cooling down system shall control the pressure of exit hot gas
  - FR5: the hot and pressured exit gas needs to meet the cooler external ambient temperature to become a liquid.

Sources

Section 12.1

7.2 Constrains

A constraint is a choice that makes certain designs "not allowed" or inappropriate for their intended use.

- The constraint is a restriction, limit, or regulation imposed on a product.
- There are two kinds of constraints: input constraints and system constraints.
  - Input constraints are imposed as part of the design specifications.
  - System constraints are constraints imposed by the system in which the design solution must function.

  Example XYZ Cargo ADD-ONS, constraints for maker of ADD-ONS

  - User should be able to dismantle ADD-ONS with a maximum one wrench and one screwdriver
  - Users should be able to customize the modules of ADD-ONS to fit their use.
  - The ADD-ONS should enable the users to do the assembly of components in a short time (10 minutes) and the maker shall select the resistance material for using ADD-ONS in different weather conditions.
  - ADD-ONS should be dismantled for recycling purposes.

Sources

Section 12.1

7.3 Capabilities

A service or capability is an effect intended by a actor/user resulting from the interaction of the product with its environment (i.e. what the product is for).

NB: this will relate to the user analysis section of the documentation that defines the actors and interactions.

- **Services can be stated as follows: The [Product] shall enable [the actor] to [Action verb] (for example The product shall enable end-user to clean its teeth)**
- Services provide users with an exchange value that can be included in an economic system.
- Services are intended effects that can be observed from outside the product ("black box" external view).
- Services are defined in a solution neutral-way.

  Example of services for ADD-ONS of XYZ Cargo

  - The ADD-ONS shall enable the food producer to store food
    - 1.1 solid (10 kilos)
    - 1.2 liquid (5 litrs)
  - The ADD-ONS shall enable the food producer to heat food
    - 2.1solid (150 deg Celcius)
    - 2.2 liquid (80 deg Celcius)

Sources

Section 12.1

7.4 Foreseen cost (money and time)

Indicate how expensive the hardware will be, and how much time will be needed to manufacture, assemble and test the hardware.

This is estimates done before the prototyping stage and will disappear once real numbers are available.

7.5 Similar projects

It is advisable to look for similar project one can join, instead of starting a new one.

It is reasonable to invest quite a lot of time in looking for similar projects. OSH are often difficult to find and comprehend, but finding projects may save you a lot of time and hassle. Indeed, you may learn a lot about your needs, refine your ideas, and may even find collaborators while browsing existing OSH projects.

Sources

Section 12.5

7.6 Use cases and application

Highlight at least one relevant use case, theoretical or existent.

7.7 Reuse possibilities

Indicate how you think the hardware modules could be reused for other objectives than the one you had.

7.8 Diverse actors and ecosystem

```
this is sometimes refered to a "stakeholder analysis", but we avoided that
term in this template.
```

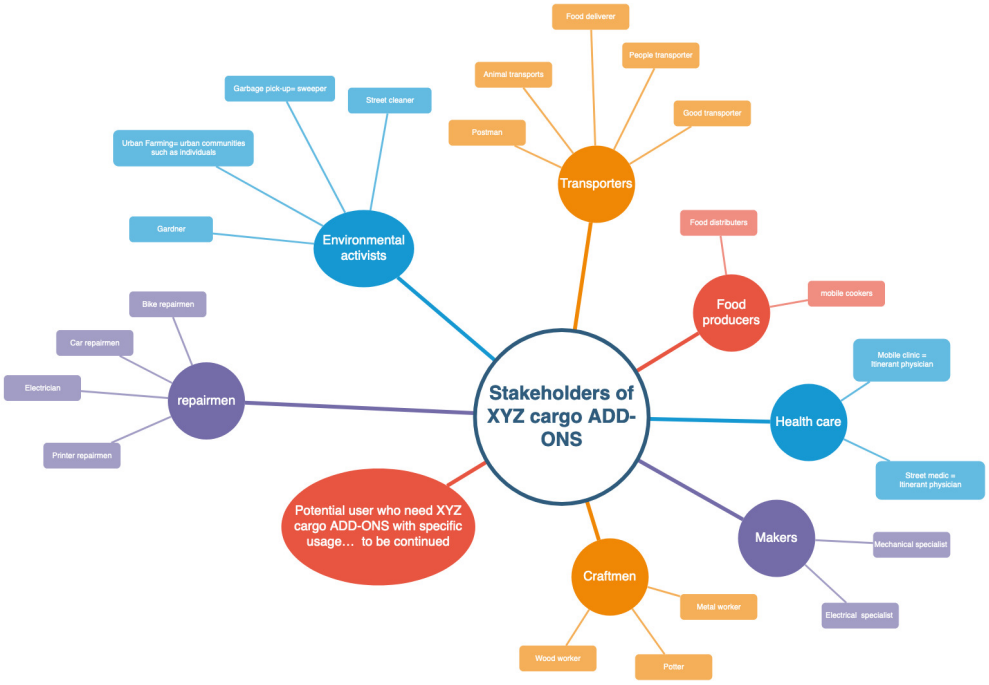
The ecosystem generally refer to all the actors (human and non-human) who (may) have an interest in a product. Among them, there are both internal players, such as users and participants of the project, and external players that are represented by the potential user of products or external entities.

- It is not necessarily a person (for example: airports as an actor when designing a two-deck aircraft).
- They can indirectly affect, be affected by the product (for example: neighborhood or biodiversity when designing an airport).

The ecosystem is often best represented via a graphics or a mindmap. This analysis may be necessary to make design choice that will fit the ecosystem inside which the hardware is supposed to work.

NB: The user target groups is one of these actors and should be determined with more accuracy, it is defined more extensively elsewhere.

Example



XYZ Cargo-ADD ONS

Sources

Section 12.1

7.9 User analysis: target groups

Target group (who will use the product) is a specific actor in hardware development, one should take much care in defining who they are and what they want. A persona analysis may be particularly interesting here.

See Section 7.8 chapter for information about actor definition and tools to map them.

7.10 User analysis: External interfaces

External interfaces (how will the product be used) are interactions between the product and the actors (including target users).

- An interface has a direction (in, out, or in-out)
- An interface is made of a flow (matter, energy, or signal)

  Example: XYZ Cargo ADD-ONS

  Identify the interactions between food producer and the product, **specify needs and uses**: - out: mechanical countainment - out: warmer and cooler - out: thermal energy

