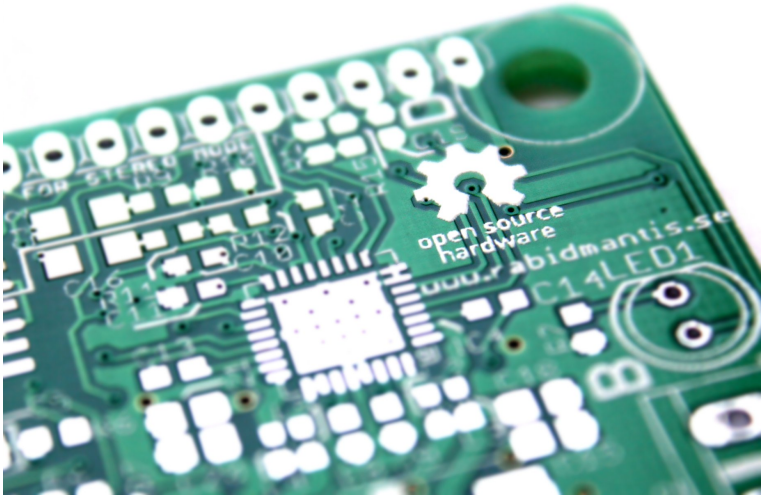
CHALMERS



Open Source Hardware

Can embedded electronics companies thrive through the use and/or development of open source hardware?

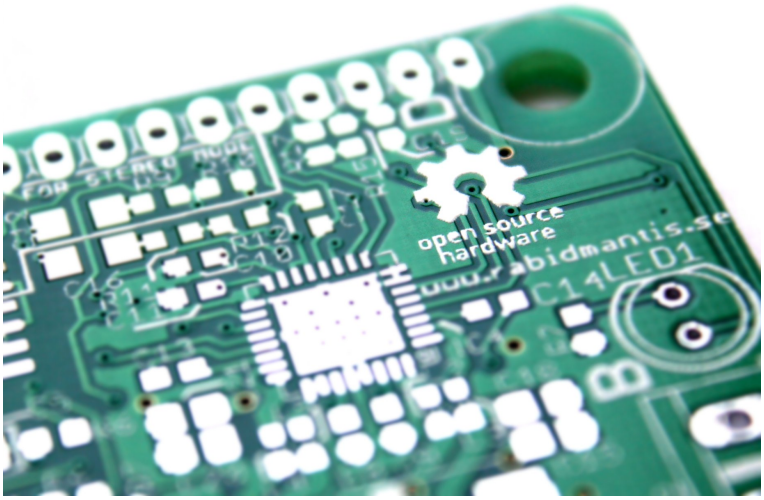
Jonathan Lock

Department of Technology Management and Economics
Division of Management of Organizational Renewal and Entreprenuersip – MORE
Chalmers University of Technology
Gothenburg, Sweden, June 2013
Report No. E2013:107

OPEN SOURCE HARDWARE

CAN EMBEDDED ELECTRONICS COMPANIES THRIVE THROUGH THE USE AND/OR DEVELOPMENT OF OPEN SOURCE HARDWARE?

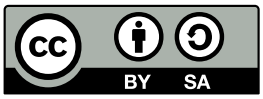
JONATHAN LOCK



Department of Technology Management and Economics
Division of Management of Organizational Renewal and Entreprenuersip – MORE
CHALMERS UNIVERSITY OF TECHNOLOGY
SE-412 96 Gothenburg, Sweden, June 2013

Jonathan Lock: *Open Source Hardware*, Can embedded electronics companies thrive through the use and/or development of open source hardware?, June 2013.

Department of Technology Management and Economics
Division of Management of Organizational Renewal and Entreprenuersip – MORE
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Telephone +46 (0)31 772 1000



The front cover illustration shows a Printed Circuit Board (PCB) for an Open Source Hardware (OSHW) device (titled *CheapAmp*) developed by the author and available at `http://www.rabidmantis.se`. This was the first OSHW device developed by the author.

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography *"The Elements of Typographic Style"*.

OPEN SOURCE HARDWARE

CAN EMBEDDED ELECTRONICS COMPANIES THRIVE THROUGH THE USE AND/OR DEVELOPMENT OF OPEN SOURCE HARDWARE?

Jonathan Lock, Department of Technology Management and Economics, Chalmers University of Technology

ABSTRACT

This report studies the ability for embedded electronics companies to effectively use and/or develop Open Source Hardware (OSHW)[1]. A lack of existing research into OSHW has led the author to conduct a web-based survey and two interviews with retailers and developers of embedded electronics modules, one with an OSHW work-flow and one without.

Analysis of the results show that both use and development of OSHW has the potential to be useful in a business context. Use of OSHW was found to be less risky than development, and particularly beneficial when used as a tool in a development setting; allowing for modification to match the requirements of the development process, as well as part of a product; where the OSHW device can be integrated into product development undertaken by the firm. Poor documentation and support, and the potential difficulty in determining the quality of a device were found to be the foremost negative aspects to using OSHW.

Developing OSHW as a part of business activities was also identified as feasible; users value OSHW devices highly, in particular if accurately documented and supported. However, the risk of third parties cloning or duplicating sections of the device was identified as a large risk due to the poor Intellectual Property (IP) protection available. Though successful companies developing OSHW have been identified, it remains to be determined which methods of OSHW development are realistic and can be effectively used.

Keywords: Open Source Hardware, Embedded Electronics, Business Models in Open Source Hardware, Open Source Hardware Documentation

1 Very briefly, OSHW is where the design files for physical devices (such as drawings, schematics, Printed Circuit Board (PCB) artwork, bill of materials, and so on) are distributed under a liberal license allowing users to freely use, copy, change, study, and improve the original device in a manner somewhat similar to that of Open Source Software (OSS).

OPEN SOURCE HARDWARE

CAN EMBEDDED ELECTRONICS COMPANIES THRIVE THROUGH THE USE AND/OR DEVELOPMENT OF OPEN SOURCE HARDWARE?

Jonathan Lock, Department of Technology Management and Economics, Chalmers University of Technology

SAMMANFATTNING

Den här rapporten undersöker möjligheterna för företag som arbetar inom inbyggd elektronik att använda och/eller utveckla Open Source Hardware (OSHW)[2]. Bristande forskning inom området har lett till att författaren har genomfört en internet-baserad enkät samt två intervjuer med företag som utvecklar och säljer moduler inom inbyggd elektronik, varav den ena arbetar med OSHW och den andra inte gör det.

Analys av resultaten visar att både användning och utveckling av OSHW har potentialen att vara mycket effektiv inom företag. Användning visade sig vara mindre riskfyllt än utveckling, och särskilt gynnsam dels som ett verktyg i utvecklingsmiljön; där möjligheten finns att göra förändringar som gör att enheten uppfyller utvecklingsmiljöns krav, och dels som en del av en produkt; där den integreras i produktutvecklingsprocessen på företaget. Undermålig dokumentation och support, samt att det kan vara svårt att bedöma kvalitén har uppmärksammats som nackdelar med användning av OSHW.

Att utveckla OSHW som en del av företagets verksamhet har också identifierats som en möjlighet, där användare värderar OSHW-produkter högt, särskilt om de är väldokumenterade och väl underhållna. Däremot är risken att en tredje part gör exakta kopior eller duplicerar delar av enheten stor, eftersom det endast finns mycket svaga immaterialrättsskydd för OSHW-produkter. Även om flera företag som framgångsrikt utvecklar OSHW har identifierats så är det fortfarande oklart vilka metoder av OSHW-utveckling som är realistiska och genomförbara.

Nyckelord: Öppen Hårdvara, Open Source Hardware, Inbyggd Elektronik, Affärsmodeller inom Öppen Hårdvara, Dokumentation inom Öppen Hårdvara

2 Kortfattat; fysiska enheter där utvecklingsfiler (såsom ritningar, scheman, mönsterkort, inköpslistor, och så vidare) distribueras under en licens där användaren fritt får möjligheten att använda, kopiera, ändra, studera, och förbättra ursprungsenheten på ett sätt som delvis liknar Open Source Software (OSS).

ACKNOWLEDGMENTS

I would like to thank my supervisor Mats Lundqvist for the useful comments and feedback throughout the development of this report; Chelsea Moll, for the extensive and thought-through responses to the interview with SparkFun Electronics; Jan Maléšek of Pololu for the clarifications to the interview with Make Magazine; Brain Benchoff of hackaday for posting the survey on http://hackaday.com; all of the participants of the survey, whose responses the majority of the conclusions of this report are based on; and Goli Mohammadi from Make Magazine, who permitted including their entire interview with Jan Maléšek of Pololu.

Note that Section A.1, Interview with Chelsea Moll from SparkFun Electronics [26]; Section A.2, Interview with Jan Maléšek, company president of Pololu [25]; and Section A.3, Maker Media's interview with Jan Maléšek [32] include verbatim content not written by the author which is reproduced with permission.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

CAD Computer Aided Design

EMI Electromagnetic Interference

EMC Electromagnetic Compatibility

FCC Federal Communications Commission

FPGA Field-Programmable Gate Array

HDL Hardware Description Language

IC Integrated Circuit

IDE Integrated Development Environment

IP Intellectual Property

OSHW Open Source Hardware

OSI Open Source Initiative

OSS Open Source Software

PCB Printed Circuit Board

RoHS Restriction of Hazardous Substances Directive

TTL Transistor-transistor logic

Part I

INTRODUCTION & THEORY

1 INTRODUCTION

Open Source Software (OSS) has fundamentally shaped the software industry; a radical notion where software is distributed with its source code, freely allowing anyone to modify and redistribute it for any purpose. This may sound like a completely unrealistic utopia — but nothing could be farther from the truth. OSS powered over 65% of the entire web-server market[1], 75% of all smart-phones[2], and nearly 94% of the top 500 supercomputers in the world[3] in 2012.

Open Source Hardware (OSHW) is an attempt to bring the same freedom and power to physical devices, but is as of yet still a newborn compared to OSS. The future success of OSHW largely depends on its adoption among developers and users; individuals, organizations, companies, and governments.

This report aims to shed some light on a few questions related to OSHW that have as of yet been unanswered in academic publications, primarily;

> Can embedded electronics companies thrive through the use and/or development of open source hardware?

This section introduces the concept of OSHW (briefly comparing it to the concept of OSS), lists the research questions this report will study along with applicable scope limitations, and finally describes the structure of this document.

1.1 BACKGROUND

In order to understand OSHW it is important to understand its conceptual parent OSS, which disrupted previously held notions of *property* and *rights* in the field of software. Weber [34, p. 16] describes this as;

> *Property in open source is configured fundamentally around the right to distribute, not the right to exclude.*

Perens et al. [27] define the core rights that OSS gives users as;

- The right to make copies of the program, and distribute those copies.
- The right to have access to the software's source code, a necessary preliminary before you can change it.

1 See http://news.netcraft.com/archives/category/web-server-survey/ and http://w3techs.com/technologies/overview/operating_system/all.
2 See http://www.idc.com/getdoc.jsp?containerId=prUS23771812.
3 See http://www.top500.org/statistics/overtime/.

- The right to make improvements to the program.

These rights allow anyone to without limitation study the workings of software released under an open source license, improve on it, and freely redistribute the derivative work.

OSHW is based on the same concept, one definition of this is, as per the Open Source Hardware Statement of Principles 1.0[1];

> *Open source hardware is hardware whose design is made publicly available so that anyone can study, modify, distribute, make, and sell the design or hardware based on that design. The hardware's source, the design from which it is made, is available in the preferred format for making modifications to it. Ideally, open source hardware uses readily-available components and materials, standard processes, open infrastructure, unrestricted content, and open-source design tools to maximize the ability of individuals to make and use hardware. Open source hardware gives people the freedom to control their technology while sharing knowledge and encouraging commerce through the open exchange of designs.*

Many of the topics that will be discussed later in the report are centered around the inherent differences between software and hardware; modifying software does not usually require hardware beyond an ordinary personal computer, has (virtually) no distribution cost, and has licenses based on copyright law that can give control over the way the code can be used — attributes that are not always true when it comes to hardware.

1.2 PURPOSE

Though there have been numerous studies into the feasibility of businesses centered around or heavily involving OSS, the same can not be said about OSHW. This report will attempt to shed some light on the ability of and suitable methods for embedded electronics firms to work with OSHW.

1.2.1 *Research questions*

The primary research question of this report;

> Can embedded electronics companies thrive through the use and/or development of OSHW?

will be discussed in itself and strengthened by addressing the secondary research questions;

- Which business models are feasible for embedded electronics companies working with OSHW?

- What potential limitations may an OSHW work-flow apply to embedded electronics companies?
- What potential gains could an OSHW work-flow give to embedded electronics companies?

The discussion of these questions will be backed up by using previous research where it exists, as well as adding new data through interviews and a web-based survey administered by the author. For the purpose of this report, embedded electronics companies refers to companies working primarily with embedded electronics — the development, sales, and support of electrical devices using passive components, diodes, transistors, and integrated circuits; both analog and digital.

1.2.2 *Scope limitations of report*

This report will study the ability for companies within the embedded electronics field to use or develop OSHW. Limiting the scope to companies within the embedded electronics field is done due to several factors,

- most companies developing, selling, or supporting OSHW today primarily work with embedded electronics, giving a suitable study group,
- most development within the OSHW field, from a hobbyist to commercial level, are centered around embedded electronics,
- the author's experience of OSHW lies within an embedded electronics setting.

Additionally, the business models presented in the discussion and conclusion have neither been tested as a part of the this report, nor will the actual method of implementing these business models be discussed.

Finally, though other authors in the OSHW field (such as Acosta [14]) define Field-Programmable Gate Array (FPGA) application code (such as Hardware Description Language (HDL) source code) as OSHW, this will not be done in this report. This is done as even though HDL describes hardware, the development and distribution of HDL source code is nearly identical to that of software.

1.2.3 *Report outline*

In Chapter 2 existing research on OSS is summarized, after which the primary differences between OSS and OSHW are listed, and finalized with a very brief summary of successful companies developing and selling OSHW devices.

In Chapter 3 the method of the two conducted interviews and the Internet-based survey is described, which is followed by the results of the survey in Chapter 4. In Chapter 5 and Chapter 6 the results are discussed and combined with existing knowledge on OSHW in order to draw conclusions based on the research questions of this report.

Chapter A in the appendix contains full transcripts of the conducted interviews, Chapter B lists the collected responses from the survey in a numerical format as well as listing the comments received in the survey, and Chapter C presents a proposed documentation style for OSHW devices.

2 THEORY

OSHW, being a relatively new phenomenon, has not been as thoroughly studied as its conceptual parent OSS. Due to this there is very little written academically about OSHW, which is why many of the sources used from here onwards consist of academic publications on OSS in combination with non-academic sources on OSHW (such as Internet blog posts, data from OSHW advocacy groups, and so on). As much of the data on OSHW remains unverified primary sources — such as individuals discussing OSHW — it is difficult to maintain as high a degree of scientific rigor as had there been a large amount of peer-reviewed data to rely on.

The goal of this section is to highlight the attributes of OSS that are relevant for OSHW, as well as bringing up what academic research there is on OSHW and "filling in" as much as possible with non-academic sources.

One potential issue with the non-academic sources is their volatility; many of the sources only exist as an online blog post that may be removed for little or no reason (in contrast to a published article which is generally archived and stored for a longer period). In order to preserve the usefulness of this report in the event that some of the non-academic content becomes unavailable, the core of the content referred to will be quoted and contained in this report; increasing the useful lifespan of this document.

2.1 OPEN SOURCE SOFTWARE

OSS is a concept that has very widely varying acceptance and beliefs, and is often hotly argued for or against, ranging from;

> *Linux* [and OSS with a reciprocal license] *is a cancer that attaches itself in an intellectual property sense to everything it touches*
> (Steve Ballmer, CEO of Microsoft [6])

to;

> *Software is like sex: it's better when it's free.*
> (Linus Torvalds, founder and current project coordinator of Linux [11])

The very varied and often emotionally laden opinions that individuals hold stems from (among others) the disruption of previously held notions of *property* and *rights*. Weber [34, p. 16] describes this

as "*Property in open source is configured fundamentally around the right to distribute, not the right to exclude*". To wit, the concept of OSS centers around the right for anyone to study, modify, and redistribute software, without cost, to anyone, and for any purpose.

Weber [34] gives some background to this by writing that for the early computing systems in the 1950's there was no distinction between *machine code* and *source code. Machine code* — the fundamental instructions executed by a computer — is easy enough to write for small and compact programs, but as the complexity increases it becomes difficult to write directly (and is very rarely used today in desktop PC's). *Source code* is an intermediate descriptor that is easier for humans to use when describing instructions that would be very complex using the limited set of instructions available in machine code. The source code can then be translated (or *compiled*) in a one-way process to machine code for execution.

As time progressed more sophisticated programming languages and compilers were developed by computer scientists, making it feasible to develop more complex programs. At the same time, this made improving or otherwise altering software dependent on access to its source code, as the compilation process is one-way and is not very effectively reversible. This made it very difficult to modify software unless the source code was distributed along with the machine code.

In the early days when software was developed at academic institutions source code was very often freely distributed, but this changed over time and today a very large amount of software is released under closed source licenses, where only the machine code is available (well known examples are the Microsoft Windows operating system, their Office software package, and Apple's OS X and iOS operating systems for desktop and mobile devices). The source code is viewed as a very important Intellectual Property (IP) asset and is fiercely guarded. At the same time, a significant amount of software is released under an open source license, which requires that the source code be distributed along with the machine code to allow for the user to study, modify, and redistribute the source and machine code.

There is no "one true" definition of OSS, with different groups defining OSS in slightly different ways. The Open Source Initiative (OSI), an organization that promotes OSS, defines OSS as software that is supplied under a license that complies with the following criterion[2];

1. Free Redistribution

   The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works

   The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

   The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups

   The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor

   The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

   The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product

   The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software

   The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

10. License Must Be Technology-Neutral

    No provision of the license may be predicated on any individual technology or style of interface.

Rosen [29] describes many of the numerous licenses that OSS can be released under, and groups them primarily into two types;

ACADEMIC LICENSES which allow using the source code for any purpose without any obligation for the licensee to distribute the derivative work under an open source license, for example allowing a company to use open source software in a closed source proprietary program.

RECIPROCAL LICENSES also allow for using the source code for any purpose, but require the derivative work to be distributed under the same license (and thereby making the source code freely available to those who use the derivative work).

The BSD license [7] is one example of an academic license, it was developed at the University of California when in need of a license to use when distributing software to the public.

The GNU GPL [9], developed at the Free Software Foundation[1], is an example of a widely-used reciprocal license. The reciprocity of this license is what Steve Ballmer refers to when stating that "*Linux is a cancer* [...]" in the quote at the start of this section.

The above description only briefly touches on the many subtleties present in the many OSS licenses available (all with varying degrees of rights). Feller et al. [17], Von Hippel [33], Weber [34], Rosen [29] offer a more thorough study of OSS licenses and what they imply with regards to those who develop and/or use OSS.

Bonaccorsi and Rossi [16] consider the behavior of individuals contributing to OSS and ask (among others);

- why do people contribute to OSS if they are not paid for it, and
- how do people spread over the entire globe manage to co-ordinate their development work without a traditional hierarchical structure?

The thought of developing software without monetary compensation may seem strange, as Glass [19] puts it;

1 See http://www.fsf.org/.

> *I don't know who these crazy people are who want to write, read and even revise all that code without being paid anything for it at all.*

Lerner and Tirole [24] describe a few non-monetary reasons for people to contribute to projects, such as; an improved reputation leading to an increased status and attention among peers and corporations; an increased ability to acquire venture capital; and receiving credit for contributions to a project.

Bonaccorsi and Rossi [16] list some slightly different reasons, firstly arguing that releasing software under an OSS license gives some intellectual gratification — having solved a problem, why not release the software so that others can concentrate on other problems rather than reinventing the wheel? For larger projects with multiple collaborators contributing leads to an increased reputation and prestige, leading to a strong and visible community. Bonaccorsi and Rossi [16] continue by describing that contributers often regard programming as an art form, where solving complex problems gives immense satisfaction. Finally, the pleasure of creativity is listed as an important aspect, where "[in] *the commercial world, the nightmare of delivery deadlines is transforming production into an assembly line*". Developers that are pressured to do work they feel is substandard, crude, or ugly can instead express themselves though OSS by developing software in a way that is more satisfying for themselves.

Bonaccorsi and Rossi [16] continue by stating that for larger OSS projects, the community of developers both ties together the project as well as generates a dynamic hierarchy that changes over time, unlike proprietary projects where a clear project leader is often designated initially. Ghosh and Prakash [18] estimate that 10% of the developers write over 70% of the source code, where this concentration of development work leads to a project leaders growing from the bottom up.

The ability to write modular source code is also credited by Bonaccorsi and Rossi [16] as a critical factor for developers to be able to be in completely different countries and yet be able to collaborate on a project. Far from being a monolithic whole, most OSS is written in modular sections with clearly defined methods of intercommunication, modules that can even be re-used between entirely different programs. Weber [34], Bonaccorsi and Rossi [16] both credit the Internet for being critical for the coordination of OSS projects, where email listings and newsgroups are publicly available to allow both long-time and completely new developers to have an equally clear view of the current status of a project, such as features that are under development, bug-fixes that are underway, and proposals for new designs.

OSS has shown itself to be more than an academic thought experiment, it is today very wide-spread and used in a large range of sit-

uations from simple developer utilities, to office software suites like OpenOffice[2], and entire operating systems such as Linux and BSD variants. The use of OSS is far from simple fringe cases of hobbyist programmers playing around in their free time; over 65% of the web-server market is powered by OSS and run on open source software operating systems[3], the open source Linux-based Android operating system developed by Google powered 75% of all smart-phones in 2012[4], and nearly 94% of the top 500 supercomputers run applications on a Linux operating system[5].

2.1.1 *Business Models in Open Source Software*

Krishnamurthy [23] and Hecker [21] list some applicable business models for companies whose products center (more or less) around OSS. In particular, Hecker [21] lists strategies of (among others);

SUPPORT SELLERS who can make a profit on media distribution, branding, training, consulting, custom development, and support of OSS. Red Hat[6] is one company that primarily works with this method (whose most well-known product is their Fedora Linux variant) and in 2012 had a yearly revenue of $1.13 billion.

LOSS LEADERS allow a subset of a company's products to be released under an OSS license, with more advanced or customized products available under other, proprietary, licenses.

WIDGET FROSTING can be applicable for companies working primarily with hardware where the drivers and interfaces for their hardware is released under an OSS license, allowing for the use of their hardware in situations beyond their first-party supported applications.