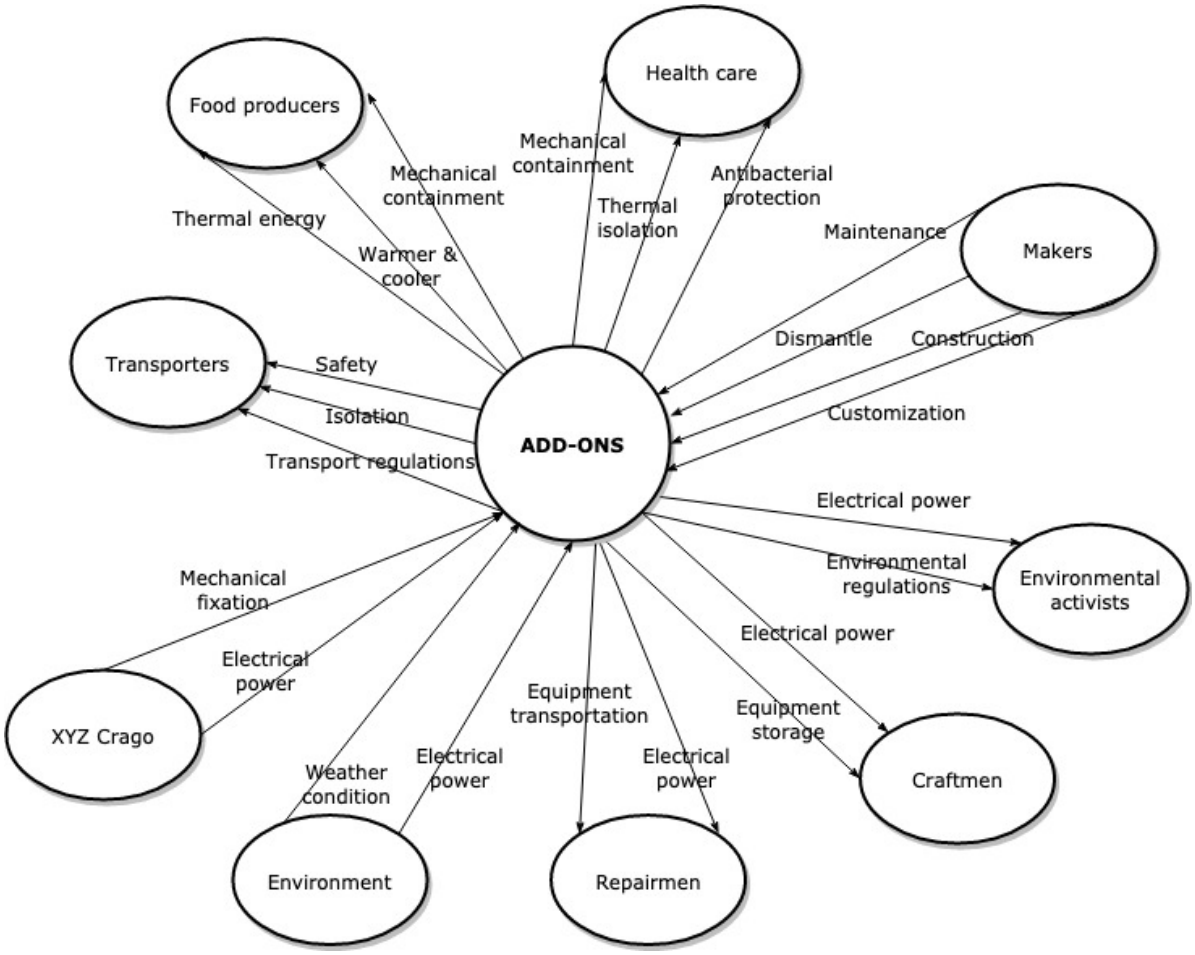


Figure 7.1: Image of External interfaces of XYZ cargo-ADD ONS

Sources

Section 12.1

7.11 User analysis - Skills needed to use

What is the specific knowledge the target users shall own to reuse - without modification - your product ??

An estimate of the time needed to acquire this knowledge, and link to education material may be useful here.

For example:

> The project echopen need basic knowledge about the medical ultrasound technology such as ultrasound imaging, a matter of acoustical impedance, etc.

Sources

Section 12.1

8 FBS modeling

8.1 FBS modeling Overview

Functional-behabioral-strucutral modeling (FSB) is a way to deconstruct the hardware design work. It starts with a functional model (what the hardware should be doing), then goes to a behavioral model (what behavior should the hardware have, this part is often not expressed by engineers) and finally the structural model is derived from the behavioral model. The different models are nurturing each other.

For example for a fridge, the function will be “keep the temperature at 14 degrees”, the behavior is “cool when the temperature is above 14 degrees”, the structure will be “have a temperature sensor and a electronic device which switch cooling on when the temperature is above 14 degrees” (one could think of different structures which would have the same or similar behavior, depending on the type of structure, one may refine the function to “keep the temperature between 12 and 14 degrees”,…)

8.2 How to document models

The documentation of technical functions, which requires adopting an internal (white box) viewpoint on the product, consists in breaking down the service function into sub-functions. The decomposition process is no more solution neutral as it requires to make a decision at every indenture level. The functional decomposition requires two modelling approaches: function tree and functional graph.

Provide both an image and the editable files, as well as instruction how to use them.

This approach can be taken for functional Section 8.3 and the different types of structural models Section 8.5.

Functional tree

The functional tree is a top-down decomposition of function into sub-functions that helps to simplify the problem to solve.

The decomposition of technical functions creates a functional tree and, the technical functions are defined based on the functional requirements.

- A top-down and bottom-up reading of the functional tree provides insight on the "how" and "why", respectively.
- The decomposition process should be stopped when the technical function is sufficiently detailed to reuse, make, or buy a design solution.

Tools/software:

- Use functional modeling language for representation, such as
  - UML (Use Case diagram)
  - SysML (Block Definition, Activity, or Internal Block diagram)
  - SADT/IDEF0
  - Functional flow block diagram
- Use open-source software for modeling the tree representation, such as:
  - draw.io (diagram.net)
  - excalidraw
  - Papyrus
  - Modelio
  - Capella

Functional graph

The functional graph is a multi-level logical articulation of technical functions. - Relationships between functions are in/out-going flows of matter, energy, or information. - Logical AND/OR gates can be used to define concurrent or sequential functions. - Articulation of technical function can describe as input-output relationships transforming flows by using the functional modeling language in the format of the graph

Tools/software:

- Use functional modeling language for representation, such as
  - draw.io (diagram.net)
  - excalidraw
  - UML (Use Case diagram)
  - SysML (Block Definition, Activity, or Internal Block diagram)
  - SADT/IDEF0
  - Functional flow block diagram

- Use open-source software for modeling the tree representation, such as
  - Papyrus
  - Modelio
  - Capella

Sources

Section 12.1

8.3 Functional model

A Functional model explains what the product is made for. It is:

- A description of the functions performed by a product.
- An opportunity to break down a product into smaller pieces (modules) that can be more easily understood.
- At the highest level of a functional breakdown (black box view), service functions are the effects (intended by its ecosystem actors) of the interaction of the product with its environment. (See actors analysis, Section 7.8).
- At the intermediate and lowest levels of a functional breakdown (white box view), technical functions are input-output relationships transforming matter, energy or information flows. They are expressing in a non-solution neutral way and observable from inside the product. A set of technical functions is necessary for the realization of a service function (in contrast to solution neutral expression of the capabilities, Section 7.3).

Why should you define functional model?

- A functional model helps to break down a complicated problem into simple sub-problems.
- A functional model helps to anticipate failures occurring when an intended effect of the product is no longer produced on its environment.
- A function is the main input to derive the functional requirements required to define the conditions of use of the product as well as to provide objective evidences through the validation and verification activities.

How to document a functional model?

The documentation of technical functions, which requires adopting an internal (white box) viewpoint on the product, consists in breaking down the service function into sub-functions. The decomposition process is no more solution neutral as it requires to make a decision at every indenture level. The functional decomposition requires two modelling approaches: function tree and functional graph, see Section 8.2.

Example: funtional tree

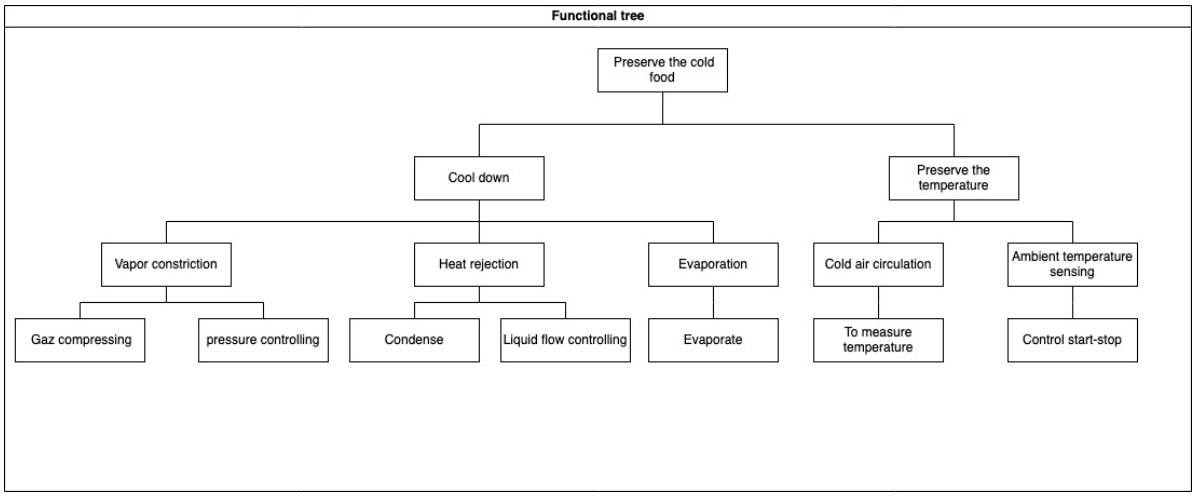
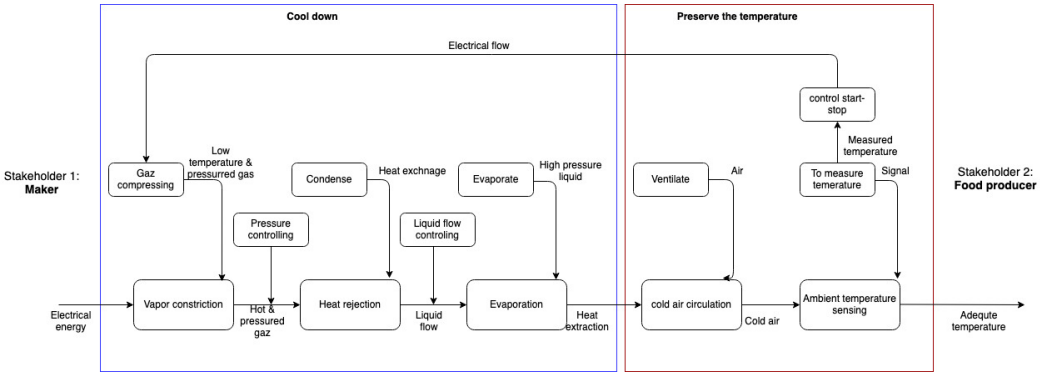


Figure 8.1: Image of functional tree of XYZ cargo-ADD ONS

Example functional graph



The image shows the functional graph of the relationship between technical functions for maintaining food quality by ADD-ONS of XYZ cargo.

Sources

Section 12.1

8.4 Behavioral model

The model will enable the makers to understand the analysis of the physical behavior of a product, this analysis supports the decision made at the later stages of design. This analysis is most often done using simulation software, or is made unconsciously in the designer head. Having some explicit model (even when no software is used) can be very useful to share ideas between designers.

The behavior model:

- describe the behavior of a product when it receives a stimulus.
- could be the mathematical description of the physical product, this description may be made via a modelling software (Simulation model) that should be included in the documentation.
- is the physical interactions between the components of a design, as well as between the design and its environment. An artifact exhibits certain behaviors not only by the change or maintaining of its physical state, but also by several interactions that take place inside the artifact, as well as with its environment.

Why should you define behavioral model?

- The behavioral model identifies the properties for understanding the calculation, simulation, and environment of the product.
- The behavioral model could provide the simulation of any given physical phenomenon using numerical techniques.
- Behavior model describes how the artifact implements its function and is managed by engineering principles and physical rules that are included in a behavioral model.

How to document a behavioral model?

Documentation should indicate the type of model, variables used to define the model, software used for the simulation, and results of the simulations.

Examples

- type of model:
  - mechanical simulation (finite element analysis (FEA) and computational fluid dynamics (CFD) are two types of mechanical simulation)
  - physical simulation
  - Thermo-mechanical simulation
  - Electronical simulations
- variables used in the model:
  - Specification of the geometrical model (refer to editable file format in the structural model)
  - Material characteristics (refer to structural model)
  - Initial conditions such as initial stresses, temperatures, velocities.
  - Boundary conditions can be imposed on individual solution variables such as displacements or rotations.
  - Kinematic constraints that are several of the fundamental solution variables in the model (Linear constraint equations) or multi-point constraints (General multi-point constraints) can be defined.
  - Interactions that are contact and other interactions between parts can be defined.
- (open-source) Software :
  - Open Modelica
  - ADINA

  Examples

  Example 1: FinEtools: Finite Element tools

  Example 2: Image below shows the simulation of the torsion of the fixed part from below and its evaluation of the reality.

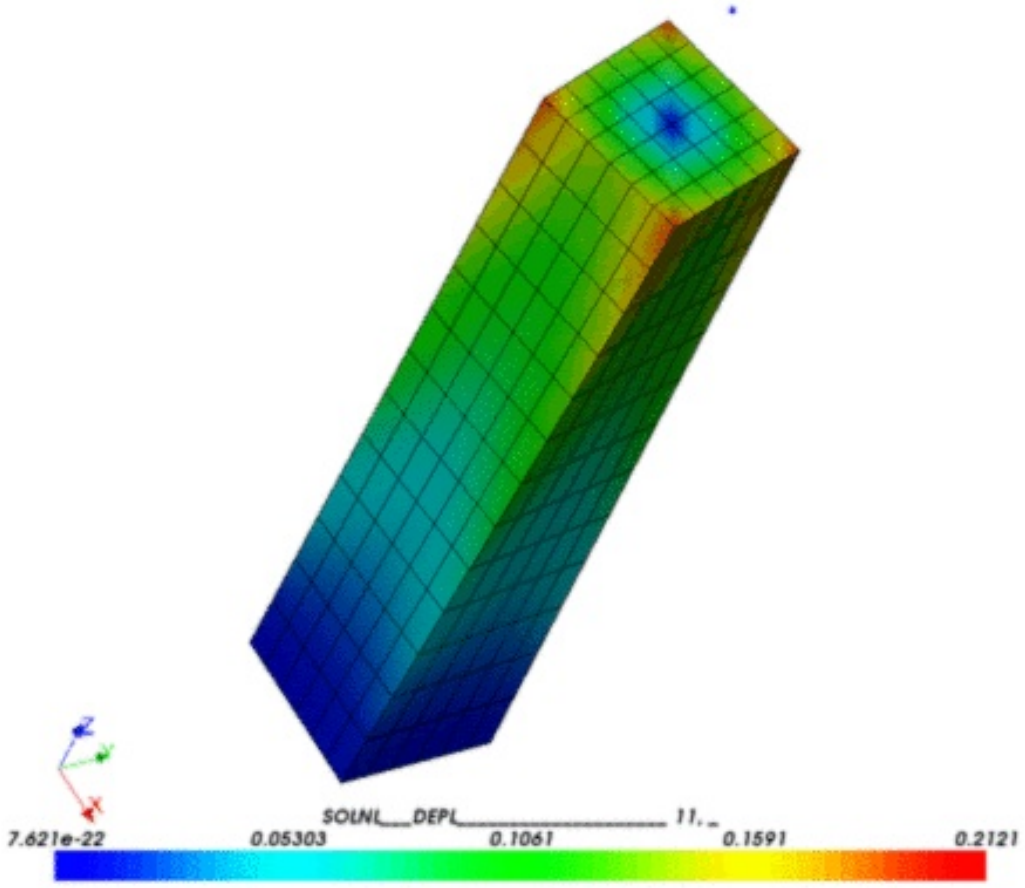


Figure 8.2: Image of Finite element analysis

Sources

Section 12.1

8.5 Structural model

This section will give an overview of the structural model, while details are given in the specific mechanical (also called architectural, Section 8.6), electrical (Section 8.8) and software

(Section 8.7) architectures sections. Note that sometimes, these different models are presented together in a single graphics.

The Structural model explains the physical structure of the product and its components, it is:

- A description of the components (the combination of parts) of a product and their relationships.
- An opportunity to specify the geometric elements, dimensions, topology, and other physical properties of the product.
- The potential solutions (concepts), which are the result of the conceptual design phase.
- The set of mechanics theories that obey physical laws required to study and predict the behavior of structures.

Why should you define structural model?

- A structural model helps to describe the geometric elements (design feature, dimensions, constraints, etc.), topology (assembly constraint between components, tolerances, components mating conditions, etc.), and characteristics of the product.
- A structural model helps to decide the physical form of the product and its components to ensure that the structure is fit for its intended purpose.
- Structural model provides users with a physical model of the product, components, and characteristics of the material at the design phase that enable the stakeholder to understand the geometry, material reaction to external factors, etc.
- The structural model ensures that the structures are safe and fulfill the functions for which they were built.

How should you define structural model?

The first level of definition is often to show how things are related together in a tree or graph, using "modeling language", such as "SysML (Block Definition, Activity, or Internal Block diagram)" or "UML". See Section 8.2 for more information.

Sources

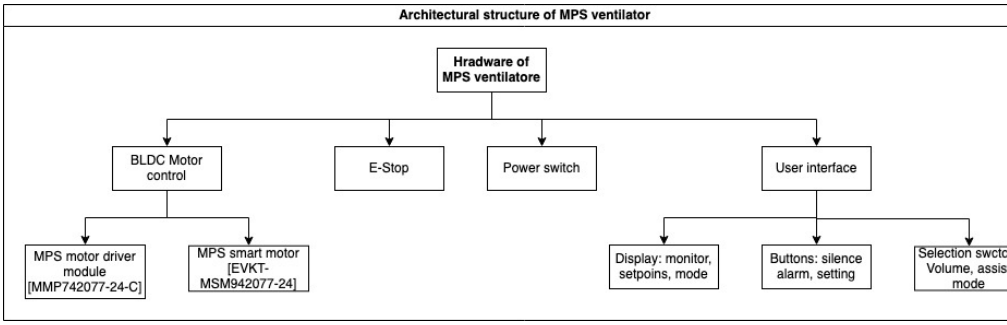
Section 12.1

8.6 Mechanical architecture

The architectural structure is a physical or logical layout of the components of a system design and their internal and external connections.