ACCESSORIZING may be useful for companies distributing books, computer hardware, and other physical items associated with or supportive of OSS.

SELL IT, FREE IT where a company initially releases its software under a closed source license, and eventually converts it to OSS when appropriate. For example, the source code for the game engine for the immensely popular Quake III computer game was released[7] under an OSS license approximately five years

2 See http://www.openoffice.org/.

3 See http://news.netcraft.com/archives/category/web-server-survey/ and http://w3techs.com/technologies/overview/operating_system/all.

4 See http://www.idc.com/getdoc.jsp?containerId=prUS23771812.

5 See http://www.top500.org/statistics/overtime/.

6 http://www.redhat.com/

7 And is available at ftp://ftp.idsoftware.com/idstuff/source/quake3-1.32b-source.zip.

> after the initial game release, generating massive interest and spawning several new games based on the old game engine[8].

Krishnamurthy [23], Weber [34] highlight the importance of maintaining good relations with the community that develops the OSS parts of the product — though the community is not (usually) driven by profit there is a great interest in making the product, and the OSS sections of it in particular, available to as broad an audience and as wide an application range as possible. As Krishnamurthy [23, p. 4] writes, "[the community] *wants any interested developer to have access to the entire code so that the person can tinker with it to make improvements*".

2.2 OPEN SOURCE HARDWARE

The OSHW concept has largely been born from the same philosophical concept that OSS was constructed from; the freedom of redistribution of verbatim copies, access to source files, and the right for others to freely distribute derivative versions of the device. OSHW differs from OSS in that it focuses on hardware — physical artifacts — rather than immaterial source code.

One description of OSHW[9] is, as per the Open Source Hardware Statement of Principles and Definition v1.0[1];

> *Open source hardware is hardware whose design is made publicly available so that anyone can study, modify, distribute, make, and sell the design or hardware based on that design. The hardware's source, the design from which it is made, is available in the preferred format for making modifications to it. Ideally, open source hardware uses readily-available components and materials, standard processes, open infrastructure, unrestricted content, and open-source design tools to maximize the ability of individuals to make and use hardware. Open source hardware gives people the freedom to control their technology while sharing knowledge and encouraging commerce through the open exchange of designs.*

Though the term OSHW is relatively modern, the concept of publicly releasing designs so that others can further build upon them is not new — one of the fundamental bases of the scientific theory is the ability for others to be able to replicate the results of experiments, both for the validation of original results as well as for furthering development.

Though OSHW shares many philosophical attributes with OSS, there are several practical differences between the two which are often cited as potential problems for the wide-spread adoption of OSHW. The

8 For example, ioquake3 http://ioquake3.org/.

9 One description among many, where most are at least philosophically like-minded.

most commonly cited issues (Thompson [30], Acosta [14], Torrone [32]) can typically be grouped into the following categories ;

DISTRIBUTION COSTS Though software can be widely distributed at next to no cost over the Internet, hardware has intrinsic distribution and storage costs.

AUTOMATED UPDATING Software updates can be distributed and applied automatically over the Internet, while changing hardware involves physically replacing sections and/or an entire unit, which is neither automatic nor free of cost.

EXPENSIVE EQUIPMENT Software development typically has limited hardware requirements, only requiring access to a personal computer and the device the program is intended for. Personal computers are relatively inexpensive and within the reach of nearly all individuals in industrialized countries. Electrical hardware development can have extensive equipment requirements, and though most equipment is within the economic range of companies, this may not be the case for amateur and semi-professional developers[32]. Though Ghosh and Prakash [18] found that 90% of the OSS projects they studied were developed by only one or two authors, Bonaccorsi and Rossi [16] note that the community of developers around OSS projects is critical for their survival. The same may also apply for OSHW projects, making the cost of equipment a potentially important factor.

HETEROGENEOUS DESIGN SOFTWARE Source code for nearly all programming languages is written in text format and stored in plain-text files that can be opened by nearly any platform without any proprietary applications. Electrical hardware development is done in a multitude of different competing Computer Aided Design (CAD) programs, nearly all of which use their own file format and many of which are expensive to purchase. Though the different CAD drawings can often be exported to generic file formats that can be used without exotic software these files are not easily modified and any changes cannot be applied to the original file.

VERSION MERGING As software source code is nearly always written in text format, automated version difference checking is feasible and often used[10] when merging different branches of software versions. This is in contrast to electrical hardware development where very few CAD programs offer merge functionality (also requiring the developers to use the same CAD software). This is further exacerbated when working with Printed Circuit Board (PCB) CAD, where it is nearly impossible to automatically

10 See http://pubs.opengroup.org/onlinepubs/9699919799/utilities/diff.html.

merge two or more different designs without extensive reworking.

WEAK LICENSES There are many licenses available for OSS, both academic and reciprocal, that have been upheld in courts[22]. Most reciprocal licenses use copyright law for enforcement of terms; by failing to abide by the terms stated, the user has no right to use the source code and can be convicted of copyright infringement. Many of the components of an OSHW design are not protected by copyright, such as a specific circuit design or component choice. This has caused most OSHW licenses to be more similar to academic licenses, allowing others to essentially do anything with the source files. This is not always desirable, reciprocal licenses in software are very popular — the Linux kernel for example is released under a reciprocal license[11] and Linus Torvalds, the founder of Linux, has stated *"Making Linux GPL'd was definitely the best thing I ever did"*[12]. One of few options available for OSHW licenses to be more reciprocal is through the use of patent licensing, the TAPR Open Hardware License[3] requires *"... those who benefit from an OHL design may not bring lawsuits claiming that design infringes their patents or other intellectual property."* It may be possible to construct other licenses where the patented sections of a design are only permitted to be used if derivative versions are released under the same license, thereby using patents instead of copyright as an enforcement mechanism. Boettiger and Burk [15] give an example of this in the biotechnology field, where certain research tools and techniques are released under a reciprocal license. Patent protection is not a complete substitute for the copyright protection that OSS receives, as it requires active work to apply for and maintain a patent, requires an inventive step, and is relatively expensive.

COMPLIANCE REQUIREMENTS OSS development for non safety-critical and non-medical applications can generally be done without compliance testing. Selling and distributing electrical hardware however requires adherence to regulations such as Electromagnetic Compatibility (EMC) limitations, Restriction of Hazardous Substances Directive (RoHS), and CE marking directives as applicable in Europe and by the Federal Communications Commission (FCC) in the United States. Regulation compliance testing can be very expensive, and beyond the abilities of amateur and semi-professional developers.

11 See http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/COPYING.

12 See http://web.archive.org/web/20070210224351/http://hotwired.goo.ne.jp/matrix/9709/5_linus.html.

SUPPLIER DEPENDENCE Software, being an intangible entity and essentially only information, does in itself require any suppliers. In practice, software development requires development hardware such as a PC and supporting software such as a compiler and an Integrated Development Environment (IDE). However, these are available from a broad range of suppliers and software written for widely-used platforms is at no major risk of obsoletion in the foreseeable future. OSHW devices however are typically constructed from numerous parts, and though some are commonly available from multiple distributors (such as 7400-series digital logic[13]) many parts are only available from a single supplier without any equivalent replacement (such as microprocessors, which are often produced by a single manufacturer; replacement would entail extensive modification to both the electrical design and software running on the microprocessor).

Though these attributes may seem to be unsurmountable disadvantages, Acosta [14] describes several business models that are applicable for OSHW businesses, and there are many companies that have become highly successful with business models where nearly all of their output is OSHW.

Acosta [14] describes applicable business models for companies working with OSHW that are nearly identical to those presented for OSS in 2.1.1. Described business models include OSHW adaptions of the previously mentioned support sellers, loss leaders, and widget frosters.

Creators and distributors of a broad range of electronic modules for hobbyist and prototyping purposes are one of groups [4, 8] of companies that have been successful; SparkFun Electronics had a staff of 115 and a revenue of $27.5 million during 2011[26].

Designers and manufacturers of single devices are another group that have achieved a high level of success in some regions; the Arduino, a low-cost and simple programmable microcontroller board primarily intended for hobbyists and non-programmers, has sold over 300,000 units and spawned a broad range of accessories[31]. The Raspberry Pi, a small very low-cost single-board computer originally intended for teaching computer science in schools, has sold over 700,000 units in only one year since its release in February 2012[20].

Though the success of the aforementioned companies and products may or may not be due to their adoption of an OSHW-oriented business model, their success proves at the very least that adopting an OSHW-centered business model can be successful.

13 An historically important family of Transistor-transistor logic (TTL) Integrated Circuit (IC) still used and widely available today.

Part II

METHODOLOGY & RESULTS

3 METHODOLOGY

As was noted in Chapter 2 there is a lack of empirical research into the viewpoints and attitudes of developers and users of OSHW, with most literature instead considering hypotheticals and drawing comparisons to studies on OSS.

In an attempt to remedy this lack of empirical data an online survey has been conducted, targeting individuals with a background or interest in embedded electronics. Another survey has also recently been conducted studying participation in OSHW [12], focusing on user's behavior and contribution. The survey conducted in association with this report instead focuses on the different perceptions and attitudes that individuals have of OSHW, spread across five different demographic groups, and is tailored for the specific research questions studied in this report.

In addition to the survey two interviews have been conducted, one with an embedded electronics company working with an OSHW workflow, and one with a relatively similar company working with a more traditional process. The goal of the interviews is to give some additional background when analyzing the responses to the survey, hopefully shedding more light on subtle aspects of the responses.

3.1 SURVEY

The behavior and attitude of developers and users of OSHW — the community that often develops and sustains it — is as important to study as the companies that work with OSHW. Understanding the reasons that developers and users contribute to and choose to use OSHW is crucial. A relatively simple online survey was administered to an audience with a background and/or interest in embedded electronics, primarily readers of the online magazine hackaday[10]. The survey audience was intentionally chosen to be individuals with previous knowledge of OSHW, as many of the questions relate to their experiences of OSHW. This does add a risk of self-selection bias, in that only those who persue OSHW chose to conduct the survey, which is expanded on in Chapter 5. Section 3.3 lists the questions and response choices presented in the survey.

3.2 INTERVIEWS

The goal of conducting interviews with two successful companies targeting roughly the same market (embedded electronics modules),

one with an OSHW work-flow, and one without, is to highlight the relative benefits and drawbacks that each style generates. The two interviewed companies are SparkFun Electronics "[...] *an online retail store that sells the bits and pieces to make your electronics projects possible.*"[1] and Pololu Corporation "[selling] *small electronics modules for robot and motion control.*"[2]. These companies were chosen for their similarities; developing and selling devices in roughly the same field, with a similar company size of 115 employees at SparkFun and 45 at Pololu (as of 2013), and both with knowledge and thoughts of OSHW. For the purpose of the interview the main differences highlighted are their perception towards and plans for OSHW; where SparkFun Electronics has extensively placed their products under an OSHW license, while Pololu has maintained a more traditional work-flow and has not released the inner workings of their devices.

Maker Media, owner of the MAKE magazine[3] has previously conducted an interview with Jan Maléšek of Pololu where many relevant questions were discussed; this interview is used as a base for the brief interview with Pololu conducted in connection with this report.

The two interviews conducted for the purpose of this report were done through an e-mail medium and full transcripts, both from the interview MAKE magazine conducted as well as the two conducted in connection with this report, are shown in Section A.1, Section A.2, and Section A.3.

Though there are several companies working within the same field as Pololu and SparkFun Electronics, adopters of OSHW or otherwise, scope limitations limit the number of interviews that can be done. Some examples of other successful companies working with OSHW (that may be useful to study when considering further research) are Adafruit Industries [4], the Arduino group [5], and Evil Mad Science LLC [8].

1 http://www.sparkfun.com
2 http://www.pololu.com
3 http://blog.makezine.com

3.3 SURVEY QUESTIONS

The survey, presented to those who participated as described in Section 3.1, follows in near-verbatim (with only stylistic changes necessitated by adapting from a web-based format to a paper format).

Each shaded box below corresponds to one web page as seen by the survey participants.

Open source hardware survey

This quick, 5-minute survey, aims to study the way Open Source Hardware (referred to as OSHW, see https://en.wikipedia.org/wiki/Open_source_hardware) is used and created. The goal of this survey is to aid a Master's thesis studying the regions where electronics firms can sustain themselves with OSHW.

This survey is primarily targeted towards people with some prior experience with electrical OSHW projects, such as the Arduino, Raspberry pi, or any project you may have made yourself. In this survey some comparisons are made to proprietary devices, which in this setting means any device where the source documents (design calculations, CAD drawings, PCB artwork, and so on) are **not** freely distributed. Think of devices like an ordinary notebook computer, an alarm clock, or audio amplifier.

The entire survey is completely anonymous and only aggregate data will be publicly released.

Thank you for your time and input,
Jonathan Lock
http://www.rabidmantis.se

OSHW Usage

Have you used an or several OSHW device(s) within the past year?

- Yes *(continue)*
- No *(go to OSHW Development)*
- Unsure *(go to OSHW Development)*

OSHW Usage (contd.)

In which setting did you use the OSHW device? (If both, please do this survey twice, once for each choice).

- Personal, educational, or noncommercial
- Commercial

In which way(s) have you used OSHW device(s)? (Check all that apply)

- Directly, as a finished product, with next to no modifications
- With relatively minor adjustments (such as minor code changes or component substitution)
- With relatively large changes (such as a near complete code rewrite or PCB changes)
- As a base to design a completely different device (such as re-using small amounts of code or duplicating schematic or PCB sections)
- Other: .....

Would you have been able to use an otherwise equivalent proprietary solution to solve the same problem(s)?

- Always
- Most of the time
- Sometimes
- Rarely
- Never
- Unsure
- Other: .....

OSHW Development

Have you contributed to or started OSHW project(s) within the past year? (A publicly available or published project; available online, in a magazine, or elsewhere)

- Yes *(continue)*
- No *(go to OSHW in general)*
- Unsure *(go to OSHW in general)*

OSHW Development (contd.)

In which form have you contributed to the project(s)? (Check all that apply)

- Started personally or in a group
- A major contribution to an already existing project (such as an architectural code rewrite or a change that required extensive discussion)
- A minor contribution to an already existing project (such as a minor code bug-fix or a change that does not require much discussion)
- Other: .....

Have other people used/modified/hacked with the project(s) you started or contributed to after publication? (Check all that apply)

- Yes
- Unsure, but people have considered / discussed doing it
- Unsure, but probably
- Unsure
- No
- Other: .....

Why did you contribute/start the project? (Check all that apply)

- To be able to use the updated project to solve an immediate problem
- As an educational activity
- As a hobby (for the lulz)
- Other: .....

OSHW in General

In general, in your experience, how would you rate a simple (such as a DC/DC converter module, Arduino shield, headphone amplifier, or similar) OSHW device and proprietary device solving the same problem as in <Table 1 on page 24>?

In general, in your experience, how would you rate a complex (such as a cell phone, raspberry pi, autonomous helicopter, or similar) OSHW device and proprietary device solving the same problem as in <Table 1 on page 24>?

In your experience, how much value do you place on the attributes of OSHW as in Table 2 on page 25?

Do you have any other thoughts on the various benefits and drawbacks of OSHW compared to proprietary solutions?

- .....

	OSHW clear winner	OSHW slightly better	Roughly equally good	Proprietary slightly better	Proprietary clear winner	Unsure
Functionality (what it can do)	☐	☐	☐	☐	☐	☐
Stability (how well engineered and thought-through the device is)	☐	☐	☐	☐	☐	☐
Cost (for an equivalent product)	☐	☐	☐	☐	☐	☐
Documentation (how clearly described it is to use)	☐	☐	☐	☐	☐	☐
Support (how easy it is to get help if something goes wrong)	☐	☐	☐	☐	☐	☐
Gut feeling (all in all, your preference in total)	☐	☐	☐	☐	☐	☐

Table 1: Rating table for OSHW and proprietary devices.

	Very important	Slightly important	Neither important nor unimportant	Somewhat unimportant	Completely unimportant
The ability to see the way a project works (electrical schematic, component choices, PCB layout, and so on)	☐	☐	☐	☐	☐
The ability to modify a project (such as altering the program code or changing the PCB layout)	☐	☐	☐	☐	☐
The ability to reuse parts of a project as a reference for own designs (such as re-using small amounts of code or duplicating schematic or PCB sections)	☐	☐	☐	☐	☐
The philosophical and ethical aspects of OSHW (the perceived net benefit to society by designs and their derivatives being available to the public)	☐	☐	☐	☐	☐

Table 2: Rating table for OSHW values.

Your background

What is your educational background?

- Some high school
- Finished high school
- Some college/university (without any degree)
- Finished bachelor's degree or equivalent
- Finished master's degree or equivalent
- Finished PhD or equivalent or higher

What is your age?

- Under 15
- 15 to 18
- 19 to 22
- 23 to 27
- 28 to 33
- 34 to 39
- 40 to 47
- 48 to 57
- 58 to 68
- Over 68

Where are you from? (Choose the region you feel most accurately describes your background)

- Africa
- Asia
- Australia
- Europe
- North America
- South America

4 RESULTS

The results of the survey described in Section 3.1 is contained in this section, together with a brief analysis of the results on a question-by-question basis. A more broad and comprehensive analysis of the results is performed in Chapter 5. The numerical values that underly each graphic are listed in Table 4 on page 97 (in Appendix B). As part of the survey, some users added an optional comment. These comments are listed in Section B.1 and analyzed in Chapter 5.

The survey was made publicly available on the Internet and received responses from 2013-02-24 through 2013-03-25. The survey was was publicly seen in several locations, however, the vast majority of the responses collected originated from viewers of hackaday [10]. Approximately 420 responses (over 90%) came from participants through from hackaday [10], with the remainder, a 30-odd participants, from AVR Freaks[1], Svenska Elektronikforumet[2], and personal contacts. These sources were all largely similar in their distribution of responses and no distinction has been made in the analysis of the data regarding the different response sources.

Five different participant groups have been identified as taking part in the survey;

NONCOMMERCIAL USERS Who use but do not develop projects and devices for their own, educational, or non-profit purposes.

NONCOMMERCIAL DEVELOPERS Who develop (by modifying existing devices or creating something from scratch) for their own, educational, or non-profit purposes. Individuals in this group may also be noncommercial users.

COMMERCIAL USERS Who use but do not develop devices for commercial (for-profit) purposes.

COMMERCIAL DEVELOPERS Who develop devices (by modifying existing devices or creating something from scratch) for commercial (for-profit) purposes. Individuals in this group may also be commercial or non-commercial users.

INEXPERIENCED INDIVIDUALS Who do not have any experience using or developing OSHW devices at all.

1 An online community primarily centered on Atmel's AVR family of microprocessors, available at http://www.avrfreaks.com.

2 A Swedish internet forum primarily targeting hobbyists for the discussion of electronics and related subjects, available at http://elektronikforumet.com.

All of the results are shown in Figure 2 on page 32 through Figure 24 on page 51. The responses are divided into the aforementioned participant groups and shown side-by-side for comparison.

In some of the questions posed some participants chose not to answer. This causes the sum of the number of answers chosen for that question to be less than 100% of that participant's group size, and is shown in the figures as a white space (such as in Figure 7 on page 38, seen most clearly for the commercial users, where some 5-10% chose not to answer). Additionally, in some questions there was an "other" response option where the user could add a custom response should none of the other response choices be applicable. There were very few who elected to use this option[3], and **for the sake of simplicity these responses have been ignored from all analysis**. For increased readability, the "other" choices are not shown in the question excerpts in the following sections.

Due to the asymmetric distribution of individuals in each participant group — varying from 271 to 12 — the relative confidence of statistical certainty for the answers in each group varies greatly. For questions with binary response choices, such as "Have you directly used an OSHW device?" (paraphrased from the second question in the *OSHW Usage* section), a 95% ($\pm 2\sigma$) confidence interval has been calculated. This applies to Figure 1 on page 31, Figure 3 on page 33, Figure 4 on page 34, and Figure 5 on page 35 and is illustrated with thin red bars signifying the statistical certainty of the results — the "true" value (the value that would be found with an infinitely large sample group) lies somewhere within the bounds of the red bar.

The confidence interval is calculated by using the normal approximation interval, which states that $p = \hat{p} \pm z_{1-\frac{1}{2}\alpha}\sqrt{\frac{1}{n}\hat{p}\,(1-\hat{p})}$, where p gives the range of true values for the measurement, $\hat{p}$ is the measured value, $z_{1-\frac{1}{2}\alpha}$ is the $1-\frac{1}{2}\alpha$ percentile of a standard normal (Gaussian) distribution, α is the error percentile, and n is the number of individuals in the participant group [28]. In this case, with a 95% confidence interval, $\alpha = 0.05$ and $z_{1-\frac{1}{2}\alpha} \approx 1.96$.

This calculation assumes a normally distributed sample group and a sample size that is large enough. (A rough rule-of-thumb states that if $n\hat{p} > 5$ and $n\,(1-\hat{p}) > 5$ the sample size is large enough). In this case, neither of these attributes can be said to be true — there is nothing to say the audience is a unbiased independently chosen set of individuals, and for some questions $n\hat{p} = 1.2 \ll 5$. This makes the confidence intervals shown useless for statistical inferences, though they can still be used for the intuitive benefit of the reader[4]. In short, the error bars are useful when eyeballing the figures to get a sense of

3 Most of the responses in the "other" category were non-responses, such as "hard to say" and similar formulations.

4 This is also why the relatively crude normal approximation interval method is chosen, rather than a more sophisticated method of calculating the confidence interval.

Table 3: Number of individuals in each participant group.

Participant group	Sample size	Proportion of survey
Noncommercial user	271	59.7%
Noncommercial developer	103	22.7%
Commercial user	17	3.7%
Commercial developer	12	2.7%
Inexperienced participant	51	11.2%
Total participants	454	100%

accuracy, **but should not be used to draw sensitive statistical conclusions!**

For non-binomial result types (such as where the user rates an attribute on a scale with several choices) a confidence interval is not shown, as this would make the graphical plots very unclear without adding any immediate benefits.

Table 3 on page 29 lists the number of responses in each participant group. In general, there are more users than developers, and more in the non-commercial group than the commercial group. When analyzing the results, keep in mind that the commercial users and developers are of a relatively small sample size, making the confidence interval relatively coarse.

4.1 TERMS USED IN RESULTS

Due to space constraints, some terminology is used in the presented results.

In Figure 1 on page 31 though Figure 24 on page 51, the five different participant groups; noncommercial users, noncommercial developers, commercial users, commercial developers, and individuals with no prior experience of OSHW are referred to as "NC user", "NC dev.", "C user", "C dev.", and "No Exp." respectively.

In Figure 6 on page 37 through Figure 17 on page 44, which lists the results from the questions in Table 1 on page 24, the six response categories "OSHW clear winner", "OSHW slightly better", "Roughly equally good", "Proprietary slightly better", "Proprietary clear winner", and "Unsure" are referred to as "OSHW++", "OSHW+", "Equal", "Prop. +", "Prop ++", and "Unsure" respectively.

In Figure 18 on page 46 through Figure 21 on page 48, which list the results from the questions in Table 2 on page 25, the five response categories "Very important", "Slightly important", "Neither important nor unimportant", "Somewhat unimportant", and "Completely unimportant" are referred to as "Very imp.", "Slightly imp.", "Neither", "Slightly unimp.", and "Comp. unimp." respectively.

4.2 OPEN SOURCE HARDWARE USAGE

The first part of the study involves looking at how OSHW is used and contains responses from all participant groups other than those with no prior experience of OSHW. The collected answers to the first question;

> *In which way(s) have you used OSHW device(s)? (Check all that apply)*
>
> - Directly, as a finished product, with next to no modifications
> - With relatively minor adjustments (such as minor code changes or component substitution)
> - With relatively large changes (such as a near complete code rewrite or PCB changes)
> - As a base to design a completely different device (such as re-using small amounts of code or duplicating schematic or PCB sections)

are shown in Figure 1 on page 31, with 95% confidence intervals shown with red bounding lines.

Though the small sample size of the commercial user and commercial developer groups make it difficult to draw very many inferences, some statistically significant results are;

- OSHW is for the most part used directly "as-is" or with relatively minor changes, both among developers and users. Commercial developers are more likely than non-commercial developers to use various parts of an existing design when making something new.
- The sum of all responses for the non-commercial users is approximately 1.9; meaning that the average non-commercial user of OSHW has used OSHW in slightly below two of the ways listed in the question. As all but one of the responses involves some development work, **the average non-commercial user is also, to some extent, a developer**. Due to the small sample size, it cannot be said whether or not this is also true for commercial users.
- Noncommercial users (and possibly commercial users as well) are most likely to use a device as-is or with relatively minor changes, while developers are more likely to modify or use devices as a reference for other development work.

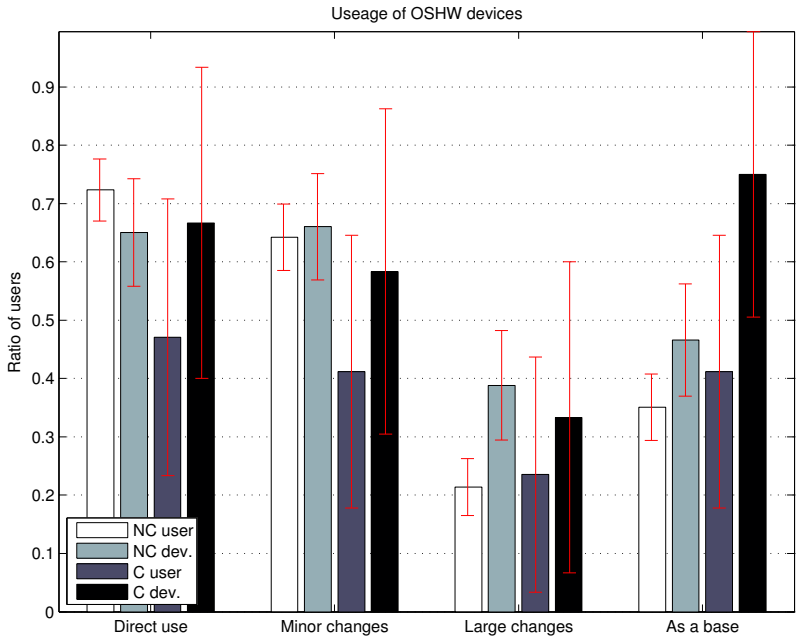


Figure 1: Responses to survey question "In which way(s) have you used OSHW device(s)?".

The responses to the second question;

Would you have been able to use an otherwise equivalent proprietary solution to solve the same problem(s)?

- Always
- Most of the time
- Sometimes
- Rarely
- Never
- Unsure

are shown in Figure 2 on page 32 with the response choices presented in a stacked format.

In general, commercial users and developers reported an increased ability to use proprietary equivalents, while noncommercial users and developers more often found there were fewer proprietary equivalents available. Interestingly, both developers groups displayed a reduced ability to use proprietary equivalents, while both users groups were more able to use proprietary equivalents. The largest disparity between groups is the noncommercial developers and the commercial users, who found it the most difficult and the most easy to use a proprietary equivalent respectively.

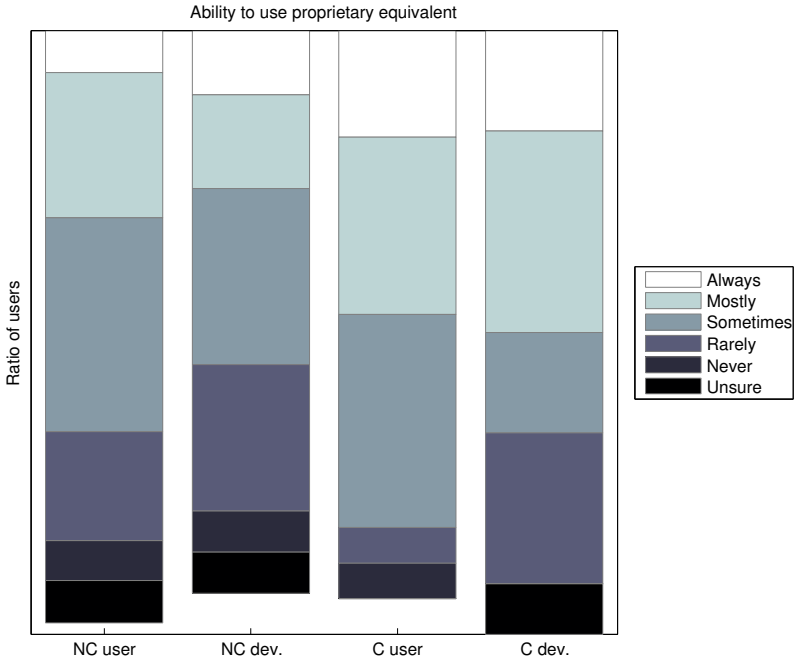


Figure 2: Responses to survey question "Would you have been able to use an otherwise equivalent proprietary solution to solve the same problem(s)?".

4.3 OPEN SOURCE HARDWARE DEVELOPMENT

The second part of the study inspects the way that developers have contributed to their OSHW projects. Due to the nature of the questions, responses only exist for noncommercial and commercial developers. The responses to the question in this section;

> In which form have you contributed to the project(s)? (Check all that apply)
> - Started personally or in a group
> - A major contribution to an already existing project (such as an architectural code rewrite or a change that required extensive discussion)
> - A minor contribution to an already existing project (such as a minor code bug-fix or a change that does not require much discussion)

are shown in Figure 3 on page 33 with 95% confidence intervals shown with red bounding lines. Due to the small sample size, no statistically significant differences can be inferred from the results on a per-question basis.

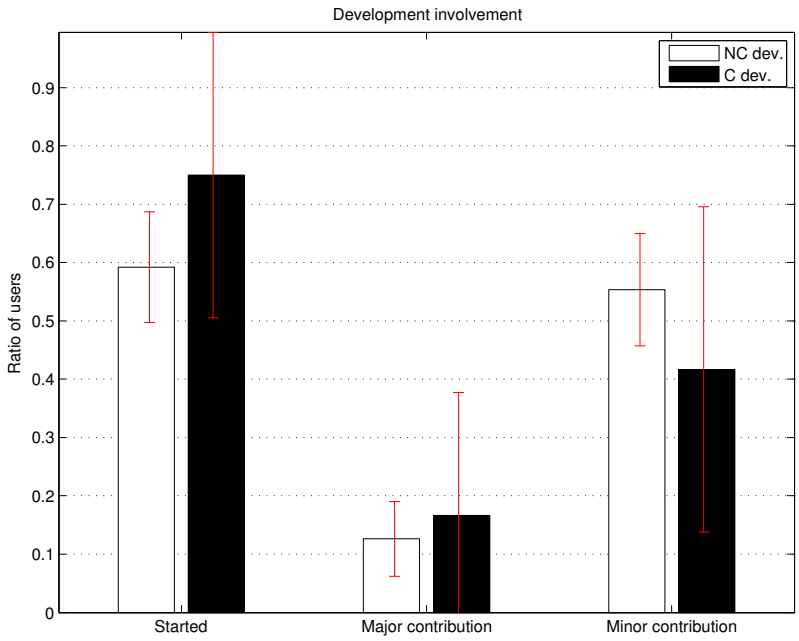


Figure 3: Responses to survey question "In which form have you contributed to the project(s)?".

The response to the second question in this section;

Have other people used/modified/hacked with the project(s) you started or contributed to after publication? (Check all that apply)

- Yes
- Unsure, but people have considered / discussed doing it
- Unsure, but probably
- Unsure
- No

is shown in Figure 4 on page 34 with 95% confidence intervals shown with red bounding lines. In the same way as with the previous question, the small sample size makes it impossible to draw statistically significant differences from the results on a per-question basis.

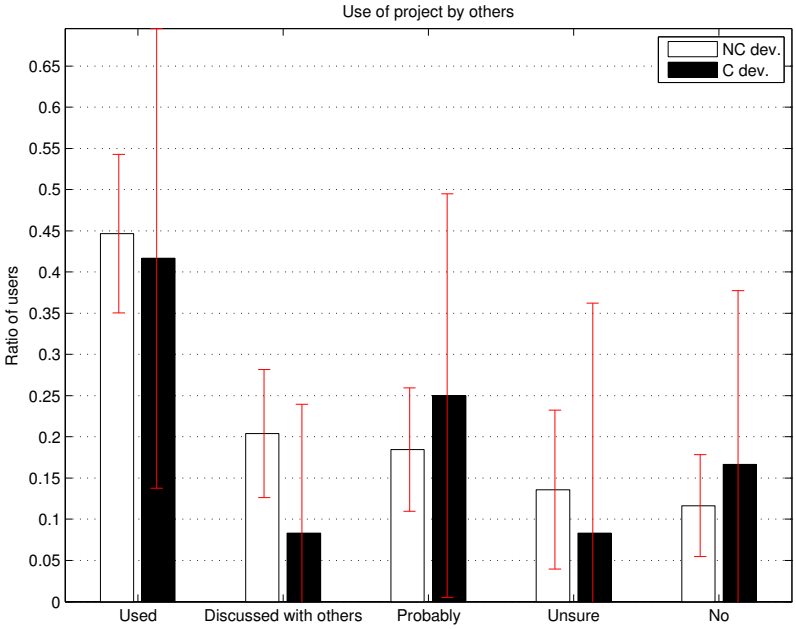


Figure 4: Responses to survey question "Have other people used/modified/hacked with the project(s) you started or contributed to after publication?".

The responses to the final question in this section;

Why did you contribute/start the project? (Check all that apply)

- To be able to use the updated project to solve an immediate problem
- As an educational activity
- As a hobby (for the lulz)

are shown in Figure 5 on page 35 with 95% confidence intervals shown with red bounding lines. This question, in contrast to previous

questions in this category, does have responses that are statistically significant on a per-question basis. Commercial developers are more likely than non-commercial developers to contribute or start a project to solve an immediate problem, while non-commercial developers reported that significant contributions are made for hobby purposes (with no commercial developers choosing this option).

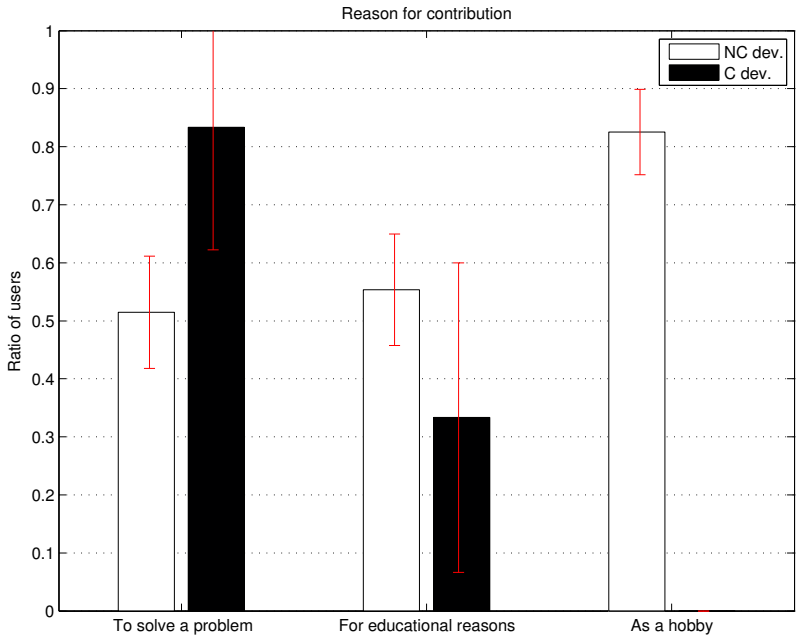


Figure 5: Responses to survey question "Why did you contribute/start the project?".

In light of this data — non-commercial developers quite frequently contribute to projects "for fun", while commercial developers are more likely than non-commercial developers to contribute to solve a problem — it is possible to generate broad results based on the data in, among others Figure 3 on page 33 and Figure 4 on page 34; this is presented in 5.2.1.

4.4 OPEN SOURCE HARDWARE VS. PROPRIETARY HARDWARE

This section of the survey studies the perceived benefits and drawbacks of various aspects of OSHW compared to proprietary hardware. This is the first part of the survey where all five participant groups have been asked questions. The relative strengths and weaknesses of OSHW are grouped into simple and complex device groups, as formulated below;

> In general, in your experience, how would you rate a simple (such as a DC/DC converter module, Arduino shield, headphone amplifier, or similar) OSHW device and proprietary device solving the same problem as in <*rating table*>?
>
> In general, in your experience, how would you rate a complex (such as a cell phone, raspberry pi, autonomous helicopter, or similar) OSHW device and proprietary device solving the same problem as in <*rating table*>?

and scored along six different performance metrics, as shown in the rating table below;

	OSHW clear winner	OSHW slightly better	Roughly equally good	Proprietary slightly better	Proprietary clear winner	Unsure
Functionality (what it can do)	☐	☐	☐	☐	☐	☐
Stability (how well engineered and thought-through the device is)	☐	☐	☐	☐	☐	☐
Cost (for an equivalent product)	☐	☐	☐	☐	☐	☐
Documentation (how clearly described it is to use)	☐	☐	☐	☐	☐	☐
Support (how easy it is to get help if something goes wrong)	☐	☐	☐	☐	☐	☐
Gut feeling (all in all, your preference in total)	☐	☐	☐	☐	☐	☐

The results of the scoring of these twelve questions follows, divided into simple devices (in 4.4.1) and complex devices (in 4.4.2). Each performance metric is shown in a stacked format and divided into each of the five participant groups.

4.4.1 *Simple devices*

Figure 6 on page 37 through Figure 11 on page 40 displays the participant perception of simple devices rated by six performance metrics. Some results are directly apparent from these responses;

- The perception of simple OSHW devices in general is very positive, Figure 11 on page 40 shows a very high gut feeling among all users. This is upheld throughout the other metrics as well, with the average rating among all participants ranging from "*OSHW clear winner*" to "*Roughly equally good*".
- The cost of OSHW devices is the most favorable metric for simple devices, while stability and support were the attributes that OSHW devices fared most poorly with.
- There is a clear disparity among the non-commercial and commercial groups. Commercial users and developers were generally less inclined towards OSHW devices, especially among functionality, documentation, and support. In particular, commercial users rated OSHW documentation as significantly poorer than the rating noncommercial users gave. However, commercial users and developers rated the cost metric even more positively than noncommercial users and developers.

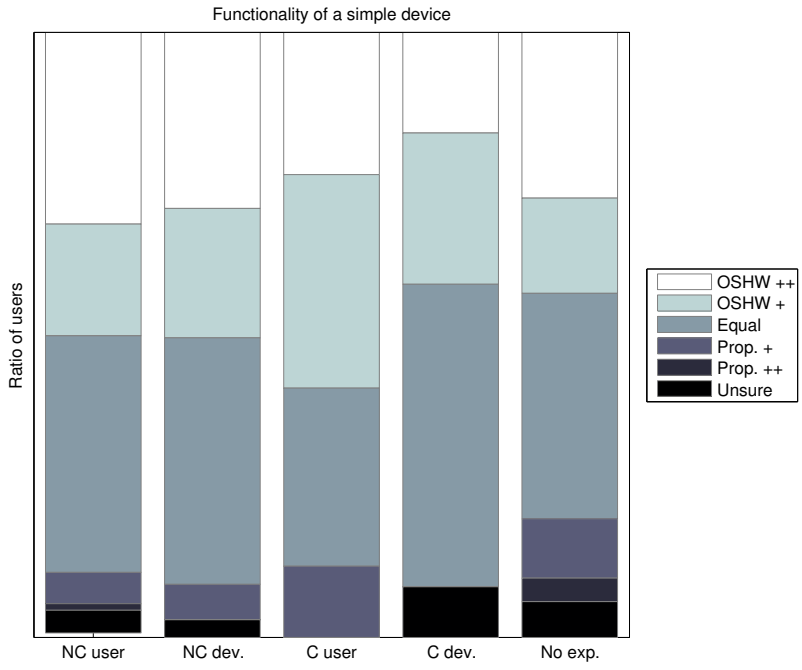


Figure 6: Responses to functionality metric for simple devices.

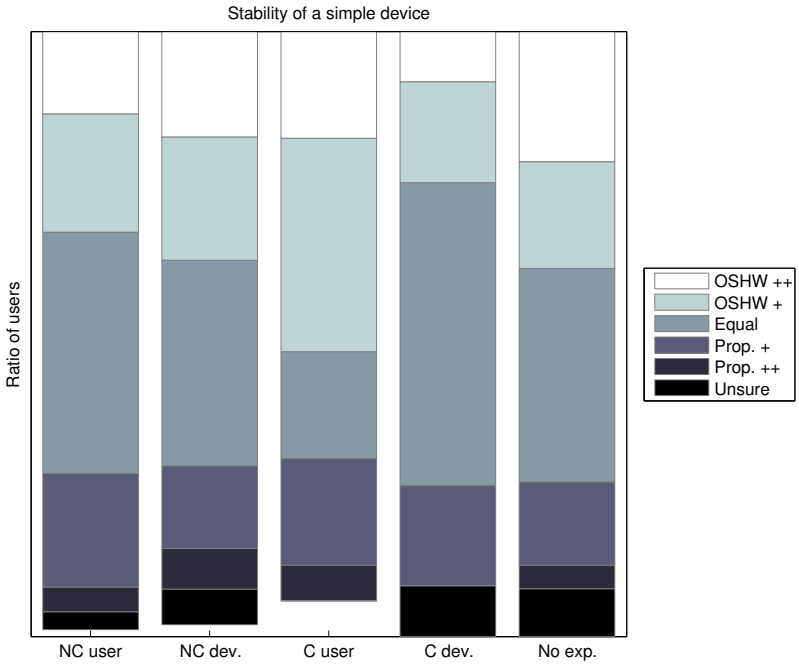


Figure 7: Responses to stability metric for simple devices.

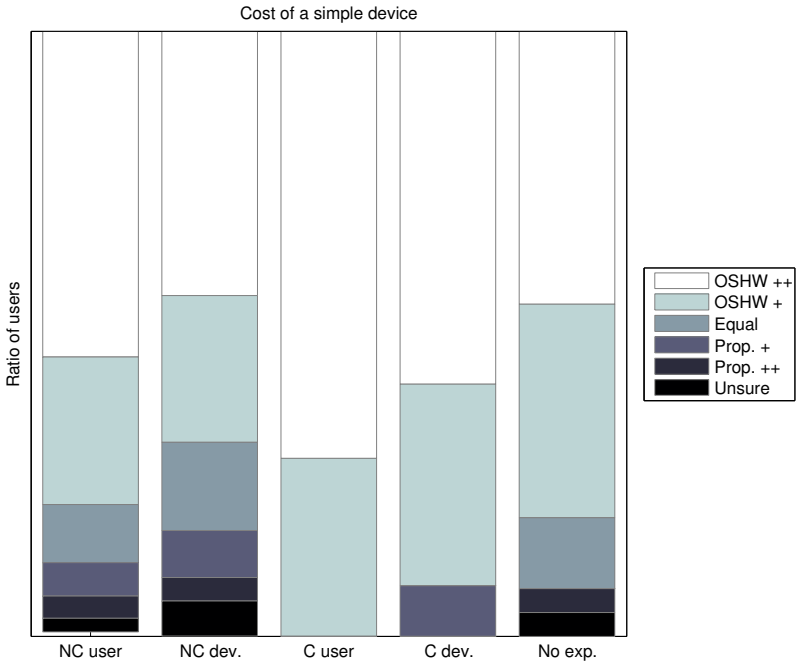


Figure 8: Responses to cost metric for simple devices.

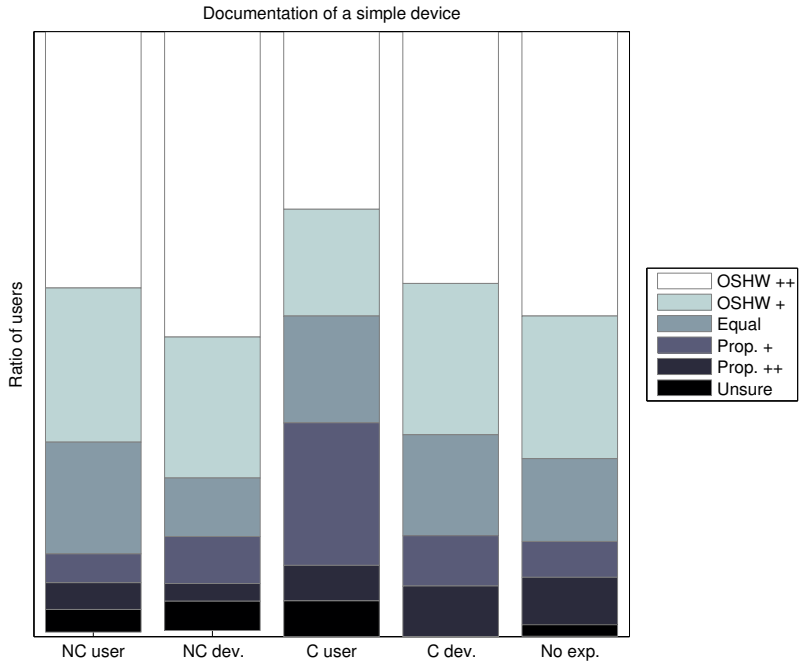


Figure 9: Response to documentation metric for simple devices.