- A model specifying the kind of components and their sub-components in the format of a tree or a graph

See for information about how to document this. Provide both an image and the editable files, as well as instruction how to use them.



Examples MPS ventilator: {width=400}

Sources

OpenNext work project resutls: Section 12.1

8.7 Software and Firmware architecture

The software architecture represents the repository details of all the software and firmware that is necessary for reusing and running the project. Indicate how the different software depend on each other.

It might be interesting to describe the data flow and its format.

See Section 8.2 for information about how to document this. Provide both an image and the editable files, as well as instruction how to use them.

Software and Firmware details

Please provide :

- Clear installation guide
- Description of programming algorithm
- The source code
- Version of software and its dependencies (both hardware and software dependencies)

Documentation of different parts of software

Examples:

Nasa-JPL*

AmboVent*

Poppy project*

Sources

Section 12.1

8.8 Electrical architecture

The architectural structure is a physical or logical layout of the components of a system design and their internal and external connections.

1. What minimum documentation should the architectural structure provide?
    - A model specifying the kind of components and their sub-components in the format of a tree or a graph including
        - DC motor
        - A/D converter
        - DC converters
        - Rotor
        - Sensor system
        - Motherboard
        - kit
        - Resistor

- Transistors
- IC
- Sensors
- Etc.

See Section 8.2 for information about how to document this. Provide both an image and the editable files, as well as instruction how to use them.

example

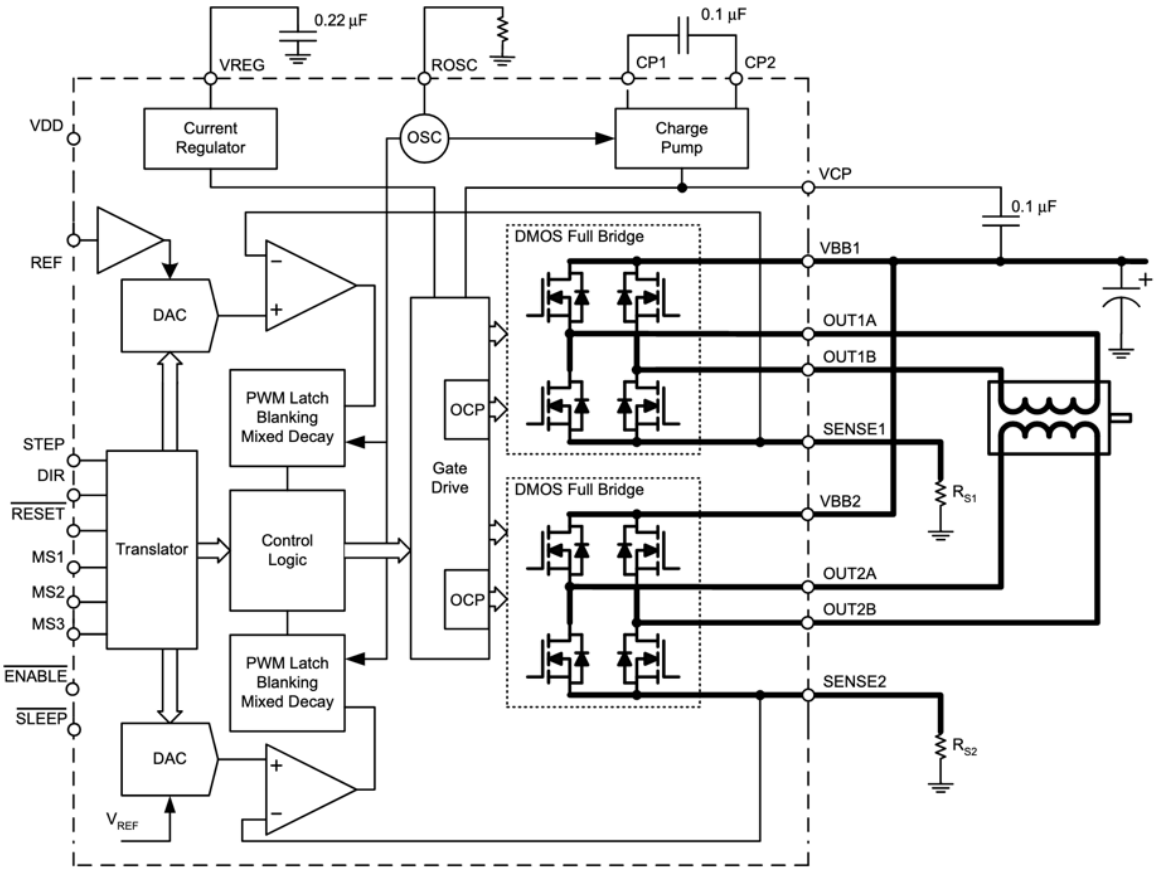


Figure 8.3: Image of Structural graph of Open-Source-Ventilator

---

Sources

Section 12.1

8.9 Product design and CAD files

Computer assisted design is usually used to create structures from the models. The designs should be best provided:

- in their raw format (*Native formats*, best for making changes),
- an editable standard format (*Neutral formats*, interchange and export formats that can be opened or imported by other CAD programs, they usually have large restriction in what one can edit in comparison with source format),
- AND a exported format (*Auxiliary format*, ready-to-view outputs, may be used for production or for study the design) used for manufacture.

For example, a fusion .f3d file in the native format, a Step file is a neutrla format and a STL file in an auxiliary format.

Always indicate what software (including version number) was used to create the native format file (see Section 5.12).

CAD files may contain different parts/modules in a single file. It is good practices to export an assemble view and an exploded view of the hardware (if the software does this). These views may be included in the BOM overview and the assembly instructions.

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. Also try to name files with a clear name, add material specification and number of pieces, the name of the file should correspond to the name of the part in the BOM (Section 9.7).

Brief advice for object design:

- Design the project to be easily adaptable for common differences (for example metric vs imperial units) and adaptations.
- Make the design parametric (for example make product length and width dependent on one single parametric unit).
- Utilize asymmetric design to make parts fit together uniquely and avoid miss-assemblies.
- Use automation tools of the used programs
- 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.

CAD for electronics, PCB design

To reuse a electrical design, it should provide information consist of:

1. Preferable file format

- Editable file formats that could be:
  - Source formats such as .gbr, .lib format
  - Neutral formats such as Kicad_mod, kicad_pcb

2. Preferable open-source software

- Tiny CAD
- KiCAD
- ADINA

Examples:

Nasa-JPL, neutral files, eschange format*

AmbovVent, Neutral files*

CAD for structure models

Model design should be provided in a format that can be used to manufacture each piece: usually STL and STEP file for 3D objects, and SVG file for 2D design, electrical design can be shared in the editable format directly. These formats are often easier to preview, and will help understand the structural/electrical design.

Note that the file is often not sufficient to reproduce one piece and manufacturing information must be given, as explained in Section 9.12.

Editable file format

An editable file format is a standard way that information is encoded for storage and allows the makers to study, modify the geometry of a model and reuse it.

To reuse a design model, it should provide information consist of:

1. Preferable 3D/2D file format

- Editable file formats that could be:
  - Native formats such as .FCStd format of FreeCAD

   - Neutral formats such as STEP files

2. Preferable open-source software

- OpensCAD
- Inkscape
- FreeCAD

Example of editable file formats:

- Farmbot, Native CAD files*
- MIT Emergency Ventilator, Neutral CAD files*
- Types of CAD format of transmagic*

Sources

Section 12.1, Section 12.6, Section 12.9, Section 12.10

9 Hardware production

9.1 Generalities

Production instructions mean full description and instructions concerning raw material, operating conditions, and process to be employed for the manufacture and assembly of the product. This includes also skills and tools needed for manufacture and assembly.

Technical documents provide the source needed to study and replicate a hardware design. In contrast to project documentation and community building, technical documents for OSH are quite specific, but can be considered analogous to what source code is for software. Depending on the project, technical documents may include technical drawings, images describing electronic schematics, computer-aided design (CAD) files, or assembly instructions to replicate the design. A thoroughly documented project will have all types of documents. It may also include code (firmware and software) necessary to run the hardware. The source files (like CAD files) are best accompanied by textual and multi-medial documentation, such as guides for manufacturing, assembly, maintenance, and development.

Production usually starts with the sourcing of material, and a bill of material (BOM) is required. It describes all the components and their references. If the component is to be purchased, one should find all the information required to buy the part. If the part is to be manufactured, one should find all the descriptions of the manufacturing instructions, as well as all the components needed for this manufacture (for instance, The BOM should report the amount of PLA needed when parts are 3D-printed).

9.2 Relation to structural modeling

Technical documents include both the structural modeling and the production instructions, they provide the source needed to study and replicate a hardware design. In contrast to project documentation and community building, technical documents for OSH are quite specific, but can be considered analogous to what source code is for software. Depending on the project, technical documents may include technical drawings, images describing electronic schematics, computer-aided design (CAD) files, or assembly instructions to replicate the design. A thoroughly documented project will have all types of documents. It may also include code (firmware and software) necessary to run the hardware. The source files (like CAD files) are best

accompanied by textual and multi-medial documentation, such as guides for manufacturing, assembly, maintenance, and development.

Following the guide for structural modeling, one should share both raw and derived source files. For instance, 3D object designs should be shared as print-ready files (.stl file for instance), but also as modifiable 3D objects (the format of these files will depend on the software used). It is necessary to provide the raw files to enable modification, even if they can often only be opened in proprietary software, and their use requires specific skills. The derived versions are used to build the hardware, but often are not suited for modification. Users with access to the tools that can read and manipulate these raw source files can update and improve the physical device. If they wish, they can proceed to share such modifications.

9.3 Production instructions

Production instructions should include:

- A bill of material (BOM): it gives an overview of all the material that needs to be sourced and/or manufactured, and describes all the components and their references: Section 9.7
- Manufacturing information: they can guide the makers to follow a process for replicating a product, and mean full description and instructions concerning raw material, operating conditions, and process to be employed for the manufacture of the hardware parts.
  - Manufacturing safety instruction, skills needed and tools: Section 9.11
  - Manufacturing sequences: Section 9.12
- Assembly instructions: they can guide the makers to follow the process of assembly or disassembly of components of a product, and illustrate visually and with words and text how to assemble or disassemble the mechanical and electrical components of the product.
  - Assembly tools and skills needed: Section 9.13
  - Assembly sequences: Section 9.14
  - If relevant, the electric and electronic plan should be provided (Section 9.9). Note that firmware and software installation (described at Section 9.10) may be included in the assembly sequence.

9.4 Modularisation

When there are multiple parts, it is best practice to split the project into different modules and have a BOM for each part, and one BOM for the whole hardware (see Section 9.7). For complex project, it is therefore best to create BOM automatically. Some software are meant to created

BOM from the CAD files (CAD-coupled documentation, https://doi.org/10.5334/joh.56) or from the assembly instruction (using Gitbuilding).

These questions may help you define what a module is: What parts of the hardware could be used as a component of another project or a variation of this hardware? What solutions do you find most noteworthy given your experiences during the design iterations? What separate functions or actions does the hardware perform?

9.5 Linking products and documentation

- Provide links to the source (original design files) for your hardware on the product itself, its packaging, and/or its documentation. Add a version number on the hardware (or a release date) so that people can match the physical object with the corresponding version of its documentation.
- If linking to a website, make it easy to find the source (original design files) from the website for a product.

  Example

  Example 1: JPL Open Source Rover

Manufacturer Part Number	Manufacturer	Digi-Key Part Number	Customer Reference	Reference Designator	Packaging	Part Status	Quantity	Unit Price	Extended Price
3169	Adafruit Industries LLC	1528-1737-ND			Bulk	Active	2	4.95	$9.90
3166	Adafruit Industries LLC	1528-1734-ND			Bulk	Active	2	4.95	$9.90
3167	Adafruit Industries LLC	1528-1735-ND			Bulk	Active	2	4.95	$9.90
3168	Adafruit Industries LLC	1528-1736-ND			Bulk	Active	2	4.95	$9.90
LM358P	Texas Instruments	296-1395-5-ND			Tube	Active	10	0.367	$3.67
76512.18.01	General Cable/Carol Brand	C76512B-50-ND			Bulk	Active	2	15.59	$31.18
1569-20-1-0500-001-1-TS	CNC Tech	CN538B-50-ND			Spool	Active	2	11.92	$23.84
76512.18.03	General Cable/Carol Brand	C76512R-50-ND			Bulk	Active	2	15.59	$31.18
1569-20-1-0500-004-1-TS	CNC Tech	CN541R-50-ND			Spool	Active	2	11.92	$23.84
PRPC040SAAN-RC	Sullins Connector Solutions	S1011EC-40-ND			Bag	Active	10	0.592	$5.92
C3900BA	Bulgin	1091-1026-ND			Bulk	Active	1	4.49	$4.49
3165	Adafruit Industries LLC	1528-1733-ND			Bulk	Active	2	4.95	$9.90
3164	Adafruit Industries LLC	1528-1732-ND			Bulk	Active	2	4.95	$9.90
CF14JT4K70	Stackpole Electronics Inc.	CF14JT4K70CT-ND			Cut Tape (CT)	Active	10	0.04	$0.40
CF14JT10K0	Stackpole Electronics Inc.	CF14JT10K0CT-ND			Cut Tape (CT)	Active	25	0.0288	$0.72
CF14JT22K0	Stackpole Electronics Inc.	CF14JT22K0CT-ND			Cut Tape (CT)	Active	25	0.0288	$0.72

Figure 9.1: JPL BOM

Example 2: SatNOGS Rotator v3

Example 3: Krab v1.0

9.6 Helping workflow and software

It is sometimes easier to create a guide for manufacturing and assembly. For instance, using the Gitbuilding software, one can write the manufacture and assembly instructions, and when using specific tags for tools and material, the software creates BOM, part lists and tool lists for each step and for the whole project.

Sources

Section 12.1, Section 12.5, Section 12.6, Section 12.10

9.7 Product Build: Bill of material

A bill of materials (BOM) is a comprehensive list of parts, items, and other materials required to create and assemble a product, as well as instructions required for gathering and using the required materials.

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.) 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.

When the hardware is made of different modules or parts, the bill of material usually reflects this modularisation, one can have different BOMs for each module (as is produced with Gitbuilding), or have the part numbering reflects the modules (for instance with all materials of module one being numbered 1.x). One can also provide assembled and "explosed" 3D views of the hardware. See the Section 9.4 chapter.

The BOM list should be computer readable, preferably in CSV format, with UTF8 encoding. Follow best practices in spreadsheet design, gives parts in rows, give one and only one column for each type of information, one cell containing only one value. Never merge cells and do not use colors in your software (they will disappear in the csv version). Note that it is easy to automatically merge to cells into one, but difficult to split one cells into two.

If the component is to be purchased, one should find all the information required to buy the part. There are two types of purchased components. 1. the standard components (such as screws, connectors, standardized electrical components) can usually be bought at different

places and have a specific persistent identifier (see Section 9.7). 2. the non standard components need to be described in more details indicating at least technical data, supplier, order number, and website.

If the part is to be manufactured, one should find all the descriptions of the manufacturing instructions (see Section 9.12). There are also two categories here: parts that were designed in this project, and parts that were designed by others and were made available under a permissive license.

A bill of material usually includes:

1. Part number
2. Part name
3. Quantity (amount and unit)
4. Part type: `Production` (needs to be manufactured, self-designed part), `Standard`, or `Buy` (proprietary components)
5. Specifications, with 5 columns:

- Manufacturer
- Manufacturer part number
- Supplier
- Supplier part number
- Link to manufacturing instructions (step by step instructions)

One may also add:

6. Part short description
7. Manufacturing process/tool (3D prining, machining, laser cutting)
8. URL, link to an internet source of the part
9. Price
10. Manufacturing standard lead time
11. Packaging instruction
12. Pre-processing needed
13. Notes
14. Link to the documentation:

    - Part description
    - images
    - videos
    - source: CAD files
    - export files (STP, STL, PDF file)
    - auxiliary (files that are neither source files nor their exports, but still useful)

NB: Digi-Key part number is a special case of supplier part number

NB: The OSH-LOSH metadata schema has special entries for parts information, including tsdc (Technology-specific Documentation Criteria), which is a computer readable version of the information asked in the list above.

Standard components

For standard components such as fasteners, pipe fittings, among others, search in the following order in order to find persistent identifiers (for instance DIN designation) and suppliers:

- DIN libraries of the CAD systems used
- Other libraries of the CAD systems used
- Links provided by OHO
- Internet libraries, mainly in metric system
- Wikipedia

Sources

Section 12.1, Section 12.6, Section 12.9, Section 12.8, Section 12.10

9.8 Product build - material characteristics

Definition: The characteristics of the materials are those that identify the reactions of materials reactions to heat, electricity, light, force, etc.