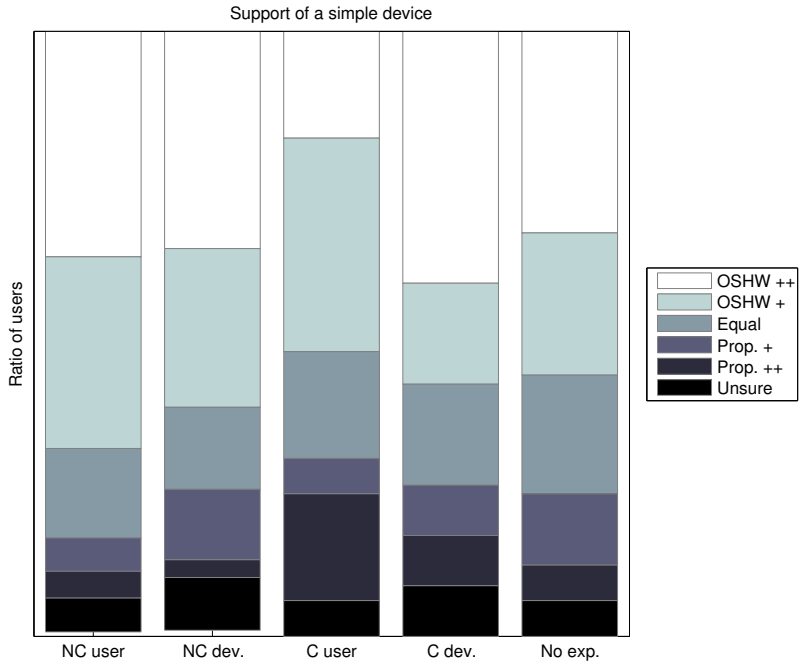


Figure 10: Response to support metric for simple devices.

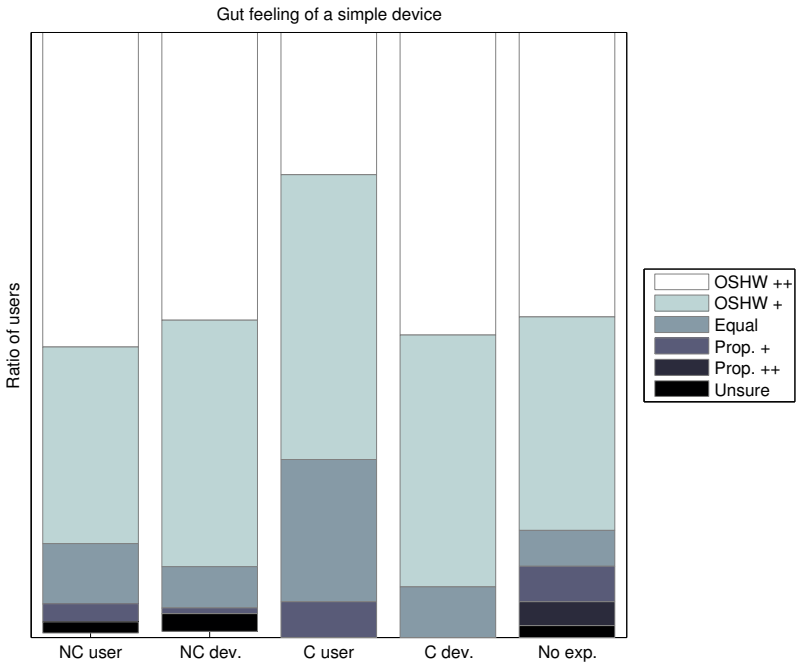


Figure 11: Response to gut feeling metric for simple devices..

4.4.2 *Complex devices*

Figure 12 on page 41 through Figure 17 on page 44 displays the participant perception of complex devices rated by six performance metrics. Some results are directly apparent from these responses;

- Through the responses to the gut feeling metric (Figure 17 on page 44) were generally positive with a mean response of "*OSHW slightly better*" for all participant groups, the responses to four of the five remaining metrics were significantly poorer (the cost metric being the exception to this).
- There is a significant degree of relative similarity among the responses between the simple and complex device categories. However, the average level is less favorable of OSHW in the complex device category. Two notable aspects are;
  - As was the case as for simple devices, commercial users in particular found the quality of documentation to be worse than the noncommercial groups.
  - Commercial developers in particular are more negative towards complex OSHW devices.

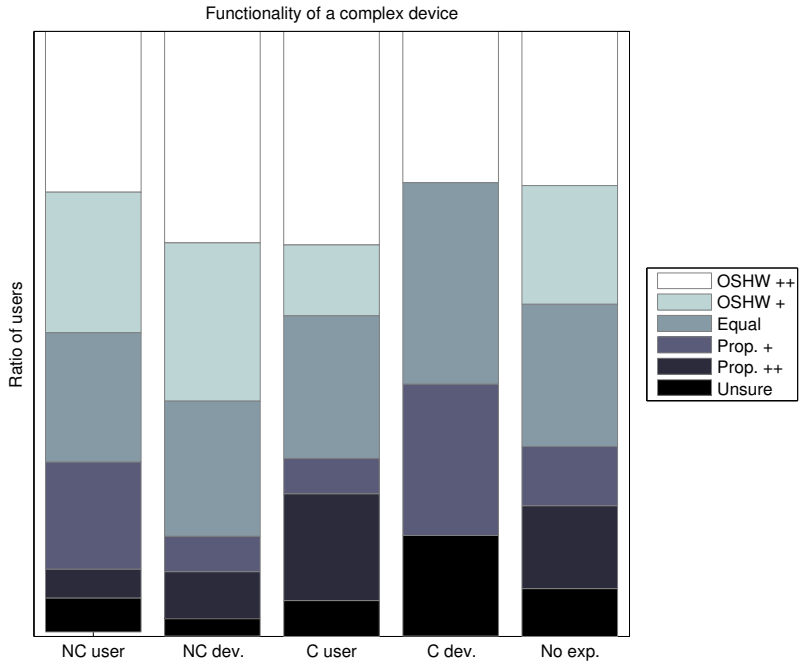


Figure 12: Response to functionality metric for complex devices.

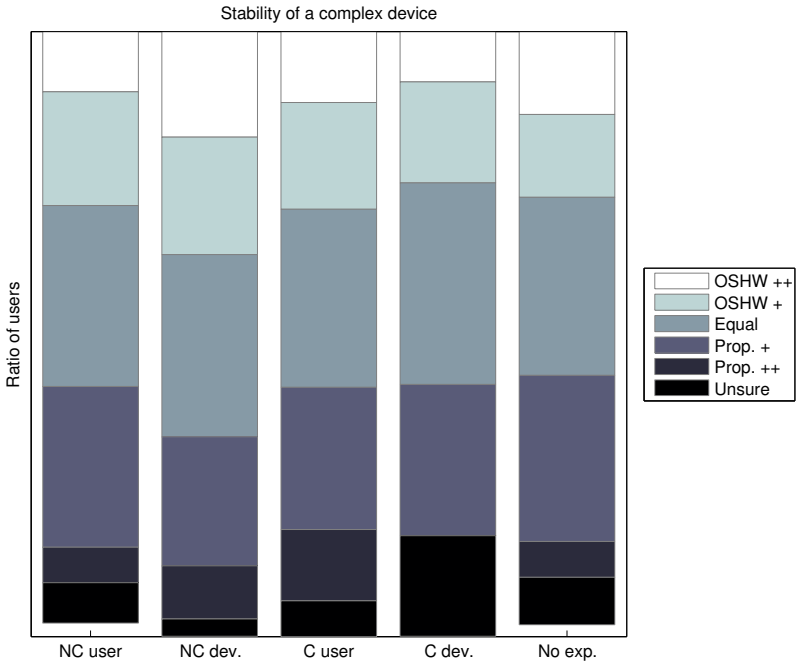


Figure 13: Response to stability metric for complex devices.

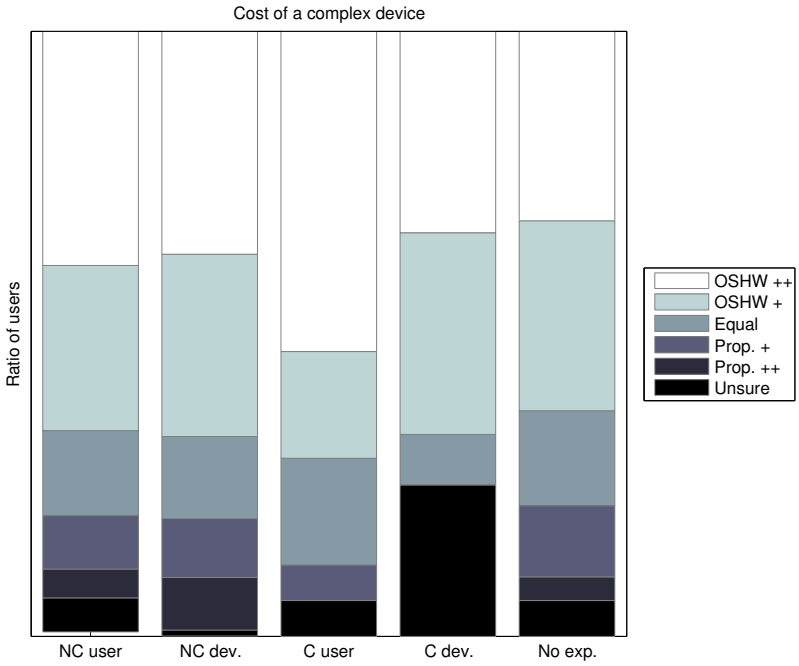


Figure 14: Response to cost metric for complex devices.

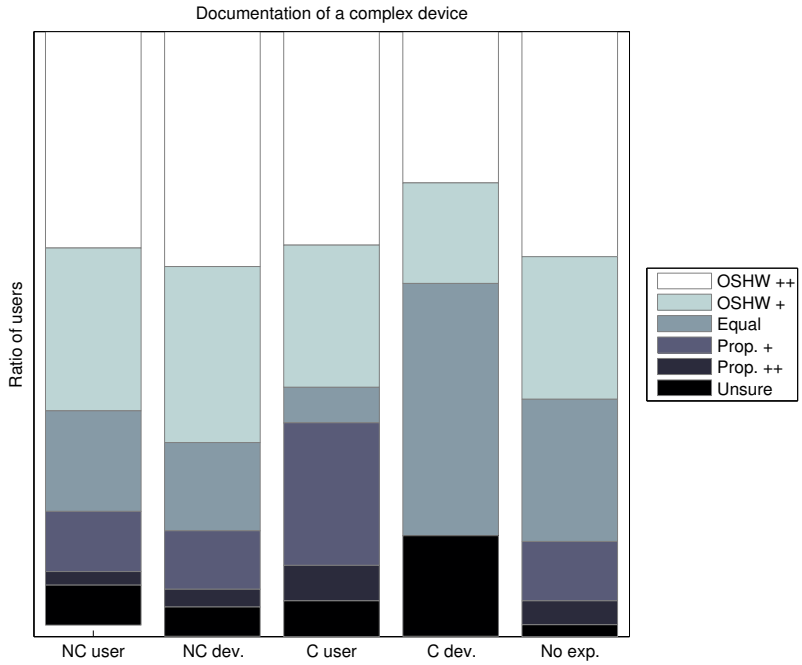


Figure 15: Response to documentation metric for complex devices.

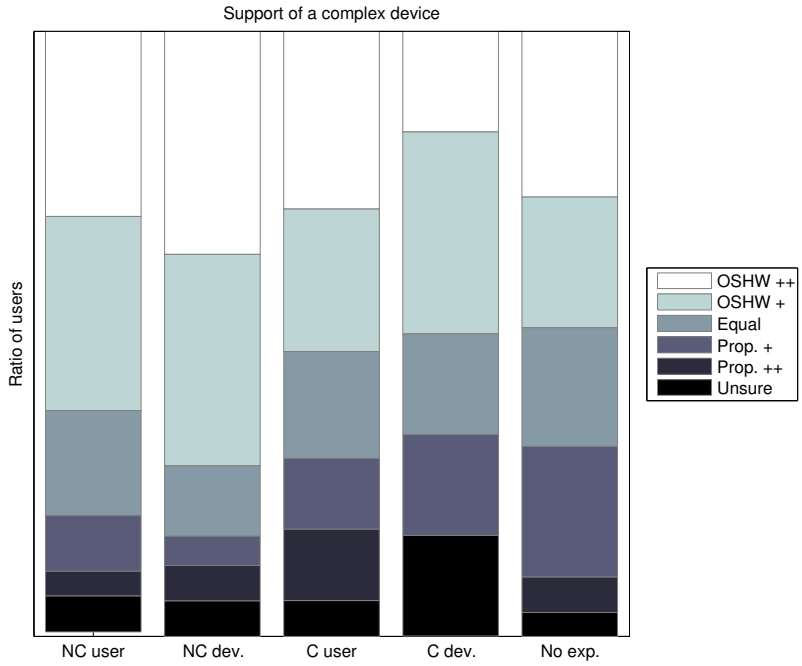


Figure 16: Response to support metric for complex devices.

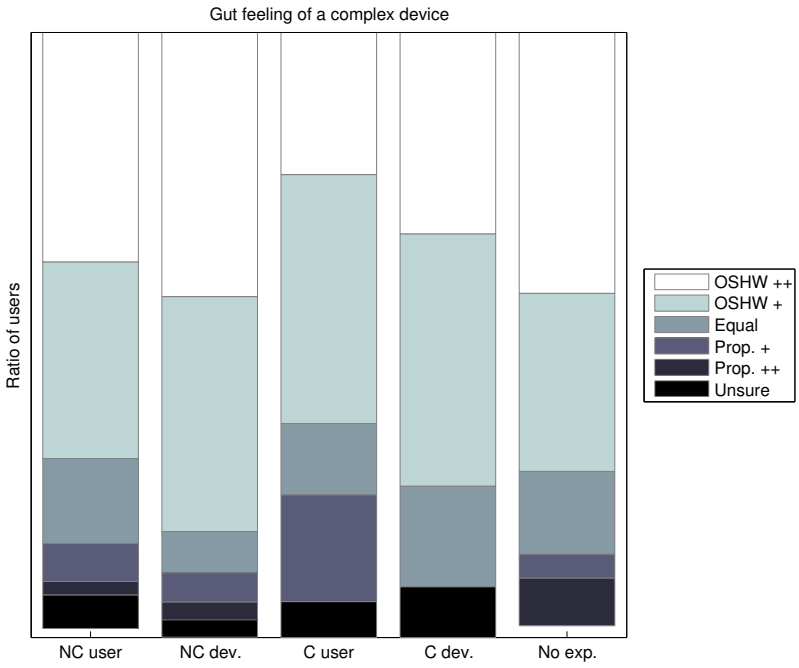


Figure 17: Response to gut feeling metric for complex devices.

4.5 IMPORTANCE OF OPEN SOURCE HARDWARE ATTRIBUTES

This section of the survey looks at the importance of some of the indirect effects and attributes of OSHW, divided into four groups. The question as formulated to the participants of the survey was;

In your experience, how much value do you place on the attributes of OSHW as below?

	Very important	Slightly important	Neither important nor unimportant	Somewhat unimportant	Completely unimportant
The ability to see the way a project works (electrical schematic, component choices, PCB layout, and so on)	☐	☐	☐	☐	☐
The ability to modify a project (such as altering the program code or changing the PCB layout)	☐	☐	☐	☐	☐
The ability to reuse parts of a project as a reference for own designs (such as re-using small amounts of code or duplicating schematic or PCB sections)	☐	☐	☐	☐	☐
The philosophical and ethical aspects of OSHW (the perceived net benefit to society by designs and their derivatives being available to the public)	☐	☐	☐	☐	☐

The responses to these attributes are shown in Figure 18 on page 46 through Figure 21 on page 48 and are presented in a stacked format. For the sake of brevity, the four categories are referred to as transparency, modification, re-use, and philosophy in the figures.

Some results that can be directly seen in these plots are;

- Nearly all of the four attributes are generally regarded as very important over all user groups. The ability to re-use parts of a design as a reference for another project was generally regarded as the least important, while the abilities of studying the way a

design works and performing modifications to a design were regarded as the most important.

- There is a clear difference between the commercial and non-commercial groups with regards to the philosophy of OSHW. The commercial users in particular stand out as being different, while the four other participant groups have very similar values. The group of commercial users found the philosophical aspects of OSHW to be of far less importance than the other four groups. Some 35% rated the philosophical aspects to be between "neither important nor unimportant" and "completely unimportant", while the average for the other groups is approximately 12.4%.

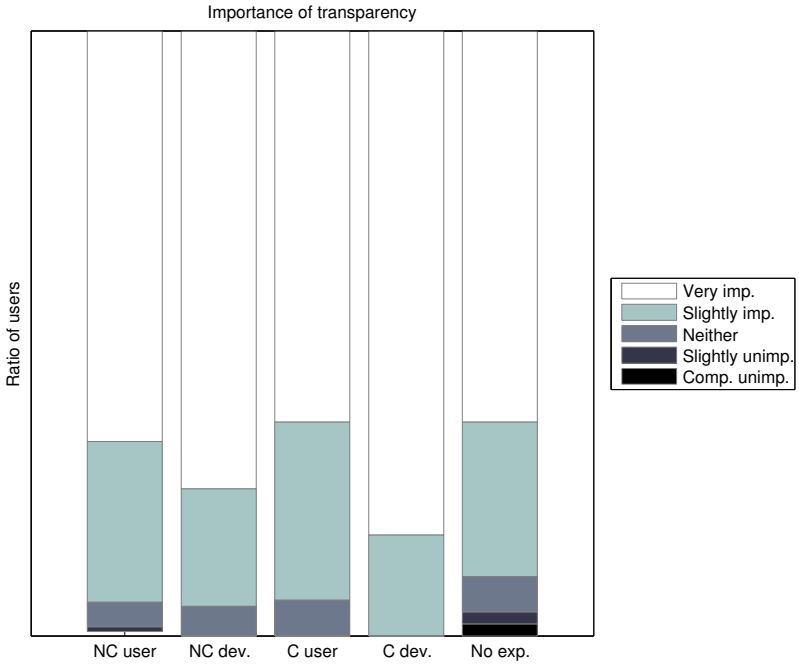


Figure 18: Response to the importance of transparency attribute.

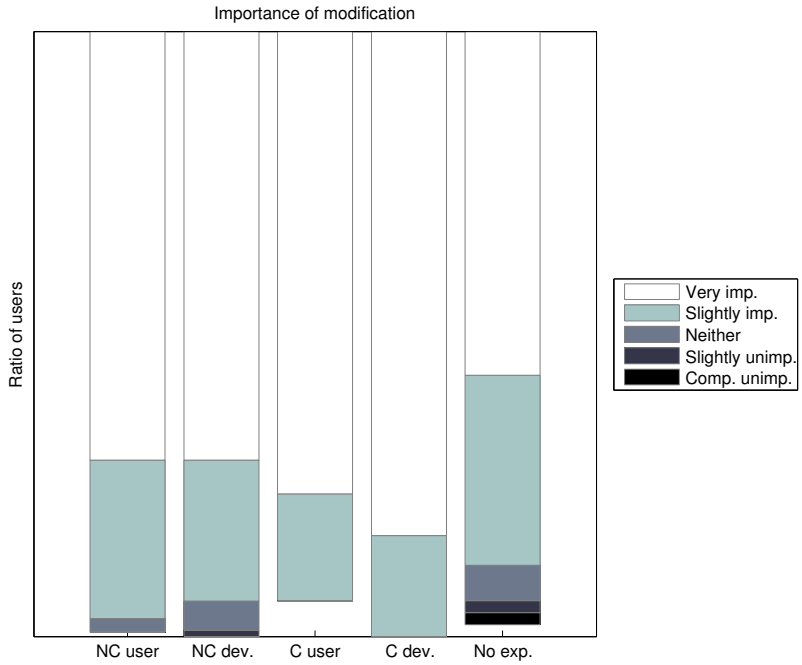


Figure 19: Response to the importance of the modification attribute.

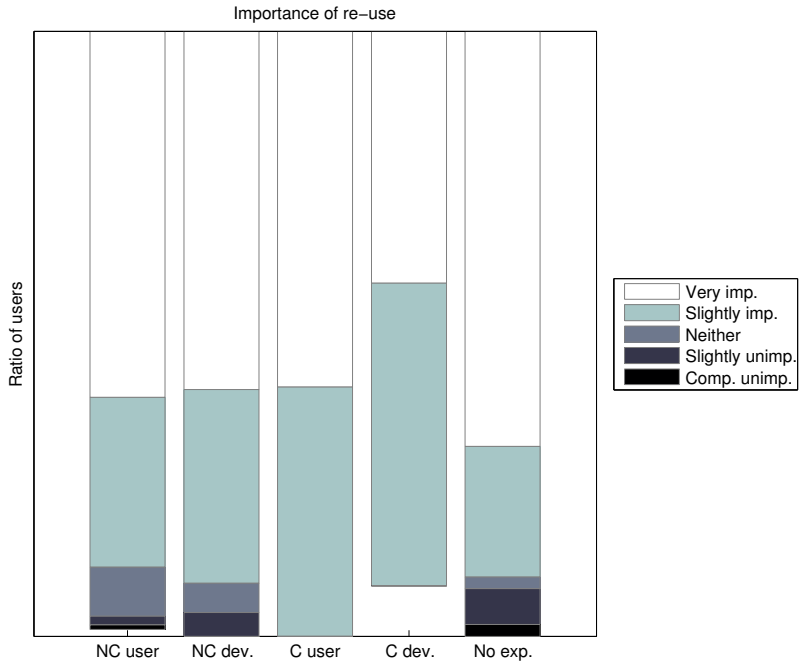


Figure 20: Responses to the importance of the re-use attribute.

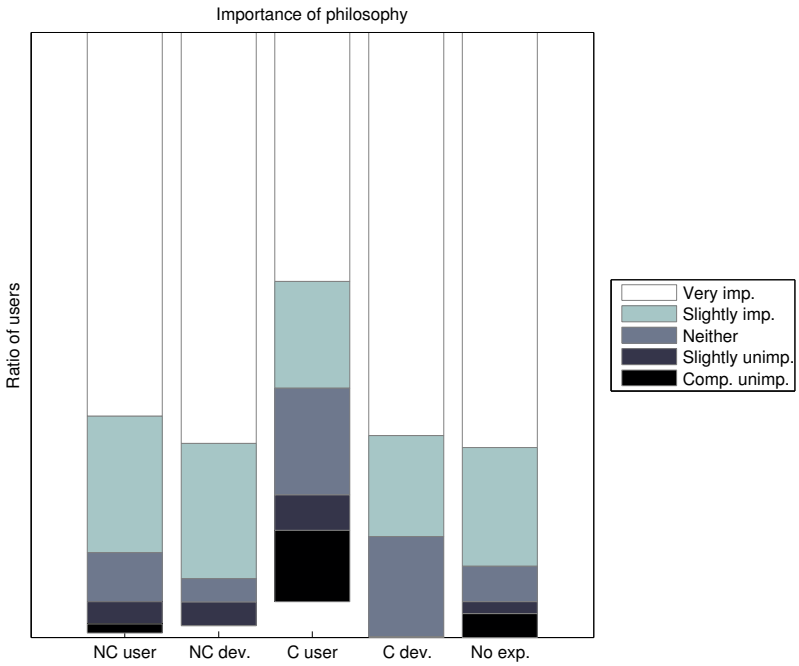


Figure 21: Response to the importance of the philosophy attribute.

4.6 PARTICIPANT DEMOGRAPHICS

This final section of the survey looks at the demographics of the participants. The first question posed to the participants was;

> What is your educational background?
> - Some high school
> - Finished high school
> - Some college/university (without any degree)
> - Finished bachelor's degree or equivalent
> - Finished master's degree or equivalent
> - Finished PhD or equivalent or higher

whose responses are shown in Figure 22 on page 49. Nearly all participants in all groups have at least some university or college education. The group of commercial users is particularly well educated, with over 80% holding a bachelor's degree or equivalent and some 35% with a master's degree.

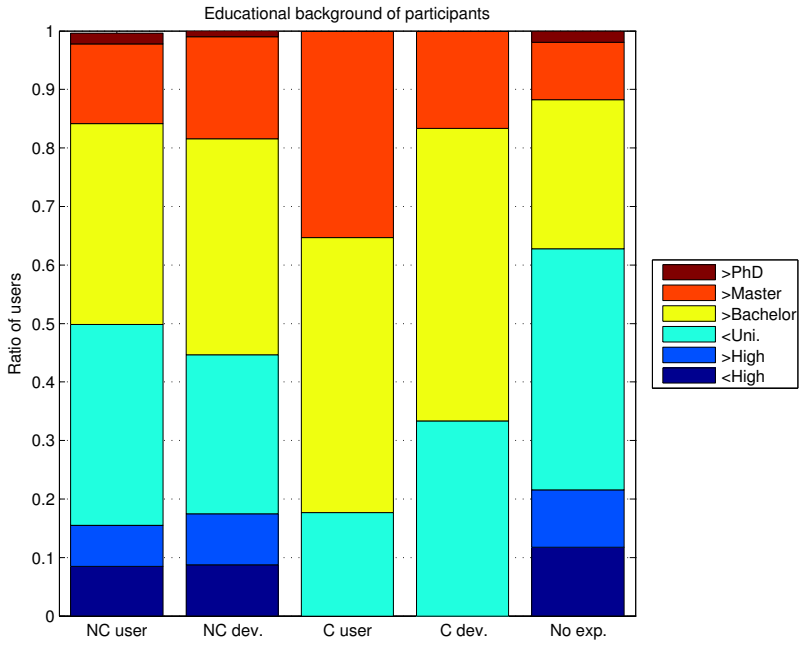


Figure 22: Survey response to "What is your educational background?".

The responses to the second question;

What is your age?
- Under 15
- 15 to 18
- 19 to 22
- 23 to 27
- 28 to 33
- 34 to 39
- 40 to 47
- 48 to 57
- 58 to 68
- Over 68

are shown in Figure 23 on page 51[5]. As can be seen, the age span in all groups is large — from 15-year-olds to those up to 68. Generally, the most represented group are individuals in the range of 23 to 47 years of age. Developers are generally older than users, and the individuals in the commercial groups are significantly older than individuals in the non-commercial groups; over half of the non-commercial users are under 27 years of age, while over half of the commercial developers are over 34 years of age.

The responses to the final question in this section;

Where are you from? (Choose the region you feel most accurately describes your background)
- Africa
- Asia
- Australia
- Europe
- North America
- South America

are shown in Figure 24 on page 51. The distribution of responses is very similar across all participant groups, with North America and Europe representing approximately 90% of the responses.

5 Note that the response choices are distributed logarithmically; this is intentional and done in order to increase the ability to discern small age differences for younger participants.

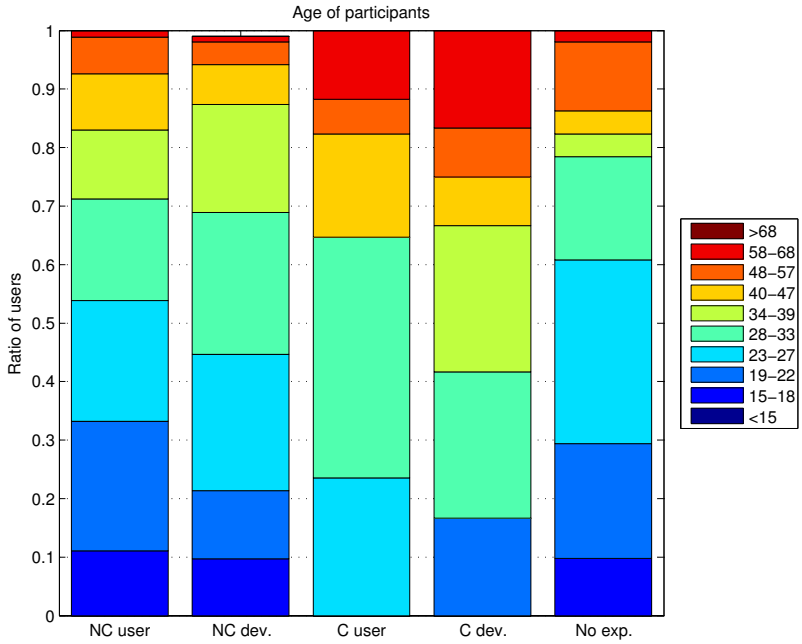


Figure 23: Survey response to "What is your age?".

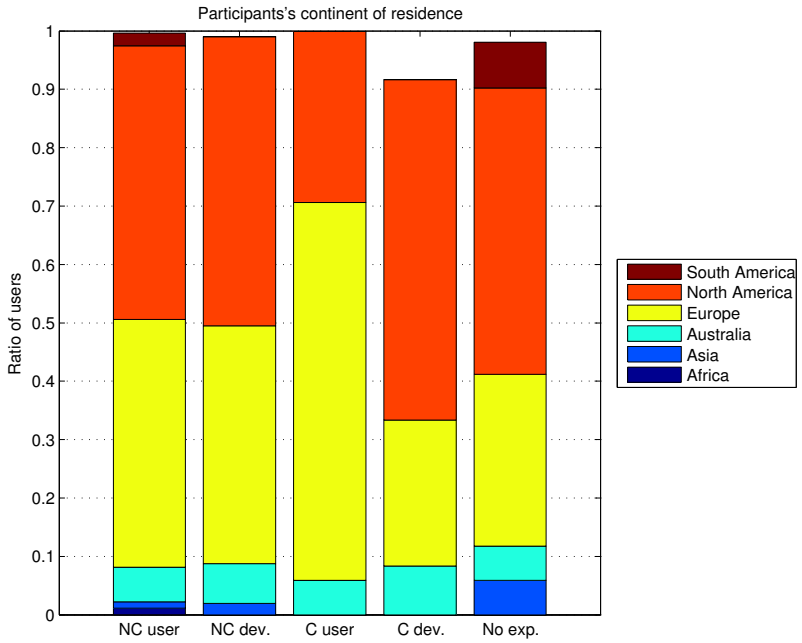


Figure 24: Survey response to "Where are you from?".

Part III

DISCUSSION & CONCLUSIONS

5 DISCUSSION

Combining existing research into OSS and OSHW (as brought up in Chapter 2) with the results found in the survey (as described in Section 3.1) and the interviews (as shown in Chapter A in the appendix) allows for shedding light on the research questions this report studies;

- Can embedded electronics companies thrive through the use and/or development of OSHW?
- Which business models are feasible for embedded electronics companies working with OSHW?
- What potential limitations may an OSHW work-flow apply to embedded electronics companies?
- What potential gains could an OSHW work-flow give to embedded electronics companies?

The content in this section will first examine the similarities and differences between preexisting research and the results found in the survey and interviews. Afterwards, the survey results and comments are analyzed in more detail, followed by a general analysis using all the data. Finally, the implications of the results are considered from the standpoint of the research questions.

5.1 ORIGINAL AND PREEXISTING RESEARCH AT A GLANCE

In general, both the interviewees and the survey participants who left comments highlighted similar aspects as those brought by academic research in Chapter 2. In particular, the philosophical attributes that are equally applicable to both OSS and OSHW were expressed and viewed in similar ways.

For example, the ability to study the source files of a device, one of the fundamentals of an open source work-flow, was widely regarded as being very important both in the survey as well as the interview with SparkFun Electronics. This was the most common comment left by the participants in the survey with 23 expressing this opinion (B.1.1). Additionally, 70% of the survey participants rated this as very important, while only 10% rated this as less than slightly important (as shown in Table 4 on page 97, transparency, very important). SparkFun Electronics highlights this in their interview with

> [...] *we published our source code, schematics and eagle and gerber files for our early products. We did this because we weren't experts yet, and we figured if someone had an idea on how to improve our designs, then it would be in the best interest of everyone if we could work together.*
>
> (Section A.1, Interview with Chelsea Moll from Spark-Fun Electronics [26])

This very closely matches what is said about OSS, where Weber [34, p. 16] writes; *"Property in open source is configured fundamentally around the right to distribute, not the right to exclude"*, and in a similar vein Krishnamurthy [23] writes; "[the community] *wants any interested developer to have access to the entire code so that the person can tinker with it to make improvements"*.

The largest difference between the original research into OSHW and existing research into OSS is, somewhat unsurprisingly, due to inherent practical differences between OSHW and OSS. The little academic research that exists studying OSHW (as described in Chapter 2) is largely in line with the results found.

Acosta [14, p.14] brings up attributes such as reliance on manufacturers for parts, difficulty in merging or updating to different versions, and often needing in-depth knowledge and expensive equipment for developing complex devices, attributes that have been expressed in the survey and in the interviews.

Comments from participants in the survey (shown in full in Section B.1) list similar attributes; highlighting a lack of standardization between devices making it difficult to use them together or merge them; a dependence on suppliers (often even a single supplier) of the physical parts used to construct the device; and weak IP protection, thereby allowing others to "clone" designs without releasing them under an OSHW license.

Additionally, many (though not all) of the reasons that Pololu brought up for not adopting OSHW (Section A.2 and Section A.3) are related to the same practical issues, such as requiring mechanical parts and expensive hardware and software for development.

On the other hand, a few of the other potentially problematic attributes discussed in Section 2.2 were not mentioned at all in the survey comments or the interviews, such as; the unavailability of automated updates, heterogeneous design software, and difficulty with version merging. However, with the relatively small sample size of the survey and interviews absence of evidence is not evidence of absence — it may be possible that these attributes are relevant but were simply not brought up in the comments or interviews.

In general, the results gathered in this report adds much-needed empirical data regarding the reported importance of OSHW among both its users and developers. This is useful both for general studies of OSHW, as well as for the research questions of this report.

5.2 ORIGINAL RESEARCH ANALYSIS

There are many aspects of the survey and interviews that are worth considering; ranging from the way OSHW is used, to the expressed value of the philosophical aspects of OSHW.

Many of the identified results in the survey are based on the differences between the different participant groups (described in Chapter 4). Briefly, the survey participants are grouped into five different categories based on their way of using or developing OSHW; noncommercial users, noncommercial developers, commercial users, commercial developers, and participants with no experience of OSHW.

It is important to keep in mind that neither level of bias in the participants nor the truthfulness of their responses to the survey can be determined. Hackaday is generally is favorable of OSHW, making it possible that an unknown contribution to the positive view of OSHW in the survey is due to a self-selection bias, though this is somewhat mitigated by comparing groups rather than studying the levels on their own. Additionally, it is possible that some or many of the participants selected answers that are not in line with their opinions as a form of humor, vandalism, boredom, or for no particular reason. However, the relatively small scale of the survey (posted in an enthusiast/hobbyist setting) in combination with no direct incentives to respond untruthfully (as the responses are anonymised and no compensation was given for responses) reduces the risk of this. Additionally, although not all of the participants chose to leave a comment, of those who did there were no obviously untrustworthy ones.

The results of the interviews with Pololu and SparkFun Electronics are brought up occasionally in this section, though the majority of the analysis is based on the survey results.

5.2.1 *Use and Development of Open Source Hardware*

As is stated in Section 4.2, use of OSHW is for the most part done directly or with relatively minor changes, both among developers and users. Additionally, commercial developers are more likely than non-commercial developers to use various parts of an existing design as a base when making something new. Commercial developers are also more likely than non-commercial developers to contribute or start a project to solve an immediate problem, **making it probable that commercial developers often use existing OSHW as a base when developing devices as a part of their scheduled work, rather than making contributions to existing hardware as is the case for non-commercial developers**.

Additionally, as can be seen in Figure 1 on page 31, most non-commercial users have also performed development work with the devices they have used, with the majority performing relatively mi-

nor changes. Additionally, as stated by five separate participants in the comments by the non-commercial users, the community of developers and users is regarded as very important. **It is likely that many of the minor changes made by the non-commercial user group are fed back into the original device**, leading to improvements by the users as well as the developers.

Though there is no statistical ground to stand on for a question-by-question basis for the development of OSHW as described in Section 4.3, some broad conclusions can be drawn. In general, it seems that **commercial developers are more focused on the end-goal of developing a specific device, while non-commercial developers are more likely to contribute to projects "for fun" or for educational reasons**. This is also in line with the previous analysis of how commercial developers use OSHW, lending credibility to this viewpoint. This is somewhat expressed in two of the three the comments received from commercial developers, who present a very pragmatic view of OSHW;

> [...] *On the positive side* [OSHW] *buys goodwill and exposure from the users but on the negative side you are giving away your labor for someone else to exploit. At the end of the day you have to feed yourself and family.*
>
> (See B.1.5, Commercial Developers)

and

> *Getting to the place of allowing everyone to have your design for free from the beginning is risky if you are trying to build a business from it.* [...] [OSHW] *relies on the idea that everyone [should] be able to use your idea immediately to create their own. If their idea is better, they can be more successful than you with that product even though the idea was originally yours.* [...] *This can cripple the newcomer if their idea is great but they don't get the word out (marketing).*
>
> (See B.1.5, Commercial Developers)

5.2.2 *Open Source Hardware compared to Proprietary Hardware*

In general OSHW devices are rated quite positively, with a very positive gut feeling for both simple and complex devices (as seen in Figure 11 on page 40 and Figure 17 on page 44). OSHW in both device categories fares best with the cost attribute and worst with the stability[1] and support[2] attributes, with functionality[3] and documentation[4] on a similar level. Complex OSHW devices were continuously rated more poorly than simple devices.

1 Defined as "How well engineered and thought-through [a] device is" in the survey.
2 Defined as "How easy it is to get help if something goes wrong" in the survey.
3 Defined as "What it can do" in the survey.
4 Defined as "How clearly described it is to use" in the survey.

5.2.2.1 *Stability and Testing Requirements*

One possible reason, in the author's opinion, for the stability attribute's poor rating is the large number of poorly thought-through do-it-yourself designs generally available on the Internet. Figure 25 on page 60 shows an example of two circuits found when searching for images related to "headphone amplifier circuit" and "ignition coil circuit" (these search terms chosen partly arbitrarily, and partly due to the large number of responses generated by these terms). Though other circuit schematics without as large flaws are found as well, it is difficult for an individual without a large degree of experience to be able to determine what constitutes a good design or a bad design. This is further exacerbated by widespread misinformation on public discussion forums and the like, where circuit schematics like those in Figure 25 on page 60 are publicized and used, leading to a form of cargo-cult development. This viewpoint is backed up by several comments left by participants, among others these two responses from non-commercial developers;

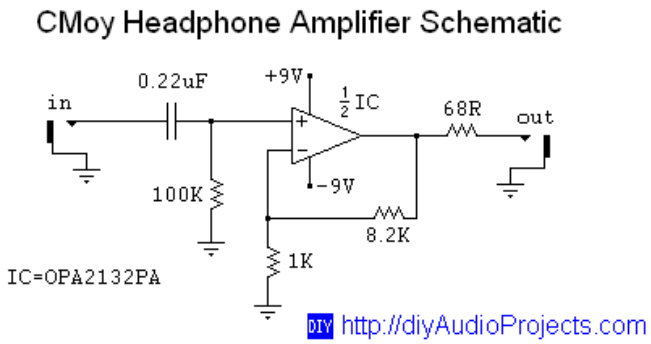
> [...] *Issues crop up that should've been addressed earlier. I regularly see incredibly poor pcb layouts lauded as great, short-sighted designs praised as commercial challengers, all only because they were released as open source. Releasing something as OSHW does not give one an excuse to release shoddy code and get praised.* [...]
>
> (See B.1.3, Noncommerical Developers)

> *OSHW is useful, but it seems* [to be] *only useful to those who possess a background of related education or to those willing to struggle to acquire the education. In the many instances I have been directly involved, and especially when trying to get others interested, the lack of plain clear information available to a layman is disappointing and discouraging.*
>
> (See B.1.3, Noncommerical Developers)

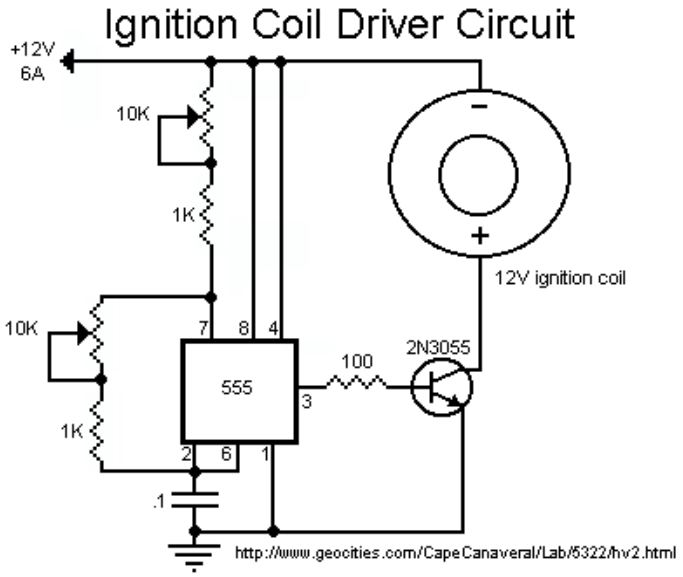
This issue is indirectly resolved for most proprietary devices, as there are legal regulations that require various safety, EMC, and Electromagnetic Interference (EMI) testing protocols to be passed before devices can be legally being sold. Though some of the flaws in the examples shown in Figure 25 on page 60 would not directly lead to failing these tests, it is probable that an engineer capable of developing a product that passes regulations would also develop a product without as many functionality flaws. (Of course, the existence of numerous safety-related product recalls[5] shows that this is not always the case.)

Though safety requirements would ideally only be a positive force, they are often regarded as a difficulty when developing OSHW de-

5 For example, the recall of 3 million notebook computer batteries due to fire hazards http://www.cpsc.gov/en/Recalls/2007/Sony-Recalls-Notebook-Computer-Batteries-Due-to-Previous-Fires/.



(a) Schematic for a headphone amplifier; lacking decoupling capacitors for the operational amplifier which may cause instability and/or worsened distortion; without a bandwidth limiting feedback capacitor across the gain-setting resistor, which increases the output noise at high frequencies (made doubly worse by selecting a relatively high-speed operational amplifier); and with an output resistor outside of the feedback loop, greatly reducing the damping factor (the control of the output) of the amplifier, increasing the total distortion and creating a varying output amplitude depending on the output impedance (which, for a loudspeaker, varies with frequency). *[http://diyaudioprojects.blogspot.se/2007/08/grado-ra1-headphone-amplifier.html]*



(b) Schematic for an ignition coil driver; missing decoupling capacitors for the timer, likely injecting a large amount of disturbance into the various parts of the device when the output switches; and without any protection for the power transistor when switching off, reducing the lifespan significantly or destroying the device completely on use due to avalanche breakdown caused by parasitic inductance in the ignition coil. *[http://www.oocities.org/capecanaveral/lab/5322/coildrv.htm]*

Figure 25: Examples of easily found flawed circuit schematics.

vices. Changes to hardware sold or distributed as a finished design often require expensive testing, even after small changes, which may be beyond the abilities of non-commercial developers (whom are significant group as they outnumber the commercial developers by 8.5 to 1 in the survey (Table 4 on page 97)). **As open source movements are generally oriented around a distributed community of developers, many whom are unpaid, compliance testing for turnkey** OSHW **devices may prove to be a problematic issue**. This is highlighted in the interview with SparkFun Electronics where some methods of avoiding compliance testing requirements are described;

> ***Testing to make sure new electronic devices work as function and pass legal requirements (EMI, safety, and so on) is typically regarded as expensive and time consuming. One of the fundamentals of OSHW is right to change or modify a circuit to become whatever you want it to be. As some testing must be done on every change to hardware, this places some limitations on the ability to (legally) change things freely. How do you at Sparkfun test and evaluate your devices before releasing them? How can a hacker use the right to make changes to a widget without paying the very large sums for testing that is traditionally viewed as a requirement?***
>
> ***Engineer Mike Hord:*** *The simple answer is, we avoid creating products that require a great deal of testing as much as possible. Most of what we make can rightly be considered a subassembly or a kit, and requires little or no testing.*
>
> *In most cases, a hacker making changes on their own, for their own use, doesn't have a great deal of regulatory onus on them. If they're making a derivative product to resell, then I would expect them to be aware of the laws regarding safety and EMC the same as they should be aware of, for instance, tax obligations. It's part of the cost of doing business.*
>
> ***Engineer Jordan McConnell:*** *We do a lot of testing for the functionality and quality of our boards, but not a lot of testing for 'legality' or safety. It's rare that one of our products needs FCC or other legal testing, and we often avoid creating these for that reason. Also, due to using low DC voltages, our boards are very safe compared to working with direct AC from a wall outlet. For hackers worried about whether or not they need FCC testing, Mike Hord wrote a FCC tutorial.*
>
> (See Section A.1 for the full interview transcript and `http://www.sparkfun.com/tutorials/398` for the tutorial mentioned by Jordan McConnell)

5.2.2.2 *Differences Between Participant Groups*

In general there is a clear difference between the commercial and non-commercial participant groups that extends to both the simple and complex device categories. Commercial users and developers were generally less favorable of OSHW devices and rated an increased ability to use proprietary equivalents, especially among functionality, documentation, and support (as seen in Figure 6 on page 37 through Figure 17 on page 44). However, the cost metric rating given by the commercial groups was even more favorable towards OSHW than the rating given by the noncommercial groups.