Summarise what materials have been used to construct the hardware and what methods to process the materials (as well as the assembly if relevant). Provide more details or references where important materials or methods are non-standard, not globally available, or produced only by one manufacturer.

Give information on the selection of materials, based on factors including properties for behavioral analysis (Section 8.4), environmental impact (Section 11.6), and manufacturing processes in design reuse. (Section 9.12)

The material characteristics of mechanical parts consist of:

1- Identifying the kind of characteristics and their properties:

- Mechanical characteristics like hardness, elasticity, plasticity, toughness, etc.
- Manufacturing properties like castability, machinability rating, etc.
- Thermal characteristics like melting point, thermal conductivity, etc.
- Electrical characteristics like electrical resistivity and conductibility, etc.
- Chemical properties like corrosion resistance, surface tension, etc.

Example of material characteristics:

Figure below shows some physical properties of superalloy base elements.

		Melting Point		Density		Expansion Coefficient[a]		Thermal Conductivity[a]	
	Crystal Structure	°F	°C	lb/in³	g/cm³	°F × 10⁻⁶	°C × 10⁻⁶	Btu/ft²/hr/°F/in.	cal/cm²/s/°C/cm
Co	HCP	2723	1493	0.32	8.9	7.0	12.4	464	0.215
Ni	FCC	2647	1452	0.32	8.9	7.4	13.3	610	0.165
Fe	BCC	2798	1535	0.28	7.87	6.7	11.7	493	0.175

[a]At room temperature.
Source: From *Superalloys II*, Wiley, 1987, p. 14.

Figure 9.2: Image of material characteristics

Source: Kutz, M. ed., 2002. Handbook of materials selection. John Wiley & Sons.

Documentation of material characteristics

- Name of characteristic
  - Properties
  - Unit of property
  - ...

Sources

Section 12.1, Section 12.3

9.9 Product Build: Electrical design

Datasheet of components for electronic parts:

- Description of features
  - Core
  - Memories
  - Advanced connectivity

- Device summary
  - Reference
  - Part number
- How to use the parts?

Sources

Section 12.1

9.10 Product build - Firmware/Software

Here comes the elements that were more briefly described in Section 8.7. Since software/firmware development follows different practices that often needs a more detailed version control system, they are usually developed independently of the hardware.

Any code or firmware required to operate the hardware should be shared. 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 (for example, third-party libraries or tools). In addition, it is helpful to provide an overview of the state of the software (for example, "stable" or "beta" or "barely-working hack").

In all cases, it is important to keep track of which version of the soft/firmwares are used with which version of the hardware.

Also indicate details on the operating software and programming language, and include minimum version compatibility, and dditional system requirements, like memory, disk space, processor, input or output devices.

> Example: the airtrack hardware: https://codeberg.org/openmake/airtrack-hardware, software: https://github.com/open-make/code-airtrack
>
> The Airtrack hardware was developed using pixycam. The hardware and the software are developed in different repositories with different people involved. In 2025, as the pixycam was not produced anymore, a new version of the hardware was created, using the pixycam2 component. This had little effects on the hardware design, but, the firmware and software needed to be modified.

Sources

Section 12.1, Section 12.3, Section 12.10

9.11 Product Build: Manufacturing safety skills, and tools

This part explain how to manufacture parts, this may be directed toward professional manufacturers or hobbyist, depending on the skills needed.

Manufacturing safety

Are there specific risks and safety measure to take during the manufacturing steps

Manufacturing skills

What is the specific knowledge or skills a maker shall master to manufacture the different parts of the hardware ?

Sources

none

Product build: Manufacturing tool

Manufacturing tools means all the machinery, equipment, and processes used to manufacture products. Manufacturing technology guide to find the type of necessary technology to produce the part. In that case, it should describe the most suitable technology according to the context.

Type of machines

Type of machines used

1. CNC machine tools for machining metal or other rigid materials
    - Milling
    - Lathe
    - Cutting
    - Drilling
2. Other common manufacturing tools

- 3D printing (FDM, SLS…)
- Thermoforming
- Burning machining technology (laser cutting, Plasma cutting, …)
- Bonding technologies (Solder, cold welding, arc welding, adhesive bonding …)

3. Finishing: to achieve the right properties such as surface quality, geometrical accuracy, and mechanical properties, finishing is essential.

    - Sanding after 3D printing
    - Gap filling
    - Blasting
    - Polishing
    - Priming and painting

    Examples

    JPL Open Source Rover

    SatNOGS Rotator v3

Sources

Section 12.1

9.12 Product build: Manufacturing sequence

The Manufacturing sequences refer to step-by-step machining and manufacturing processes in a target-oriented arrangement to enable manufacturing.

- `The machining sequence should define for the manufacturing of each part.`
- `Process parameters are all those parameters that are inherent to any machining operat`
- `Manufacturing standard file formats support some of the manufacturing processes and t`

What does include the documentation of manufacturing sequences and instructions?

1. Name of the related machine of each step
2. Describing step by step sequence of the machining process - Machine - Type of operation - Tools description - Process parameters of each machining operation

    - Process parameters of 3D printing

- Process parameters of Laser cutting
- Process parameters of CNC machines such as Lathe, Milling, etc.
- Process parameters of arc welding
  - Raw material (including size if relevant)
  - Manufacturing files (STL, SVG or G-code, …)
- CAD files in an interchange format such as STL that is suitable for 3D printing
- Nominal geometry and its allowable variation by using symbolic language on 2D drawings like SVG, JPEG, and PDF format that is suitable for laser cutting
- Manufacturing export formats such as G-code, STEP-NC is suitable for CNC machining
- Circuit board design formats such as Gerber RS-274X, excellon that is suitable for vector photoplotters 2D mechanical NC machines

Examples

Machine	Type of operations	Tools	Parameters	Unit	Value	Raw material	File	Image
3D printer	Print	Nozzle guide	First layer height	mm	2	PLA	link to STL file	
	Print	Nozzle guide	First layer speed	mm/s	3	PLA	link to STL file	
CNC machine	Drilling	Drill	Depth of cut	mm	8	Steel	CAD file	
	Milling	HSK milling cutter	Cutting speed	mm/s	4	Steel	CAD file	
Machine 3								

Figure 9.3: image of manufacturing sequence

JPL Open Source Rover

DIY Dremel CNC design and parts and its CAM file for machining

SatNOGS Rotator v3, 2D drawing file

Note: types of CAD format of transmagic

Example of parameters

1. 3D printer parameters

- Extruder setting
  - Extrusion multiplier
  - Retraction distance
  - Retraction speed
  - Coasting
- Layer setting
  - First layer height
  - First layer speed
- Laver height
- Printing bed temperature
- Infill setting
  - Internal/Eternal fill pattern
- Temperature setting
- Cooling setting

2. CNC machines parameters such as Lathe, Milling, etc.

- Cutting parameters
  - Cutting speed
  - Feed rate
  - Cutting depth
  - Cutting width
  - Cutting force
  - Spindle speed
  - Cutting temperature
- Cutting tool
  - Tool Geometry
  - Tool setting
- Coolant

3. Burning machining parameters such as laser cutting

- Beam parameters
  - Wavelength

  - Power and intensity
  - Polarization

- Process Parameters
  - Focusing of laser beams (the focal length of the lens)
  - Focal position
  - Angle of incidence
  - Cutting speed
  - Gas pressure
  - Stand-off distance
  - Expected duration

4. Bonding technologies parameters such as Arc welding

- Welding current
- Welding voltage
- Arc travel speed
- Torch angle
  - Longitudinal
  - Transverse
- Electrode force
- Electrode diameters
- Length of arc

Sources

Section 12.1

9.13 Product Build: Assembly safety, skills and tools

This document should provide information about the specific knowledge a maker shall own to assembly the hardware product, and what tools are necessary. For example, one can report how many people are needed to assemble the hardware.

Example of skills and machines:

1. Required skills for assembly
    - Operate drilling machine
    - Operate Band Saw/Dremel
2. List of the tools for assembly or disassembly
    - Mandatory
        - Allen Keyset
        - Imperial wrench set
    - Optional
        - Drill press

The skills can be listed by name and a description. In many case, it might be interesting to link skills with tools, as being able to operate each tool is a needed skill.