It is possible that some of the differences between the non-commercial and commercial groups are due to some very enthusiastic individuals in the non-commercial group, while the commercial groups may be generally more serious and pragmatic. Though it is difficult to determine the validity of this statement, some of the comments left by the noncommercial groups were extremely supportive of OSHW and arguably unrealistic. Excerpts from some of these comments include;

> [...] *So by designing and manufacturing our own cellular network hardware we can set up* [the United States] *and Canada* [with a high-bandwidth,] *very cheap network with unlimited internet and calling! This will change the world!!! Then expand into Europe and make ONE global cellular network! With unlimited everything! I would LOVE to do this project!*
>
> (See B.1.2, Noncommercial Users)

> *OSHW typically has better documentation because the community won't allow undocumented progress.*
>
> (See B.1.3, Noncommerical Developers)

This is in contrast to the majority of comments from the non-commercial groups and all of the received comments form the commercial groups, which are generally aligned with the numerical results of the survey, such as;

> [...] *Getting to the place of allowing everyone to have your design for free from the beginning is risky if you are trying to build a business from it. It takes a fair leap of faith; it's important to remember that the patent system is actually closely related to the Open movement. Patents must list prior art, and after the owner's right expire anyone is free to use the design. [OSHW] relies on the idea that everyone [should] be able to use your idea immediately to create their own. If their idea is better, they can be more successful than you with that product even though the idea was originally yours.*
>
> *This can cripple the newcomer if their idea is great but they don't get the word out (marketing). But it certainly pushes the*

> *state of the art more quickly in what could be argued a more socially beneficial manner.*
>
> (See B.1.5, Commercial Developers)

> *The "boring" parts of the development process (documentation, debugging, polish) can be hard to motivate people to do and are often left lacking.*
>
> (Said by 8 noncommercial users and 1 noncommercial developer, see B.1.1, Meta-comments)

It is the belief of the author that there are higher number of outliers in the collected data from the non-commercial groups which tend to bias the results of the analysis. Most of these outliers are believed to be strongly in favor of OSHW, and may give an unrealistic image of what is feasible.

The group of commercial users in particular was very critical of the documentation and support of OSHW devices. This is reflected in the comments from the survey, which paint an image of a very pragmatic commercial user. **Though the commercial users see many positive aspects of OSHW , the determining factor is largely due to the total cost of using the device, including set-up time (which may be long in the case of poor documentation)**. This statement is backed up by the received comments, as two of the four participants from the commercial user group write;

> *In a professional environment, where time is money, my employer is more prepared to buy a relatively expensive solution that works from the start, than having me spending hours getting OSHW working just right. Buying a proprietary, well supported product is often the quickest way forward.*
>
> *(See B.1.4, Commercial Users)*

> [...] *The flipside is that OSHW/SW documentation is often absolutely awful or non-existent, and in a commercial operation you cannot afford to spend weeks trying to get something to work - it's worth paying the premium for something closed-source that will work out of the box and is documented.*
>
> (See B.1.4, Commercial Users)

The group of commercial developers have a viewpoint similar to that of the commercial users, giving similar ratings in the survey and with a pragmatic tone in their comments. However, **the commercial developers describe the very weak licensing structures available (in contrast to OSS) as one of the major drawbacks to developing OSHW**. A poignant example of this is the comment;

> *OSHW is great for the community but has pros and cons for the developers. On the positive side it buys goodwill and exposure from the users but on the negative side you are giving*

> *away your labor for someone else to exploit. At the end of the day you have to feed yourself and family.*
> (See B.1.5, Commercial Developers)

Interestingly, both developer groups reported a reduced ability to use proprietary equivalents as compared to the user groups. This is possibly due to a selection effect, where individuals who see a lack of devices in some region decide to develop an OSHW device filling said gap.

The group of users without any previous experience very often gave ratings similar to those of the non-commercial users. It is possible that this group has previous knowledge from other open source movements, such as OSS.

5.2.3 *Importance of Open Source Hardware Attributes*

Nearly all of the four attributes of OSHW brought up in the survey; transparency[6], re-use[7], modification[8], and philosophy[9], were considered important over all four participant groups with experience of OSHW.

Transparency and modification were generally viewed as the most important attributes among all groups, with re-use viewed as less important among all groups. **Commercial users rated philosophy as relatively unimportant compared to other groups, which strengthens the previous hypothesis of the commercial users as being primarily pragmaticists.** Some 35% of the commercial users rated the philosophical aspects to be between "neither important nor unimportant" and "completely unimportant", while the average for the other groups is approximately 12%; nearly a three-fold increase.

5.2.4 *Participant Demographics*

The demographics of the participants offers some hints to the reasons for their responses in the survey. The commercial group of users is more highly educated and older than the non-commercial groups. The relative youth of the non-commercial user group, with a median age of 23-27 and 33% below 22 years of age may be a reason for the lower educational level among this group — a large portion of this

6 Defined as "The ability to see the way a project works (electrical schematic, component choices, PCB layout, and so on) " in the survey.

7 Defined as "The ability to reuse parts of a project as a reference for own designs (such as re-using small amounts of code or duplicating schematic or PCB sections)" in the survey.

8 Defined as "The ability to modify a project (such as altering the program code or changing the PCB layout)" in the survey.

9 Defined as "The philosophical and ethical aspects of OSHW (the perceived net benefit to society by designs and their derivatives being available to the public)" in the survey.

group is simply not old enough to have finished their education. With this in mind, it is not surprising that many of the non-commercial users use OSHW in a hobbyist setting; **it is possible that a sizable portion of the non-commercial users are students and use OSHW in their free time, possibly in a way that is related to their studies.**

From a geographical perspective, the absolute dominance of Europe and North America, representing approximately 90% of the responses, is likely due to hackaday (the primary source of survey participants) being far more well-known in these regions than the others.

5.2.5 *General Analysis*

In general, participants of the survey found the documentation of OSHW projects to be relatively poor. Non-commercial users often viewed this as an annoyance, while commercial users both rated documentation more poorly in the survey and described the poor level of (among others) documentation to be a reason to use a proprietary equivalent. As one commercial user wrote;

> [...] *The flipside is that OSHW/SW documentation is often absolutely awful or non-existent, and in a commercial operation you cannot afford to spend weeks trying to get something to work - it's worth paying the premium for something closed-source that will work out of the box and is documented.*
>
> (See B.1.4, Commercial Users)

In spite of the poor documentation available for projects, the most common sentiment in all of the user groups is;

> *The ability to study the source files of OSHW devices (such as a schematic or board layout file) is one of the main benefits of OSHW. This simplifies debugging and increases the useful lifespan of the device as compared to a proprietary device, where the end of support from the developing company may effectively be a death sentence for the device.*
>
> (Said by 16 noncommercial users, 4 noncommercial developers, and 3 Inexperienced users. See B.1.1, Meta-comments)

It is the opinion of the author that there is the potential for usefulness of OSHW documentation to dramatically exceed the usefulness of the typical proprietary device documentation of today. Thorough documentation allows both for a casual user, uninterested in the inner workings of the device, to get started and simply use the device, while at the same time allowing others whom wish to repair, modify, or otherwise alter the device to do so at their leisure. The importance of this is backed up by the very high rating of transparency among the survey participants, with an average of 95% of all participants rating this as very important or somewhat important (of which 75% rate this as very important).

Replacement of Components

Tektronix will supply replacement components at current net prices. However, since most of the components are standard electronic and radio parts we suggest you get them from your local dealer if you can. Be sure to consult your instruction manual first to see what tolerances are required.

We specially select some of the components, whose values must fall within prescribed limits, by sorting through our regular stocks. The components so selected will have standard RETMA color-code marks showing the values and tolerances of the stock they were selected from, but they will not in general be replaceable from dealers stocks.

Vertical-Amplifier Adjustments

1. Gain Adjustment

The main unit sensitivity is standardized at 0.1 volt per centimeter so that the calibrated gain controls of all plug-in units will be correct. Since this sensitivity is affected by the crt accelerating voltage, first check the voltage at the left end of the ceramic terminal strip near the crt socket. Adjust to —1350 volts if necessary with the H. V. ADJ. control at the right side of the instrument near the rear.

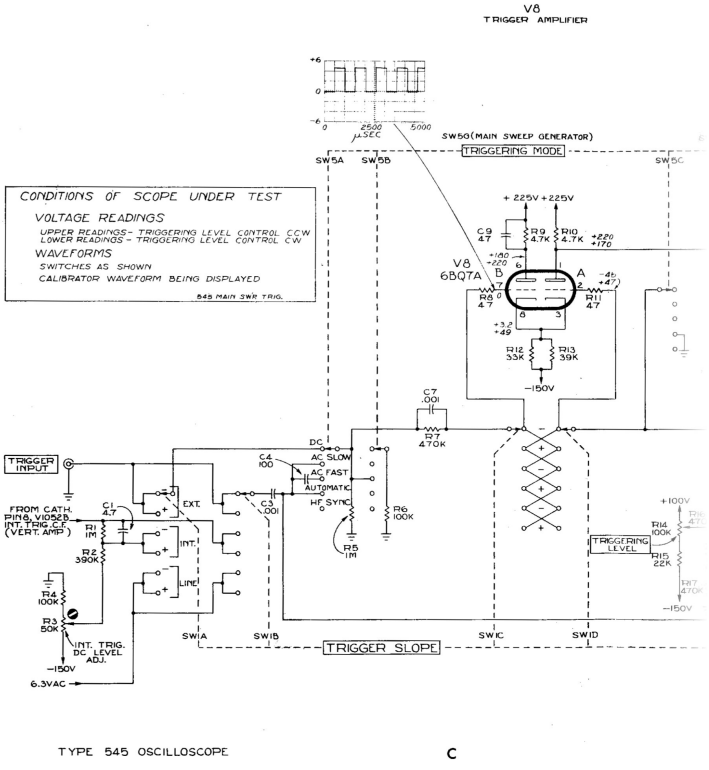
Now connect a voltmeter between pins 1 and 3 on the interconnecting plug. Position the trace two centimeters above and below center with the **VERTICAL POSITION** control and adjust the main amplifier GAIN ADJ. control, R1027, until there is a total voltage change of

(a) Functional description excerpt from the 545A.

MAIN-SWEEP TRIGGER

							Order Parts by Number
				Capacitors			
C1	4.7 μμf	Cer.	Fixed	500 v	±1 μμf		281501
C3	.001 μf	PT	Fixed	600 v	20%		285501
C4	100 μμf	Cer.	Fixed	350 v	20%		281523
C7	.001 μf	Cer.	Fixed	500 v	GMV		283000
C9	47 μμf	Cer.	Fixed	500 v	20%		281518
C17	.001 μf	Cer.	Fixed	500 v	GMV		283000
C20	.01 μf	PT	Fixed	400 v	20%		285510
C28	.005 μf	Cer.	Fixed	500 v	GMV		283001
C34	22 μμf	Cer.	Fixed	500 v	20%		281510
				Resistors			
R1	1 meg	½ w	Fixed	Comp.	5%		301105
R2	390 k	½ w	Fixed	Comp.	5%		301394
R3	50 k	2 w	Var.	Comp.	20%	Int. Trig. DC Level Adj.	311023
R4	100 k	½ w	Fixed	Comp.	10%		302104
R5	1 meg	½ w	Fixed	Comp.	10%		302105
R6	100 k	½ w	Fixed	Comp.	10%		302104
R7	470 k	½ w	Fixed	Comp.	10%		302474
R8	47 Ω	½ w	Fixed	Comp.	10%		302470

(b) Bill of materials except from the 545A.



(c) Schematic diagram excerpt from the 545A.

Figure 26: Bill of materials, functional description, and full schematic diagrams presented in the service manual for the Tektronix 545A, a state of the art cathode-ray vacuum-tube powered oscilloscope from 1957[13].

Documentation that covers the internal workings of devices for repair was relatively common in the past, Figure 26 on page 66 shows documentation for the Tektronix 545A; a cathode-ray vacuum-tube based oscilloscope from 1957 which shipped with original service manuals fully describing both how to operate the device as well as the entire internal functionality, with a bill of materials and schematic diagrams in order for the user to replace parts as needed [13]. Though this is to some extent due to the very high cost of the device (around half a year's salary for an engineer) and the low lifespan of vacuum tubes (making replacements a necessity) this is a stark contrast to the documentation of most proprietary devices released today.

Both SparkFun Electronics and Pololu have expressed the usefulness and importance of distributing the internal workings of their devices, though they differ with regards to their view of OSHW. As SparkFun Electronics writes;

> ***According to Wikipedia, Sparkfun was one of the contributors in drafting the first Open Source Hardware (OSHW) definition. How was it that open source hardware became such a key part of who you are?***
>
> *The concept of an open exchange of ideas for the good of the community makes very clear sense to us. In fact, our company motto, which sums up a lot of our motivation and convictions - is "sharing ingenuity." In the beginning, before the open source movement was organized enough to be called a movement, we published our source code, schematics and eagle and gerber files for our early products. We did this because we weren't experts yet, and we figured if someone had an idea on how to improve our designs, then it would be in the best interest of everyone if we could work together. We still don't think we're experts, and we still believe that working together yields better results, and now we're happy to be part of a recognized, burgeoning open source community.*
>
> *Open source makes sense to us from both a practical and ethical standpoint; we want to be more useful to the customer than not, and being forthcoming with all of our materials seems a simple and obvious thing. And, as it turns out, it makes amazing business sense, too. We've never had to worry about IP or patents, or, for the most part, lawyers. We just want to focus on what we love to do: build stuff with our toys and sharing the love of electronics with others. Being open source allowed us to be closer to that ideal, and less what might be considered "business norm." And it turns out that there are a lot of people out there that feel as we do.*
>
> (See Section A.1 for the full interview transcript)

Some of these thoughts are expressed by Pololu as well, who say;

> ***Though your products are not released under an open source hardware (OSHW) license, many of your devices are released with a schematic (which is not the norm for proprietary products). Why have you chosen to do this? Why do you believe that this is a good level of openness?***
>
> *Releasing schematics and otherwise explaining how our products work are good for documentation and further our goal of helping those wanting to learn about electronics. That therefore helps the end customer in a direct way. I suspect that full access to source files, on the other hand, are unlikely to be helpful to most customers and disproportionally help just those who would make knock-offs without contributing anything.*
>
> (See Section A.2 for the full interview transcript)

Though the viewpoints Pololu and SparkFun Electronics have differ to a large extent, which is reflected to a large degree in the choice of adopting an OSHW work-flow, both are of the opinion that documentation beyond basic fundamentals needed to operate the device is useful.

It is the opinion of the author that effective documentation for OSHW (as well as proprietary devices to some extent) should cover both the fundamentals for setting up and using a device[10] as well as a more detailed description[11] of how the device works, in order for individuals to repair, analyze, or modify the device. Chapter C in the appendix shows an example of what the author regards as effective documentation for an OSHW device[12].

By first describing the fundamentals needed for solely configuring and using the device, an individual not interested in the inner workings of the device can get started as quickly as possible. Following sections show the electrical schematic, PCB design, and a functional description of the hardware and software to allow for others to understand the inner workings and modify the device to suit their needs. Though many users may not be interested or even understand the inner workings of the device, many of these users still find value in these details being present. This is repeatedly brought up in the collected comments from the survey as well as the interviews. 16 noncommercial users and 3 inexperienced users (as well as 4 noncommercial developers) bring this attribute up, as do SparkFun Electronics (and to some extent Pololu), as seen in the above excepts.

Claims of devices that "just work" and being simple to use no longer hold when they eventually fail or are artificially limited to a small region of use. By distributing what is needed to repair or

10 Such as "connect cable A to point B, turn dial C to location D".

11 Such as circuit diagrams, functional descriptions, and so on.

12 A simple OSHW device developed by the author for temperature regulation by controlling a heater, a cooling fan, or almost any other heating/cooling element.

improve the device, simplicity of use can be extended to simplicity of repair and simplicity of improvement.

5.3 RESEARCH IMPLICATIONS FOR COMPANIES

In this section, the results of the prior analysis of the survey will be placed in the context of the research questions for this report;

- Can embedded electronics companies thrive through the use and/or development of OSHW?
- Which business models are feasible for embedded electronics companies working with OSHW?
- What potential limitations may an OSHW work-flow apply to embedded electronics companies?
- What potential gains could an OSHW work-flow give to embedded electronics companies?

In practice, there is a very large difference between using OSHW and developing OSHW as a part of commercial activities. Each of these two different ways of interacting with OSHW will be studied separately in 5.3.1 and 5.3.2.

5.3.1 *Using Open Source Hardware in Businesses*

Generally, use of OSHW is a smaller step than development of OSHW as it is more easily tested and usually requires a far smaller investment and "leap of faith". Though much of the research performed in this report is not entirely relevant when determining the applicability of using OSHW in businesses, there are parts that remain very applicable.

The group of commercial users in the survey are a good reference point for studying the actual usefulness of OSHW use in businesses. As was discussed in 5.2.2.2, commercial users generally rated the level of documentation and support of OSHW devices far more poorly than the other groups, while the cost attribute was rated more positively. These values, in combination with the received comments from commercial users (reproduced here in entirety, also shown in B.1.4) give a starting point for discussion of the research questions;

> *The hacker community isn't driven by a commercial model, so it's possible to find unique devices unserved by commercial means. It does mean a somewhat steeper learning curve, but the "price of entry" for virtually any technology nowadays includes that. Additionally, the OSHW community is where the future's engineers are being motivated and trained RIGHT NOW. These kids amaze me.*

> *IP can [be] an issue with OSHW if the licencing strategy isn't well thought out. The fact that you are not tied down a licence restriction is really useful (I.E. the Libre side of OSHW).*

> *At the basic end, OSHW is generally more likely to provide a very specific solution to any given problem where a large commercial venture would not be able to justify the effort of making multiple very specific devices.*
>
> *At the complex end, OSHW is adaptable to specific needs where a commercial solution relies on it being worth their while doing the R&D to cover your use-case. As an example, we previously used a Connii I2C tester which is quite expensive, now we have a BusPirate which is far more flexible and we can bend it to our needs.*
>
> *The flipside is that OSHW/SW documentation is often absolutely awful or non-existent, and in a commercial operation you cannot afford to spend weeks trying to get something to work - it's worth paying the premium for something closed-source that will work out of the box and is documented.*

> *In a professional environment, where time is money, my employer is more prepared to buy a relatively expensive solution that works from the start, than having me spending hours getting OSHW working just right. Buying a proprietary, well supported product is often the quickest way forward.*

These comments closely reflect the collected data in the survey, lending credit to the ability of these comments to act as a fair sampling of the survey responses.

The potential gains of adopting the use of OSHW in a commercial setting are primarily (based on the data collected in the above comments and in previous sections of the discussion section);

- Greatly increased flexibility of use due to all the design files being publicly available;
- The ability to incrementally or dramatically modify the design as needed in any physically realizable way;
- An improved capability of avoiding obsolescence as there are no artificial locks to single suppliers[13];

13 It remains possible for a design to require a specific device supplied by a single supplier, though in many cases it may be possible to find an equivalent replacement. Regardless of this, there is no requirement to acquire the device from the original developer, removing one source of a single supplier dependence.

- The possibility for finding specific solutions which would not be profitable in traditional commercial development;
- Very low purchasing cost (as the device can be manufactured anywhere);
- The ability to analyze and check for poor design choices beforehand;
- The relatively high prevalence of a supporting community, often developing without economic incentives;
- No licensing costs[14] and very limited IP regulation, effectively allowing most actions, such as making a closed-source derivative version.

Similarly, potential limitations of the use of OSHW in a commercial setting include;

- (Often) poor documentation and support of designs, which may not be noticed at first;
- A reported increase in time required to set up or use the device, often due to the poor documentation, leading to increased total costs;
- The prevalence of poor designs and (often) lack of certification;
- Reluctance in adopting OSHW due to lack of knowledge of quality and suitability.

Unfortunately, the answer to whether embedded electronics companies can use open source hardware effectively is; "it depends". From what has been seen in the survey, both ratings and comments, if there exists some reasonably well documented device OSHW **use would be very beneficial as;**

- **A tool in a development setting; where direct use or use with minor in-house changes would fill a gap or be significantly less expensive than traditional commercial products.**
- **A (part of a) product; where the device can be modified to create a new product (not necessarily an OSHW product) or integrated into existing product development, reducing development time and costs.**

On the other hand, **sometimes using OSHW devices does not really make sense, such as;**

- **Poorly documented or designed devices,**

14 Assuming an OSHW license that fulfills the OSI license requirements [2].

- **Situations where traditional devices are available at low cost, that require no modification for use and where lock-in effects are minor,**
- **Situations where the knowledge needed to evaluate if a design is good or not does not exist within the company and misjudging design quality would be very costly.**

Of course, there are countless other scenarios where the use of OSHW is a good business choice, and many others where it is a poor choice. In practice, the evaluation must be done on a case-by-case basis and is not particularly dependent on any specific business model. Nevertheless, the collected data points to OSHW use having a strong potential of being useful in a business context.

5.3.2 *Developing Open Source Hardware in Businesses*

Developing OSHW as a part of the business conducted by a company is a more daring and potentially risk-filled endeavor than simply using OSHW, though not without its potential benefits as well.

In order to draw conclusions on this topic the analysis of the numerical results and the comments from the survey, together with relevant material from the interviews, will be used to generate a list of conclusions regarding OSHW development in a company. This is followed up with a discussion of some of the attributes of OSHW development that may be difficult to implement. Finally, a few examples of business models or situations where OSHW development makes sense is listed.

5.3.2.1 *Applicable Findings in Original Research*

Comments received from existing commercial developers of OSHW (also shown in B.1.5) give a starting point for discussing the various benefits and drawbacks of OSHW development;

> *OSHW is great for the community but has pros and cons for the developers. On the positive side it buys goodwill and exposure from the users but on the negative side you are giving away your labor for someone else to exploit. At the end of the day you have to feed yourself and family.*

> *I think Open Source and Open Hardware is the only way to go. It is the most efficient use of people's time and resources. The ability to accelerate innovation is so great that I think companies that embrace open source can leverage the crowd to be able to iterate and innovate quicker and cheaper than any other*

> *development path. If companies lose their ego and just make good products that are open source people will still stick with* [their] *brand because they are the originators* [of the device, developed] *it first and know why things work. However, I think there needs to be a new type of licences that* [is] *essentially open source but* [gives] *protections from someone copying it whole-sale and undercutting the originators. There needs to be a way to ensure open hardware is being used in the spirit of open hardware.*

> *The question of OSHW in a commercial context is whether it can actually make a company viable. While there are a very very few companies that have made it work, there are those, like Makerbot, that have not. Makerbot likely wrapped up the Replicator 2 at the insistence of those funding them. While places like Adafruit and DIY Drones publish their designs, their commercial viability is in the little impetus the buyer has to go out and wait for their own board to be produced and sent back to them, or in soldering SMD components, or in having to buy the needed components when they could instead just buy the kit.*
>
> *Getting to the place of allowing everyone to have your design for free from the beginning is risky if you are trying to build a business from it. It takes a fair leap of faith; it's important to remember that the patent system is actually closely related to the Open movement. Patents must list prior art, and after the owner's right expire anyone is free to use the design.* [OSHW] *relies on the idea that everyone* [should] *be able to use your idea immediately to create their own. If their idea is better, they can be more successful than you with that product even though the idea was originally yours.*
>
> *This can cripple the newcomer if their idea is great but they don't get the word out (marketing). But it certainly pushes the state of the art more quickly in what could be argued a more socially beneficial manner.*

These comments reflect the analysis of the survey in previous sections, lending credit to the ability of these comments to act as a fair sampling of the survey responses.

The interviews with SparkFun and Pololu give very different, though both interesting, viewpoints of the usefulness of OSHW. However, it is very difficult to generalize large parts of the interviews as the content is heavily based on the specific business models used by the interviewees. Two relevant excerpts from the interviews follow, beginning with an excerpt from the interview with SparkFun Electronics;

> ***According to Wikipedia, Sparkfun was one of the contributors in drafting the first Open Source Hardware (OSHW)***

> ***definition. How was it that open source hardware became such a key part of who you are?***
>
> *The concept of an open exchange of ideas for the good of the community makes very clear sense to us. In fact, our company motto, which sums up a lot of our motivation and convictions - is "sharing ingenuity." In the beginning, before the open source movement was organized enough to be called a movement, we published our source code, schematics and eagle and gerber files for our early products. We did this because we weren't experts yet, and we figured if someone had an idea on how to improve our designs, then it would be in the best interest of everyone if we could work together. We still don't think we're experts, and we still believe that working together yields better results, and now we're happy to be part of a recognized, burgeoning open source community.*
>
> *Open source makes sense to us from both a practical and ethical standpoint; we want to be more useful to the customer than not, and being forthcoming with all of our materials seems a simple and obvious thing. And, as it turns out, it makes amazing business sense, too. We've never had to worry about IP or patents, or, for the most part, lawyers. We just want to focus on what we love to do: build stuff with our toys and sharing the love of electronics with others. Being open source allowed us to be closer to that ideal, and less what might be considered "business norm." And it turns out that there are a lot of people out there that feel as we do.*
>
> (See Section A.1, Interview with Chelsea Moll from SparkFun Electronics [26])

This contrasts greatly with a response from Pololu, who writes;

> ***You're in the maker world, but you do not release any of your products as open-source hardware, why?***
>
> *There are many reasons, but they mostly boil down to not thinking it's worth it from a business perspective. I am somewhat skeptical of companies pushing OSHW, so I think it would be disingenuous to use what I think is a gimmick to market a product. For instance, the 3pi robot I mentioned earlier has several custom mechanical parts, a patented circuit, and is done in Altium Designer, which costs over $5k per seat. I realize that some schools might have site licenses for Altium, but for the typical user, releasing the source documents would not mean much. We already provide schematics and explanations of how the thing works, the design is fairly well validated, and it's not a particularly useful starting point for an improved design (e.g. one with a better processor), so the main entities actually getting something out of our releasing the source would be those*

trying to make knock-offs, and I'm fine with not giving them a shortcut.

(See Section A.3, Maker Media's interview with Jan Maléšek [32])

The vast number of different ways that OSHW development can be performed makes it impossible to draw very broad positive or negative conclusions. Additionally, only so much can be backed up by the survey and the interviews without delving into completely unsubstantiated discussion. What can be substantiated from the survey results follows;

- Improvements to an OSHW device from non-commercial users are likely often fed back into the original device, where changes are often made as a hobby, as part of an education, or for pleasure.
- It is likely that many of the non-commercial users are students and use OSHW as a part of their education.
- Commercial users of OSHW are very pragmatic and the total cost of adoption is important. OSHW devices are usually regarded as having poor documentation and support.
- There is the potential for OSHW documentation to be regarded as very good; many users stated that the ability to study the inner workings of a device is very important both in the numerical part of the survey and in the comment section.
- Existing commercial developers of OSHW primarily use existing devices as a base for developing a new device rather than incremental or minor changes.
- In general, all survey groups rated OSHW devices highly, making it likely that among audiences similar to that of the survey the added value of a device being OSHW is high.
- It is possible that OSHW devices can be very useful in a technical research setting; seven of the survey participants (approximately 1.5%) reported an educational background of holding a PhD or higher. The inherent flexibility of OSHW may find many applications in research, partly due to the high level of customization that is required in novel research and partly due to the level of collaboration that is common between different teams.

5.3.2.2 *Potential Pitfalls when Implementing OSHW Development*

Previous research into OSHW (as described in Chapter 2), comments from the survey, and interview responses are generally in line and support two primary difficulties in commercially developing OSHW.

The foremost issue, brought up by Pololu, the survey comments, and Acosta [14], concerns the licensing options available for OSHW[15]. These licenses are often very limited, nearly placing the device in the public domain and allowing free use of the non-copyrightable parts of a device. This is primarily due to the lack of legal protection for hardware that serves the equivalent of copyright for software (which allows for the construction of recursive licenses such as the GNU GPL [9]). The closest existing license, the TAPR-OHL [3], primarily gives protection for patented parts of devices; thereby requiring far more effort and financial resources for protection. Though patents are typically within the financial reach of a company, this is primarily applicable for companies developing few complex devices, rather than SparkFun Electronics which develops very many devices.

The risk of clones in OSHW is brought up in the comment section of the survey; where a device is copied by a third party and then sold at a very low cost, competing with the original developer whom must recuperate development costs in order to remain profitable. This behavior is typically not seen in the OSS world, as OSS is often distributed at zero cost, thereby removing any cloning incentive (income being most often generated through other channels, such as support or consulting). Branding, trademarks, manufacturing quality, continuously updating devices, and developing new devices are methods that are often used by the developer to maintain sales, according to SparkFun Electronics (see Section A.1) and Thompson [30].

Finally, compliance testing of OSHW devices is an aspect that may be difficult to solve for business models where community improvements are integrated into the main device. SparkFun Electronics writes that for most of the devices they sell expensive testing is not required, such as kits or sub-assemblies which do not require any testing. The Arduino group [5] has developed relatively few main devices that are certified and do not change rapidly, while the community instead focuses on developing separate hardware that attaches to the main Arduino device, expanding functionality without requiring the main device to redo compliance testing. (Of course, the add-in boards themselves must undergo some certification process before they can be sold, though this does not directly influence the original developers).

5.3.2.3 *Applicable Business Models for OSHW Development*

In the end, the ability to develop OSHW in a commercial setting largely depends on the business model employed by the company and the purpose of choosing to develop OSHW. As was described in 2.1.1, there are business models that do work for OSHW development. Acosta

15 Though this is also brought up in the interview with SparkFun Electronics, this is not regarded as a negative attribute in the interview.

[14] describes several applicable business models for OSHW development;

SELLING SUPPORT, SERVICE, OR CUSTOMIZATION where OSHW devices are developed and sold, with income primarily from support and customization. Both the Arduino founders and SparkFun Electronics [26] are wildly different examples of how this can work, with SparkFun Electronics primarily selling individual devices while the primary income for the Arduino founders coming from clients integrating their devices or hiring them as consultants [30]. Makerbot, however, is one example of where this business model (likely) did not work. A developer of a 3D-printer, their devices were initially OSHW, however after clones started appearing leading to (alleged) force from their venture capital financiers, they have now adopted a closed-source model with a large outcry from the community that previously had supported them[16].

HARDWARE VERSIONING where simple devices are freely released while more complex, functional, or advanced devices are sold under a closed-source license. This is based on the hope of upgrades among the users of the devices, who begin with the relatively simple device, and as requirements increase switch to the more capable, closed-source, version.

INTEGRATED OSHW MODULES where modules that are difficult to manufacture are sold and released with full documentation. This is primarily applicable where the majority of those who use the device do not have the required hardware to manufacture the device, while still giving the users the full ability to study and understand the functionality. To some extent SparkFun Electronics matches this category as well, as many of their customers do not have the knowledge needed or find it worth their time to manufacture the devices rather than purchasing them directly from SparkFun Electronics.

There are of course nearly infinite variants and combinations of business models that may be applicable for OSHW, making the end decision of whether or not to implement OSHW development something that must be done on a case-by-case basis.

Generally, the findings shown in 5.3.2.1 and 5.3.2.2 must considered and weighed against the business models currently used by companies considering adopting OSHW development. Methods of performing this evaluation are far beyond the scope of this report, yet remain an interesting research question for further studies.

16 See http://blog.makezine.com/2012/09/19/is-one-of-our-open-source-heroes-going-closed-source/.

6 CONCLUSIONS

The core research question of this report;

> *Can embedded electronics companies thrive through the use and/or development of* OSHW*?*

has been primarily studied through the use of a survey (administered to an audience with a background and/or interest in embedded electronics) and interviews with two different companies developing embedded electronics modules (SparkFun Electronics and Pololu, which have and have not respectively adopted an OSHW work-flow).

The results from the original research, placed in a business-context, highlight the differences between using OSHW and developing OSHW. Generally, use of OSHW was found to be less risky and easier to implement than developing OSHW in a running company.

Use of OSHW shows great use as a tool in a development setting, where direct use or use with minor in-house changes would fill a gap or be significantly less expensive than traditional commercial products. Additionally, OSHW as a (part of a) product was identified to be useful; where the device can be modified to create a new product (not necessarily an OSHW product) or integrated into existing product development, reducing development time and costs.

On the other hand, sometimes using OSHW devices does not really make sense, such as; poorly documented or designed devices; situations where traditional devices are available at low cost, that require no modification for use and where lock-in effects are minor; and situations where the knowledge needed to evaluate if a design is good or not does not exist within the company and misjudging design quality would be very costly.

The utility of developing OSHW in a company is a more complex question, and not fully answered by the research in this report. Though some companies have achieved stability through development of OSHW devices, others have failed, the reasons of which are beyond the scope of this report. Generally, users of OSHW reported the ability to study the inner workings of a device a very important, and rated OSHW devices very highly because of (among others) this reason. However, the ability for others to clone or duplicate parts of a design, due to the lack of IP protection inherent in OSHW licenses, is viewed as a large risk, potentially making it difficult to maintain profitability. Though the ability for a company to succeed through the development of OSHW has been shown, both through simply observing the existence

of several companies doing this[1], as well as the benefits highlighted in the original research, the reasons for one company succeeding and another failing remains to be determined.

In the end, this report has established some much-needed empirical data into the values held by users and developers of OSHW, and answered the primary research question;

> *Can embedded electronics companies thrive through the use and/or development of* OSHW*?*

with;

> *There exist several companies that have succeeded with both development and/or use of* OSHW*.*
>
> *However, several others have not.*
>
> *Use of* OSHW *was found to be generally less risky than development of* OSHW *and offers a great ability to reduce development costs, integrate a large amount of "free" development work into the company's own development process, and reduce the dependence of external suppliers. However, the level of documentation and support of* OSHW *devices must be verified prior to adoption.*
>
> *Development of* OSHW *is a bolder endeavor, and though users value* OSHW *devices highly, in particular if accurately documented, the risk of others cloning or duplicating sections of the device remains large due to the poor* IP *protection capabilities available.*

Further research into OSHW may well find the results from the survey useful, in particular for research studying the applicability of OSHW in businesses. As this report has not considered (among others) the political and psychological effects that OSHW has on businesses, further studies into these topics would be very applicable for shedding more light onto the ability for OSHW to be used in companies and the reasons that some succeed while others fail.

1 Such as SparkFun Electronics [26], Adafruit Industries [4], and the Arduino group [5].

BIBLIOGRAPHY

[1] Open Source Hardware Statement of Principles and Definition V1.0. http://freedomdefined.org/Definition. (Cited on pages 4 and 13.)

[2] Open Source Initiative. http://opensource.org/osd. (Cited on pages 8 and 71.)

[3] TAPR Open Hardware License. http://www.tapr.org/ohl.html. (Cited on pages 15 and 76.)

[4] Adafruit Industries. http://www.adafruit.com. (Cited on pages 16, 20, and 80.)

[5] Arduino. http://www.arduino.cc/. (Cited on pages 20, 76, and 80.)

[6] Why Microsoft is wary of open source. http://news.cnet.com/2100-1001-268520.html. (Cited on page 7.)

[7] The BSD 3-Clause License. http://opensource.org/licenses/BSD-3-Clause. (Cited on page 10.)

[8] Evil Mad Science LLC. http://www.evilmadscientist.com/. (Cited on pages 16 and 20.)

[9] GNU General Public License, version 3. http://opensource.org/licenses/GPL-3.0. (Cited on pages 10 and 76.)

[10] Hackaday Online Magazine. http://www.hackaday.com. (Cited on pages 19 and 27.)

[11] 15 Great Quotes from Torvalds and Stallman about Free and Open Source Software. http://www.junauza.com/2008/09/15-great-quotes-from-torvalds-and.html. (Cited on page 7.)

[12] OSHW Community Survey 2012. http://www.oshwa.org/oshw-community-survey-2012/. (Cited on page 19.)

[13] Tektronix 545A Oscilloscope. http://www.thevalvepage.com/testeq/tek/545a/545a.htm. (Cited on pages xii, 66, and 67.)

[14] Roberto Acosta. Open source hardware. Master's thesis, Massachusetts Institute of Technology, 2009. (Cited on pages 5, 14, 16, 56, 76, and 77.)

[15] Sara Boettiger and Dan Burk. Open source patenting. *Journal of International Biotechnology Law*, 1:221–231, 2004. (Cited on page 15.)

[16] Andrea Bonaccorsi and Cristina Rossi. Why Open Source Software can succeed. *Research Policy*, 32(7):1243–1258, 2003. (Cited on pages 10, 11, and 14.)

[17] Joseph Feller, Brian Fitzgerald, et al. *Understanding open source software development*. Addison-Wesley London, 2002. (Cited on page 10.)

[18] R. Ghosh and V.V. Prakash. The Orbiten Free Software Survey. *First Monday*, 5(7), 2000. (Cited on pages 11 and 14.)

[19] Robert L. Glass. Loyal Opposition - Of Open Source, Linux...and Hype. *IEEE Software*, 16(1):128, 1999. (Cited on page 10.)

[20] Nick Heath. 'We thought we'd sell 1,000': The inside story of the Raspberry Pi. http://www.zdnet.com/we-thought-wed-sell-1000-the-inside-story-of-the-raspberry-pi-7000009718/. (Cited on page 16.)

[21] Frank Hecker. Setting Up Shop: The Business of Open-Source Software. *IEEE Software*, 16(1):45–51, 1999. (Cited on page 12.)

[22] Julian P Höppner. The GPL prevails: An analysis of the first-ever Court decision on the validity and effectivity of the GPL. *SCRIPT-ed*, 1(4):628–635, 2004. (Cited on page 15.)

[23] Sandeep Krishnamurthy. An analysis of open source business models. 2005. (Cited on pages 12, 13, and 56.)

[24] Josh Lerner and Jean Tirole. Some Simple Economics of Open Source. *Journal of Industrial Economics*, 50:197–234, 2002. URL http://dx.doi.org/10.1111/1467-6451.00174. (Cited on page 11.)

[25] Jan Malášek. Pololu Corporation. http://www.pololu.com, March 2013. (Cited on pages vii, x, and 92.)

[26] Chelsea Moll. SparkFun Electronics. http://www.sparkfun.com, March 2013. (Cited on pages vii, x, 16, 56, 74, 77, 80, 87, 89, and 91.)

[27] Bruce Perens et al. The open source definition. *Open sources: voices from the open source revolution*, pages 171–85, 1999. (Cited on page 3.)

[28] Lennart Råde and Bertil Westergren. *Mathematics Handbook for Science and Engineering*. Studentlitteratur, 5 edition, 2004. (Cited on page 28.)

[29] Lawrence Rosen. *Open Source Licensing. Software Freedom and Intellectual Property Law*. Prentice Hall, Upper Saddle River, NJ, 2005. (Cited on page 10.)

[30] Clive Thompson. Build it. share it. profit. Can open source hardware work. *Wired Magazine*, 16(11):16–11, 2008. (Cited on pages 14, 76, and 77.)

[31] Phillip Torrone. Why Google Choosing Arduino Matters and Is This the End of "Made for iPod" (TM)? http://blog.makezine.com/2011/05/12/why-google-choosing-arduino-matters-and-the-end-of-made-for-ipod-tm/. (Cited on page 16.)

[32] Phillip Torrone. MAKE's Exclusive Interview with Jan Malasek from Pololu. http://blog.makezine.com/2012/04/25/makes-interview-with-jan-malasek-from-pololu/, April 2012. (Cited on pages vii, x, 14, 75, 93, and 95.)

[33] Eric Von Hippel. Learning from open-source software. *MIT Sloan management review*, 42(4):82–86, 2001. (Cited on page 10.)

[34] Steve Weber. *The success of open source*, volume 368. Cambridge Univ Press, 2004. (Cited on pages 3, 7, 8, 10, 11, 13, and 56.)

Part IV

APPENDIX

A

INTERVIEW TRANSCRIPTS

A.1 INTERVIEW WITH CHELSEA MOLL FROM SPARKFUN ELECTRONICS [26]

On your about page you describe what Sparkfun is today, do you have a brief description of how and for what reasons Sparkfun was founded?

SparkFun was founded in 2003 when our CEO Nathan Seidle was an electrical engineering student at the University of Colorado. He was working on a project and fried an expensive board. While looking for an affordable replacement, he realized there was a frustrating lack of websites offering small, affordable quantities of individual components with clear images of their inventory, and decided it was a gap in the electronics market that needed filling.

According to Wikipedia, Sparkfun was one of the contributors in drafting the first Open Source Hardware (OSHW) definition. How was it that open source hardware became such a key part of who you are?

The concept of an open exchange of ideas for the good of the community makes very clear sense to us. In fact, our company motto, which sums up a lot of our motivation and convictions - is "sharing ingenuity." In the beginning, before the open source movement was organized enough to be called a movement, we published our source code, schematics and eagle and gerber files for our early products. We did this because we weren't experts yet, and we figured if someone had an idea on how to improve our designs, then it would be in the best interest of everyone if we could work together. We still don't think we're experts, and we still believe that working together yields better results, and now we're happy to be part of a recognized, burgeoning open source community.

Open source makes sense to us from both a practical and ethical standpoint; we want to be more useful to the customer than not, and being forthcoming with all of our materials seems a simple and obvious thing. And, as it turns out, it makes amazing business sense, too. We've never had to worry about IP or patents, or, for the most part, lawyers. We just want to focus on what we love to do: build stuff with our toys and sharing the love of electronics with others. Being open source allowed us to be closer to that ideal, and less what might be considered "business norm." And it turns out that there are a lot of people out there that feel as we do.

One of the commonly cited "issues" with OSHW is the ability for someone else to clone an idea and undercut the development costs (in particular for complex devices). What experience have you had with this behavior? Do people clone your devices, and if so, do sales drop when the clones become available?

Honestly, if someone's clever enough to take our designs and make their own boards, they're likely clever enough to improve on our work. We can't really fault them for that, and we'll emulate them as they emulated us, rolling their improvements into our own design and trying to improve further. We share attribution, everybody's got a better product and the customer is happier. Therein lies the essence of open source, and the driving force of SparkFun.

The threat of someone being able to do our job better keeps us on our toes and forces us to work harder to stay current, flexible, and on the cutting edge of the tech we sell. This is a great benefit of the open source business model – the market gets to create and evolve tools and products along with the business that produces them. We have to be willing to kill ideas that don't work, take a lot of tough criticism, and be agile. Competition is a huge motivator.

There's a multitude of new widgets that you are developing and releasing all the time, how do you develop them? Is it mostly through engineers employed at sparkfun, external people who believe in what you are doing and submitting their designs freely, or something else completely?

It's both, really. For our engineering department, developing new products and finding new ways to improve on existing designs is the job. They come up with ideas for new products (or revisions for existing ones) along with customer feedback, then they design the board, review products as a group, and then begin to prototype and write example code where necessary. Besides that, they are encouraged to write tutorials to help users with new products. We also encourage our engineers to use their products in side projects when they have a chance.

And everyone who works at SparkFun is encouraged to get involved in concepts for new products or upgrades; more brains = more ideas. We've had members of Production (manufacturing), IT and Tech Support departments contribute to product design and development. If people in the building see a problem or have an idea, we encourage them to address it.

We also pay close attention to comments and suggestions from our customers and community, and we take them to heart. Every product page on our site has room for comments on the bottom of the page. As customers leave comments we are able to respond, but we also use that information to determine which products need a revision or an accompanying product that we have yet to develop.

By staying involved in and contributing to the DIY community ourselves - either by building our own projects, communicating with our customers, or visiting events and hackerspaces - we're able to keep a finger on the pulse of the electronics market and figure out what our community needs.

How would you describe the business model that Sparkfun is employing? A materialistic way of viewing it would be that you exchange physical goods for economic payment, but is there something more gained by releasing your design files under an OSHW license?

Well, I'm not sure if materialistic is the right word, but yes, we are a business. In the end, we want to make enough money to keep running and, if we are lucky, keep growing. That being said, you are right that the open source model is intrinsically better; information is power. If we give all the information about our products to our customers, they have a more powerful tool for their project, prototype, etc.

Plus, if we are willing to be transparent with and supportive of a community that's going to experiment with our products (and possibly improve them) anyway, that community is going to be more loyal to us for it, as opposed to a company that guards its intellectual property and treats people who modify their designs as criminals. Our customers learn and cultivate their interest by getting their hands dirty and trying new things, and we're more than happy to help them get excited about electronics any way we can - that's how SparkFun started, after all.

Do you have some figures as to the number of devices you sell (as of now) on a yearly basis? Maybe even a (rough) figure of how much sales income you have?

In 2012, we sold somewhere in the neighborhood of 727,650 items on 192,000 orders. From 2011 to 2012 we grew about 9.5%, and we did around $27.5 million in revenue in 2011.