> Example
>
> To build the Airtrack, it is optional to use specific UV glue and its specific equipment. One should nevertheless have some experience in using plastic glue.

Assembly Safety instruction

Assembly instruction may include specific notes on safety.

> Example
>
> When the laser cutter machine is open, it becomes a class 4 laser. This means that when the machine is open and under electric current, you have to wear glasses and be in a closed room labeled as dangerous ("laser in function").

Sources

Section 12.1

9.14 Product Build: Assembly sequence

To help others make and modify your hardware design, you should provide instructions for going from your design files to the working physical hardware. It is good to publish annotated photographs (or video) from multiple viewpoints and at various stages of assembly. If you do not have photos, posting annotated 3D renderings of your design is a good alternative. Either way, it is good to provide captions or text that explain what is shown in each image and why it is useful.

The Assembly sequence usually start with a description and list of each part that will be assembled, and then provide a step-by-step guide. One can think of instruction for lego objects (In these special case, the part list is identical to a BOM and placed at the end).

See Chapter 9 for information about software and dependence with guide for manufacturing.

Part list

The Part list for mechanical parts is a complete list of all parts needed to build the complete product. It is different from the BOM which list material needed for the manufacture of the parts, while this document list the manufactured parts.

It constitutes of : - Item numbers: are based on the assembly structure, that is, the order in which parts are displayed in assembly. - Part number or drawing number: which is a reference back to the detail drawing (refer to the BOM). - Description: is usually a part name or a complete description of parts. - Quantity is the number of that particular part used on this assembly. - Image (or STL render) of each part.

Sequence

The set of steps necessary to properly assemble the parts should be well described at each step.

- The joining technology at each step should be clearly described: - Screwing - Bolting - Soldering - Gluing (or “gluing and screwing”)

  Example

  Poppy Robot

  JPL Open Source Rover

  SatNOGS Rotator v3 , Assembly instructions

  Open Source Powered Prosthetic Leg

Notes

It is good practice to design the parts such that the assembly is easier. One can for instance include the item number on the parts and make sure that it is difficult to assemble parts at the wrong step, for instance by designing asymetrical parts.

You may indicate any measures that have been taken in the design to make the hardware easy to build for other users (reduction of parts, features in the design to make the hardware assembly more reliable, …)

Sources

Section 12.1, Section 12.3, Section 12.5

9.15 Product Build: Disassembly instruction

Please refer to Section 9.14.

Disassembly may be different from assembly, please give information about skills, tools, safety issue and sequence instruction for dismantling the hardware. Also indicate if there are differences between disassembly for repair, or disassembly for disposal.

Sources

10 Project history

10.1 Changelog

A changelog is a plain text file that contains a record of what *notable* changes are made between versions of hardware. It includes new features, enhancements, bug fixes, and even minor tweaks. The **keep a changelog** website provides a detailed explanation of what a change log is (in the software case).

Changelogs are primarily intended for technical contributors and power users who need a detailed understanding of how the hardware has evolved over time.

Guiding Principles

- Changelogs are for humans, not machines.
- There should be an entry for every single version.
- The same types of changes should be grouped.
- Versions and sections should be linkable.
- The latest version comes first.
- The release date of each version is displayed.
- Mention whether you follow Semantic Versioning.
- Types of changes:
  - Added for new features.
  - Changed for changes in existing functionality.
  - Deprecated for soon-to-be removed features.
  - Removed for now removed features.
  - Fixed for any bug fixes.
  - Security in case of vulnerabilities.

Sources

Section 12.5, https://keepachangelog.com/en/1.1.0/

10.2 Release notes

Release notes detail the corrections, changes or enhancements (functional or non-functional) made to the service or product the company provides. They are are summaries of changelog made for target users and manufacturers. They usually come with new hardware release. Release notes can also contain test results and information about the test procedure. This kind of information gives readers of the release note more confidence in the fix/change done.

Sources

Wikipedia, see also https://blog.releasenotes.io/changelog-vs-release-notes/

10.3 Design choices

What were the decisions made in designing this hardware? Were other designs/options tried? please describe also what did not work.

This information may be included at different places in the documentation, it may be interesting to link them together, or find a way to make them discoverable in the documentation files, for instance using a specific text one can search for.

Sources

Section 12.3

11 Guide for Users

I would like to provide important information to end-users on 'how to use my product'.

11.1 What is the user guide?

The user guide consists of information that allows end-users to operate the product properly. This may includes information on setting the hardware, operating the hardware, recognizing and solving problems, run the maintenance of the hardware and dispose of the hardware when it is beyond repair. The user guide is written for non-technical people.

Note the user guide may link to other parts of the documentation (especially for installation instruction/first use instrutions), or even to specific online help (for troubleshooting, a forum may be indicated on to of "usual problems"). It usually includes:

1. Description of the device :Section 11.3
2. Safety information: Section 11.4
3. Use and calibration of the product: Section 11.5
4. Environmental management: Section 11.6
5. Troubleshooting and testing section: Section 11.7
6. Maintenance and Repair information: Section 11.8, Section 11.9
7. Disposal information: Section 11.10

11.2 How to create a user guide ?

The form of the guide is usually a printable document or a website. It is advised to use markdown file for its creation and deliver a printable pdf for the users. The use of media such as explanatory diagrams, screenshots, pictures, photos and videos are usually welcome. The instruction steps should be minute and explicit.

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.

Some popular software used to create webpage and pdf from markdown files:

- GitBook
- readthedoc
- quarto
- Gitbuilding can also be used to this purpose.

  Some examples of open-source projects that show the user guide.

  PSLab oscilloscope

  PX4 vision userguide

  Echopen project

  Poppy project

  FarmBot Genesis V1.5

Sources

Section 12.1, Section 12.6, Section 12.10

11.3 Overview of the hardware for users

This should provide a brief overview of the hardware and its functions, it may include:

- Device name (and its parts) and their definition
- Essentials and technical specifications

This may be a summary of the hardware overview provided for contributors (the file "03_specification_concept/01_hardware_overview" in following the template v1.0).

One can describe examples of application of the hardware. This should include some evidence of output, like data produced by the use of the device or a picture of other types of results. Outline how the quality control in Section 11.7 enables the use of the hardware in this context. We encourage the link to experiment results or the reference to a publication (published or to be published) where these results are detailed. We also encourage pointers to ongoing work.

Sources

Section 12.1, Section 12.3

11.4 User Guides : Safety information

Describe all relevant safety issues or references to a risk assessment if included (for example high voltage, chemical safety etc.). If appropriate, discuss the wider context of the use of the hardware and safety issues or risks that may arise in the use environment. t may include more than one link if different assessments cover different aspects or making, operation, maintenance and disposal of the hardware.

OSH makers are not always formally trained engineers and may not be able to easily differentiate between dangerous and safe manipulations.

This does not need to be an exhaustive list of all risks associated with the thing, but a summary of the most important risks and hazards associated with making, using, maintaining or disposing of the hardware. Knowledge of such issues could influence the decision to make the thing.

Sources

Section 12.3, Section 12.5, sec-source-OKH

11.5 Operation instructions

The operation instruction gives information on the use of the product, in particular in term of software. Note that it may include the calibration chapter (Section 11.5) when calibration is necessary for each use.

It may include:

- Materials required
  - App
  - Software
  - Firmware
- Procedure
  - Installation instructions including
    - Firmware installation

    - Software installation
    - App installation
- Setup instructions containing
    - Software setup
    - Firmware setup
    - App setup
- Explains how to update the firmware to the latest version

Sources

Section 12.1

User Guides: Calibration instructions

If the hardware is used for measurements, please detail here how the reliability of measurements, or other hardware properties that are relevant for measurements, has been quantified and explain the results. Be clear about the processes or procedures used to compare the hardware to a standard, as well as the description of the standard calibrated against. Detail the general procedures in place for users to calibrate their hardware before or during use. What methods can be used to relate user-generated data to data from other sources?

Sources

Section 12.3

11.6 User guides: Environmental management

This chapter deal with information about the use and storage of the hardware. It usually includes information about:

```
-   Protection against weather conditions
-   Determining the acceptable temperature range
```

Sources

Section 12.1

11.7 User Guides: Troubleshooting

Testing instructions

The user guide should give information on methods to test the accuracy of the hardware. This may include specific data to give the hardware as input and some information about how to interpret the quality of the output.

Details can be provided on the testing of hardware functionalities, that are not directly essential for the precision operation of the hardware in the given context (which are in turn, where applicable, handled under Calibration), such as automated movements to position the hardware, repeatability of tool exchanges, recyclability, water-tightness, weight or other possibly relevant characteristics.

The testing should define the safe/reliable limits in which the components can be operated (e.g. step size and repeatability of linear motion, force ranges, ratio of devices with leaks when built in a workshop, etc). This will enhance the usability of the hardware or method in other contexts.

> Example: Ink Printers usually have a testing mode where a specific data file is used where the printed output will inform the user about the accuracy of the printer. The guide then explain what users should look for in order to make a diagnostic.

Troubleshooting

- Instructions on how to solve common problems
- Instructions to get additional help and report problems (Git forge issues, forums, chat,…).

Sources

Section 12.1, Section 12.3

11.8 Maintenance

Maintenance instructions provide the necessary information to maintain the system effectively and perform the required operations to system works properly in the long run. This includes advice on the frequency of the maintenance and the risks of failure.

It may also include information on the process of modifying a system or component after delivery, in order to correct faults, improve performance or other attributes, or adapt to a changing environment.

Maintenance instructions

A maintenance instruction is a technical communication document intended to give recommendations and necessary information to maintain the system effectively. In this book, we treat the identification of defective components and their repair as separate tasks, while others have defined it as part of the maintenance.

Note that the instructions are meant for the users and should therefore be focused on the schedule of maintenance. The maintenance information (what to do) may be directed toward experienced/professional people. In addition, this latter (technical) information may be best linked to the technical documentation of each part, in order to avoid giving outdated information (for example giving repair information for version 1, while the rest of the documentation is for version 2 of a hardware).

The maintenance user guide may include:

1. Introduction of general maintenance
    - Cleaning
    - Lubricating
    - Regular inspections or services. These can be carried out on a time-based schedule or a usage-based schedule. See also Section 11.9:
        - Maintenance according to predetermined intervals
        - Maintenance according to prescribed criteria
        - Maintenance by integrating analysis, measurement, and periodic test activities
    - Regular adjusting machinery
    - Maintenance tools (Various tools necessary to perform the maintenance operation)
    - Schedule for Replacing equipment, see also Section 11.9

    examples FarmBot Genesis V1.5

Maintenance schedules

Common procedures:

- cleaning
- lubricating
- machine adjustments
- calibrations
- (periodic) test activities

Information to give:

- predetermined interval

- prescribed criteria
- tools
- verification procedure

Component lifespan

You may indicate the lifespan of the equipment parts, with links on how to replace parts. Give information on expected lifespan of parts, how use and maintenance may modify the lifespan, and when to test/replace the parts.

Sources

Section 12.1

11.9 User Guides : Repair

Identifying the defective components

This can be linked to scheduled tests, or be done when an issue is seen. The guide should answer these questions, it may be written as a table:

- How to monitor the performance of the equipment? (what additional equipment may be necessary for this monitoring)
- How to detect a defective component, and determine what is defective ?
- How to eliminate the fault (see below) ?
- How to verify that the fault was eliminated

Repairing the defective components

For each element which may be defective:

- Step-by-step procedures describing the repairing sequence
- Refer to the technical documentation where you can find the manufacturing (and assembly) instructions to rebuild the defective components
- Indicate the required tools for repairing
- Verification of repair
- It may be interesting to add images and photos.

It might be useful to give information on repairing versus replacing the parts.

Sources

Section 12.1

User Guides: Replacing equipment components

This is very similar to the repairing section Section 11.9, even if this is planned modifications.

- Links to step-by-step procedures describing the replacing sequence.
- Required tools for replacing the components

This may be linked to the dis-assembly instructions Section 9.15

> Example
>
> Replacing Ink cartridge in a printer or a lamp bulb in a video projector.

Sources

Section 12.1

11.10 Disposal instructions

Disposal instructions identify the process of removing a system or its component, ensuring the proper handling of any environmentally sensitive materials, and sending the remainder to surplus storage or sale.

They describe each component with information about their end of life (depending on their material content). Each element will be classed corresponding to their "recyclability", and information is given to reduce the negative environmental impact of the disposal and recycling of the hardware component.

Template of disposal instruction

For each element or part, indicate:

- sub-part name
- parent part number and name, with links
- link to dissassembly instructions (see Section 9.15)
- material used + how many times can it be recycled (see also Section 11.8)
- recyclable (yes/no/conditional, see Section 11.10)
- disposition information, for each option:
  - how to dispose
  - environmental assessment of disposal options (@sec-environmental-assessment):
    - described the negative consequences of disposal
    - Define and describe the main parameters and processes to decrease these negative environmental impacts
- end of life information

Classing elements

There are different types of elements:

- Recyclable: waste wich can be turn into another form of new and reusable materials without specific treatment.
- Non-recyclable: the components or products that design for single-use, which means they get discarded immediately after use.
- Conditionally recyclable: component which needs specific treatment before being recycled.

Usual workflow for disposition:

- Distinguishing the recyclable, Conditionally recyclable and Non-recyclable components or products.
- Determining what material can recycle many times
- Identify the product lifespan for disposing and/or recycling.
- Describe how to recycle the components or products and their type of materials?
- Describe how to dispose of components which cannot be recycled.

Disposable products are most often made from Polystyrene and Cotton, Non-recyclable products from Textiles and Ceramics.

Types of disposal:

- Incineration: This type of waste disposal involves the dumping off method where you eliminate waste materials via combustion.
- Landfill: It involves collecting, transporting, dumping, and burying waste in a designated land.

Environmental assessment

It is the assessment of the environmental consequences of disposing or recycling a product before the decision to move forward with the proposed action. Indicate:

- The negative consequences of disposable products on the environment if sustainability isn't factored into disposal options.

- How to select the disposal or recycling process to have a less environmental impact?

Sources

Section 12.1

12 Sources

12.1 OpenNext work

The work was presented in a template at. https://github.com/OPEN-NEXT/WP2.3-Guideline-and-template-for-documentation-of-OSH-design-reuse/tree/main, based on the results provided as deliverable 2.6 at https://github.com/OPEN-NEXT/WP2.3-Guideline-and-template-for-documentation-of-OSH-design-reuse/blob/main/Sources/Deliverable2_6%20_Final%20release%20of%20models%20and%20standards%20for%20design%20reuse_V4_20220930.pdf

12.2 OSH-dir-std work

The content of the 04_hardware folder is loosely derived from https://github.com/hoijui/osh-dir-std/tree/main in discussion with one of the author.

12.3 Journal of open hardware

The journal of open hardware provides a template with several information required in hardware metadata papers. We included some of the indication of the template into this guide.

12.4 Journal hardwareX

The journal hardwareX provides a template with several information required in hardware metadata papers. Todd Duncombe. (2021). HardwareX manuscript templates. https://doi.org/10.5281/zenodo.5078227

12.5 Turing way book

Content taken from The Turing Way Community. (2025). The Turing Way: A handbook for reproducible, ethical and collaborative research. Zenodo. https://doi.org/10.5281/zenodo.15213042. (shared under a CC-BY license).

A specific chapter on OSH and the linked chapter on OS project documentation. NB: The team behind this guide was involved in the redaction of these chapters.

12.6 Docubricks

Docubricks is/was a metadata format providing best practices. The elements in this book were found on a wayback machine copy of the website in January 2025: http://web.archive.org/web/20250120163655/https://www.docubricks.com/best-practise-guide.jsp

12.7 Open know how manifest

Open Know-How is all about sharing knowledge on how to make things. The objective of this specification is to improve the open-ness of know-how for making hardware by improving the discoverability, portability and translatability of knowledge.
https://standards.internetofproduction.org/pub/okh/release/1

12.8 OKH - LOSH extension

Open Know-How Specification for the Library of Open Source Hardware subtitle: OKH-LOSH.
https://github.com/OPEN-NEXT/OKH-LOSH/blob/master/OKH-LOSH.md
Only few addition were made from this source, as most of it is either very technical or too specialised.

12.9 OHO know how

The open hardware observatory developed some best practices aimed at facilitating reviews of hardware. This is particularly directed toward the technical aspect of the hardware project, including how to create BOM and document CAD files: https://en.oho.wiki/wiki/Parts_list#Guidelines, https://en.oho.wiki/wiki/CAD_files, https://en.oho.wiki/wiki/Technical_drawings

12.10 OSHWA best practices

The OSHWA provides documentation on best practices for open hardware

https://oshwa.org/resources/sharing-best-practices/