Many of your widgets and devices are relatively simple break-out boards for IC's with some degree of supporting components; there are far fewer complex devices available. (For example, the logic level converter (https://www.sparkfun.com/products/8745) and Xbee explorer USB (https://www.sparkfun.com/products/8687) are popular simple devices, and the Digital Oscilloscope DIY Kit (https://www.sparkfun.com/products/9484) is far more complex.) Have you felt that your customers are more interested in simple modular units? Does the development, testing, and evaluation time required for the more complex widgets make them a less interesting investment? All in all, what is the reasoning behind this distribution of device complexity?

Technical Researcher Pearce Melcher: The simple modular units appeal to a certain customer base, while the more complex kits and development boards appeal to another. The simple modules fill a void for people looking to prototype, or use a new sensor or IC, who don't want to buy a $200 full-featured development board. Instead, for a fraction of the cost, they can purchase the sensor they want to use on an easily solder-able board that breaks out all or the majority of the functional pins of the device. We've found these to be popular with students looking for a easy way to include devices in their projects, and also with engineers and researchers prototyping in the commercial world. We've been told by some of our suppliers that they refer users who are looking for this simple solution to their products to SparkFun.

The larger, more complex products are usually geared towards hobbyists and those looking to learn about electronics. The DIY Oscilloscope kit is great for someone looking to learn to solder, and provides a useful tool when finished. Other, more complex boards or kits may teach a development tool like Arduino, or may achieve a solution that your typical novice user could not do on their own. For example, the MP3 Trigger allows users to easily add sound effects to their project. This cuts down greatly on development time and frustration someone would experience trying to assemble a similar platform on their own.

Of course there is plenty of overlap between these markets. Chances are, if you work in the electronics industry as a designer or engineer, you most likely enjoy electronics as a hobby in your free time. On the other end, those who tinker in their free time might consider electronics or a similar field as a career. So the benefit of carrying both basic modules and more complex boards and development kits is to reach a broader audience - covering different markets in the electronics world, each equally valuable.

Engineer Mike Hord: It behooves us not to try and guess what our customers are going to do. The more complex a design is, the fewer uses it has.

Engineer Jordan McConnell: Modularity allows customers to pick and choose the necessary components for their projects. We also generally have example code for each modular piece so that allows for easy testing and easier integration into a larger project.

Testing to make sure new electronic devices work as function and pass legal requirements (EMI, safety, and so on) is typically regarded as expensive and time consuming. One of the fundamentals of OSHW is right to change or modify a circuit to become whatever you want it to be. As some testing must be done on every change to hardware, this places some limitations on the ability to (legally) change things freely. How do you at Sparkfun test and evaluate your devices before releasing them? How can a hacker use the right to make changes to a widget without paying the very large sums for testing that is traditionally viewed as a requirement?

Engineer Mike Hord: The simple answer is, we avoid creating products that require a great deal of testing as much as possible. Most of what we make can rightly be considered a subassembly or a kit, and requires little or no testing.

In most cases, a hacker making changes on their own, for their own use, doesn't have a great deal of regulatory onus on them. If they're making a derivative product to resell, then I would expect them to be aware of the laws regarding safety and EMC the same as they should be aware of, for instance, tax obligations. It's part of the cost of doing business.

Engineer Jordan McConnell: We do a lot of testing for the functionality and quality of our boards, but not a lot of testing for 'legality' or safety. It's rare that one of our products needs FCC or other legal testing, and we often avoid creating these for that reason. Also, due to using low DC voltages, our boards are very safe compared to working with direct AC from a wall outlet. For hackers worried about whether or not they need FCC testing, Mike Hord wrote a FCC tutorial.

Sparkfun has a relatively new and very exciting educational department, could you tell me a little bit about the causes for starting this department and how things have gone with it?

We started the SparkFun Department of Education in 2011 with the hope that it would serve as a hub to spread our passion for electronics to educators and students worldwide. The department offers free, open-source curriculum for users to customize; products; tutorials; and classes designed to encourage people of all ages and skill levels to play, create and invent.

So far the department has been a huge success - we've traveled all over the country teaching, and we've hosted a lot of classes at our Colorado headquarters. In fact, this June we're embarking on a

National Education Tour for the remainder of the year. We recently bought an RV and we're going to hit the road, teaching electronics basics at makerspaces, schools and anywhere people want to sponsor a stop — and we've had dozens sign up already!

A.2 INTERVIEW WITH JAN MALÉŠEK, COMPANY PRESIDENT OF POLOLU [25]

Though your products are not released under an open source hardware (OSHW) license, many of your devices are released with a schematic (which is not the norm for proprietary products). Why have you chosen to do this? Why do you believe that this is a good level of openness?

Releasing schematics and otherwise explaining how our products work are good for documentation and further our goal of helping those wanting to learn about electronics. That therefore helps the end customer in a direct way. I suspect that full access to source files, on the other hand, are unlikely to be helpful to most customers and disproportionally help just those who would make knock-offs without contributing anything.

You seem to be taking a lot of flak in the comment section of your interview with MAKE for not releasing your devices under an OSHW license and encrypting the firmware for microprocessor-based devices. Have you found that the users of your devices are as interested as the people in the comment section that your devices be released under an OSHW license? Why/why not do you think this is the case?

I don't think the comments in that Make interview amount to "a lot of flak", and it's a very small sample size. You can see in the few comments (again, small sample size) in my blog post that our position does not seem to be particularly controversial. I think it's not much of an issue for most of our customers.

What do you see happening to the OSHW movement in the future?

I am skeptical about the OSHW movement ever becoming very relevant to the world, for the reasons I mentioned in the blog post: "I see two main reasons OSHW does not live up to its claimed potential: the source documents tend to have little value as a starting point for anything other than very minor changes, and there are other large barriers to meaningful participation."

A.3 MAKER MEDIA'S INTERVIEW WITH JAN MALÉŠEK [32]

Reproduced verbatim with permission.

Hi Jan! Thanks for answering these questions! Where are you right now?

I'm in my office in Las Vegas.

How did Pololu get started?

We got started in 2000 as college students making an IR beacon system for MIT's 6.270 autonomous robot contest. (There's more info about it here) I was involved with organizing the contest for a few years, and the beacons they were using the first year I was there were very limited and cost a lot, so I convinced the other organizers to get the beacons from me if I could come up with something better.

What are the products you're most proud of?

At this point, I'm still shooting for "not embarrassing". Some of the products that fit that designation are our 3pi robot, Maestro servo controllers, and Wixel wireless modules. Those products have some decent engineering to them, we've sold thousands of them, and customers seem to like them.

What are your more popular products?

Besides the ones I just mentioned, some of the simple sensor and motor driver carriers are quite popular. It's kind of frustrating when a really simple carrier sells more than a design we spend months on.

How many people do you have?

Around 45.

Why did you move to Nevada?

I think a lot of Nevada is quite different from Las Vegas, though I've only driven through Reno once. Las Vegas was appealing because it doesn't get very cold, natural disasters are not very likely, and the business climate seemed relatively good (coming from Massachusetts). I like the idea of prostitution and gambling being legal in Nevada, even if I don't engage in it (my gamble was coming to Las Vegas in the first place).

Is it true the taxes are less? Was that a reason?

I think so, and it definitely contributed to the decision. Unfortunately, having no income tax didn't help that much when we weren't making

money, and the sales tax is pretty high (and keeps rising), which sucks when we're buying equipment.

Is there a lot of space for a maker company to have machines like manufacturing equipment, etc.?

If you're asking about commercial real estate space, there's quite a bit available, though it can be difficult to find a space with the right mix of warehouse and office space for a company like ours. I am quite happy with the space we just moved to, where we have almost 40,000 square feet, of which about 12,000 is warehouse.

How is the maker scene out there? Are there hackerspaces and events in Nevada?

We helped start the Las Vegas robot club, LVBots, around 2004. Participation in that was decent and growing until around 2008; it dropped off quite a bit as Las Vegas got hit by the recession. Some of the LVBots members are involved in a Las Vegas hacker space (SYN Shop) that's been slowly getting off the ground, but I have not been involved with it, so I do not really know what they are up to.

What products of yours are open-source hardware?

None.

You're in the maker world, but you do not release any of your products as open-source hardware, why?

There are many reasons, but they mostly boil down to not thinking it's worth it from a business perspective. I am somewhat skeptical of companies pushing OSHW, so I think it would be disingenuous to use what I think is a gimmick to market a product. For instance, the 3pi robot I mentioned earlier has several custom mechanical parts, a patented circuit, and is done in Altium Designer, which costs over $5k per seat. I realize that some schools might have site licenses for Altium, but for the typical user, releasing the source documents would not mean much. We already provide schematics and explanations of how the thing works, the design is fairly well validated, and it's not a particularly useful starting point for an improved design (e.g. one with a better processor), so the main entities actually getting something out of our releasing the source would be those trying to make knock-offs, and I'm fine with not giving them a shortcut.

In my previous article about Arduino being counterfeited you said "I do not think intellectual property is a morally valid concept, and I am all about freedom" – that comment surprised me and was the reason I wanted to do this interview, what did you mean?

My father escaped from communist Czechoslovakia when he was eighteen; he instilled in me a love of freedom. I think a lot of the problems in the world come from people thinking it's okay to force others to do things, especially when that force is coming from a democratic society. Physical property must be protected since exclusivity is inherent in physical things (I no longer have something if you take it from me) and the rights to the result of your own work are fundamental to basic freedom. That is not the case with intellectual property: the primary purpose of that construct is to forcibly prevent people from doing things. If I invent a wheel, of course I would rather you give me something in exchange for me making you one, but it is not right for me to go beat you up or destroy your wheel if you make one yourself. Besides being immoral, intellectual property is also empirically bad for society; however, that should not really be relevant to the discussion, just as consideration of economic costs should be irrelevant when considering the morality of slavery.

Are there any products you plan to release as open-source hardware, if so which ones?

There's nothing specific in the works. The most likely candidates are supporting boards for something like a mechanical chassis. But since the hardware would support a proprietary product (the chassis) and be done in Altium, I don't think we would make a big deal of it.

You had mentioned you filed a patent or attempted to, what was it for?

US patent 7781920 is for the pushbutton power switch we use in several of our products. To those that might think it's hypocritical to think patents are bad and then to get one, I think it's acceptable to play by the rules while saying the rules are bad. I doubt that getting a lot of patents will be part of our long-term strategy, and participating in the system might give more credibility to my criticisms of it.

You also "DRM" your bootloaders, why? Has it helped protect your intellectual property? Did it work, has anyone cracked it?

I think it's not quite right to call it DRM, in the sense that we do not talk about selling that secured firmware. We sell, for instance, a servo controller, and once you buy it, you have it. The servo controller has the capability of being updated, but we do not sell the updates or try to limit you to using a particular update on a particular servo controller. Anyway, we secure the bootloaders so that people cannot

copy our designs too easily. We shoot for a level of security such that it would be easier to just rewrite the firmware from scratch than to try to crack the security. I don't know if anyone has cracked it. You seemed to think there was no room for secrets in my freedom principle. Secrets are fine in that you do not owe it to anyone to tell them things you do not want to, and you should not be forced to tell others something.

What new products are working on?

We just released the mechanical part of our new Zumo robot, which in some sense has been in development for almost 5 years. We will be working on electronics for that such as an Arduino shield and stand-alone controllers.

Where can people see some Pololu hardware in person?

Counting our electronics boards and mechanical parts like wheels and brackets, we've sold hundreds of thousands of units, so chances are that something we made is in a maker project near you. We're also happy to give tours and show people our products in person, so if you're interested and in Vegas, let us know.

B

COLLECTED SURVEY DATA

Table 4: Numerical survey results as introduced in Chapter 4, divided into the utilization groups described.

Note; NC User refers to the non-commercial user group, NC Dev. refers to the non-commercial developer group, C User refers to the commerical user group, C Dev. refers to the commercial developer group, and No Exp. refers to the no prior experience group. See 4.1 on page 29 for an explanation of the meaning of the questions.

Question	NC User	NC Dev.	C User	C Dev.	No Exp.
Number of responses in group;	*271*	*103*	*17*	*12*	*51*
Useage of OSHW Devices; Direct use	72.3%	65.0%	47.1%	66.7%	-
Useage of OSHW Devices; Minor changes	64.2%	66.0%	41.2%	58.3%	-
Useage of OSHW Devices; Large changes	21.4%	38.8%	23.5%	33.3%	-
Useage of OSHW Devices; As a base	35.1%	46.6%	41.2%	75.0%	-
Ability to use proprietary equivalent; Always	7.0%	10.7%	17.6%	16.7%	-
Ability to use proprietary equivalent; Mostly	24.0%	15.5%	29.4%	33.3%	-
Ability to use proprietary equivalent; Sometimes	35.4%	29.1%	35.3%	16.7%	-
Ability to use proprietary equivalent; Rarely	18.1%	24.3%	5.9%	25.0%	-
Ability to use proprietary equivalent; Never	6.6%	6.8%	5.9%	0.0%	-
Ability to use proprietary equivalent; Unsure	7.0%	6.8%	0.0%	8.3%	-
Developer involvement; Founded project	-	59.2%	-	75.0%	-
Developer involvement; Major contribution	-	12.6%	-	16.7%	-
Developer involvement; Minor contribution	-	55.3%	-	41.7%	-
Use of project by others; Used by others	-	44.7%	-	41.7%	-
Use of project by others; Discussed with others	-	20.4%	-	8.3%	-
Use of project by others; Probably	-	18.4%	-	25.0%	-
Use of project by others; Unsure	-	13.6%	-	8.3%	-
Use of project by others; No	-	11.7%	-	16.7%	-
Reason for contribution; To solve a problem	-	51.5%	-	83.3%	-
Reason for contribution; Educational reasons	-	55.3%	-	33.3%	-
Reason for contribution; As a hobby	-	82.5%	-	0.0%	-
Simple device; functionality; OSHW ++	31.7%	29.1%	23.5%	16.7%	27.5%
Simple device; functionality; OSHW +	18.5%	21.4%	35.3%	25.0%	15.7%

Continued on next page

Question	NC User	NC Dev.	C User	C Dev.	No Exp.
Simple device; functionality; Equal	39.1%	40.8%	29.4%	50.0%	37.3%
Simple device; functionality; Prop. +	5.2%	5.8%	11.8%	0.0%	9.8%
Simple device; functionality; Prop ++	1.1%	0.0%	0.0%	0.0%	3.9%
Simple device; functionality; Unsure	3.7%	2.9%	0.0%	8.3%	5.9%
Simple device; stability; OSHW ++	13.7%	17.5%	17.6%	8.3%	21.6%
Simple device; stability; OSHW +	19.6%	20.4%	35.3%	16.7%	17.6%
Simple device; stability; Equal	39.9%	34.0%	17.6%	50.0%	35.3%
Simple device; stability; Prop. +	18.8%	13.6%	17.6%	16.7%	13.7%
Simple device; stability; Prop ++	4.1%	6.8%	5.9%	0.0%	3.9%
Simple device; stability; Unsure	3.0%	5.8%	0.0%	8.3%	7.8%
Simple device; cost; OSHW ++	53.9%	43.7%	70.6%	58.3%	45.1%
Simple device; cost; OSHW +	24.4%	24.3%	29.4%	33.3%	35.3%
Simple device; cost; Equal	9.6%	14.6%	0.0%	0.0%	11.8%
Simple device; cost; Prop. +	5.5%	7.8%	0.0%	8.3%	0.0%
Simple device; cost; Prop ++	3.7%	3.9%	0.0%	0.0%	3.9%
Simple device; cost; Unsure	2.2%	5.8%	0.0%	0.0%	3.9%
Simple device; documentation; OSHW ++	42.4%	50.5%	29.4%	41.7%	47.1%
Simple device; documentation; OSHW +	25.5%	23.3%	17.6%	25.0%	23.5%
Simple device; documentation; Equal	18.5%	9.7%	17.6%	16.7%	13.7%
Simple device; documentation; Prop. +	4.8%	7.8%	23.5%	8.3%	5.9%
Simple device; documentation; Prop ++	4.4%	2.9%	5.9%	8.3%	7.8%
Simple device; documentation; Unsure	3.7%	4.9%	5.9%	0.0%	2.0%
Simple device; support; OSHW ++	37.3%	35.9%	17.6%	41.7%	33.3%
Simple device; support; OSHW +	31.7%	26.2%	35.3%	16.7%	23.5%
Simple device; support; Equal	14.8%	13.6%	17.6%	16.7%	19.6%
Simple device; support; Prop. +	5.5%	11.7%	5.9%	8.3%	11.8%
Simple device; support; Prop ++	4.4%	2.9%	17.6%	8.3%	5.9%
Simple device; support; Unsure	5.5%	8.7%	5.9%	8.3%	5.9%
Simple device; gut feeling; OSHW ++	52.0%	47.6%	23.5%	50.0%	47.1%
Simple device; gut feeling; OSHW +	32.5%	40.8%	47.1%	41.7%	35.3%
Simple device; gut feeling; Equal	10.0%	6.8%	23.5%	8.3%	5.9%
Simple device; gut feeling; Prop. +	3.0%	1.0%	5.9%	0.0%	5.9%
Simple device; gut feeling; Prop ++	0.0%	0.0%	0.0%	0.0%	3.9%
Simple device; gut feeling; Unsure	1.8%	2.9%	0.0%	0.0%	2.0%
Complex device; functionality; OSHW ++	26.6%	35.0%	35.3%	25.0%	25.5%

Continued on next page

Question	NC User	NC Dev.	C User	C Dev.	No Exp.
Complex device; functionality; OSHW +	23.2%	26.2%	11.8%	0.0%	19.6%
Complex device; functionality; Equal	21.4%	22.3%	23.5%	33.3%	23.5%
Complex device; functionality; Prop. +	17.7%	5.8%	5.9%	25.0%	9.8%
Complex device; functionality; Prop ++	4.8%	7.8%	17.6%	0.0%	13.7%
Complex device; functionality; Unsure	5.5%	2.9%	5.9%	16.7%	7.8%
Complex device; stability; OSHW ++	10.0%	17.5%	11.8%	8.3%	13.7%
Complex device; stability; OSHW +	18.8%	19.4%	17.6%	16.7%	13.7%
Complex device; stability; Equal	29.9%	30.1%	29.4%	33.3%	29.4%
Complex device; stability; Prop. +	26.6%	21.4%	23.5%	25.0%	27.5%
Complex device; stability; Prop ++	5.9%	8.7%	11.8%	0.0%	5.9%
Complex device; stability; Unsure	6.6%	2.9%	5.9%	16.7%	7.8%
Complex device; cost; OSHW ++	38.7%	36.9%	52.9%	33.3%	31.4%
Complex device; cost; OSHW +	27.3%	30.1%	17.6%	33.3%	31.4%
Complex device; cost; Equal	14.0%	13.6%	17.6%	8.3%	15.7%
Complex device; cost; Prop. +	8.9%	9.7%	5.9%	0.0%	11.8%
Complex device; cost; Prop ++	4.8%	8.7%	0.0%	0.0%	3.9%
Complex device; cost; Unsure	5.5%	1.0%	5.9%	25.0%	5.9%
Complex device; documentation; OSHW ++	35.8%	38.8%	35.3%	25.0%	37.3%
Complex device; documentation; OSHW +	26.9%	29.1%	23.5%	16.7%	23.5%
Complex device; documentation; Equal	16.6%	14.6%	5.9%	41.7%	23.5%
Complex device; documentation; Prop. +	10.0%	9.7%	23.5%	0.0%	9.8%
Complex device; documentation; Prop ++	2.2%	2.9%	5.9%	0.0%	3.9%
Complex device; documentation; Unsure	6.6%	4.9%	5.9%	16.7%	2.0%
Complex device; support; OSHW ++	30.6%	36.9%	29.4%	16.7%	27.5%
Complex device; support; OSHW +	32.1%	35.0%	23.5%	33.3%	21.6%
Complex device; support; Equal	17.3%	11.7%	17.6%	16.7%	19.6%
Complex device; support; Prop. +	9.2%	4.9%	11.8%	16.7%	21.6%
Complex device; support; Prop ++	4.1%	5.8%	11.8%	0.0%	5.9%
Complex device; support; Unsure	5.9%	5.8%	5.9%	16.7%	3.9%
Complex device; gut feeling; OSHW ++	38.0%	43.7%	23.5%	33.3%	43.1%
Complex device; gut feeling; OSHW +	32.5%	38.8%	41.2%	41.7%	29.4%
Complex device; gut feeling; Equal	14.0%	6.8%	11.8%	16.7%	13.7%
Complex device; gut feeling; Prop. +	6.3%	4.9%	17.6%	0.0%	3.9%
Complex device; gut feeling; Prop ++	2.2%	2.9%	0.0%	0.0%	7.8%
Complex device; gut feeling; Unsure	5.5%	2.9%	5.9%	8.3%	0.0%

Continued on next page

Question	NC User	NC Dev.	C User	C Dev.	No Exp.
Transparency; Very important	67.9%	75.7%	64.7%	83.3%	64.7%
Transparency; Slightly important	26.6%	19.4%	29.4%	16.7%	25.5%
Ttransparency; Neither nor	4.1%	4.9%	5.9%	0.0%	5.9%
Transparency; Slightly unimportant	0.7%	0.0%	0.0%	0.0%	2.0%
Transparency; Completely unimportant	0.0%	0.0%	0.0%	0.0%	2.0%
Modification; Very important	70.8%	70.9%	76.5%	83.3%	56.9%
Modification; Slightly important	26.2%	23.3%	17.6%	16.7%	31.4%
Modification; Neither nor	2.2%	4.9%	0.0%	0.0%	5.9%
Modification; Slightly unimportant	0.0%	1.0%	0.0%	0.0%	2.0%
Modification; Completely unimportant	0.0%	0.0%	0.0%	0.0%	2.0%
Re-use; Very important	60.5%	59.2%	58.8%	41.7%	68.6%
Re-use; Slightly important	28.0%	32.0%	41.2%	50.0%	21.6%
Re-use; Neither nor	8.1%	4.9%	0.0%	0.0%	2.0%
Re-use; Slightly unimportant	1.5%	3.9%	0.0%	0.0%	5.9%
Re-use; Completely unimportant	0.7%	0.0%	0.0%	0.0%	2.0%
Philosophy; Very important	63.5%	68.0%	41.2%	66.7%	68.6%
Philosophy; Slightly important	22.5%	22.3%	17.6%	16.7%	19.6%
Philosophy; Neither nor	8.1%	3.9%	17.6%	16.7%	5.9%
Philosophy; Slightly unimportant	3.7%	3.9%	5.9%	0.0%	2.0%
Philosophy; Completely unimportant	1.5%	0.0%	11.8%	0.0%	3.9%
Education; Some high school	8.5%	8.7%	0.0%	0.0%	11.8%
Education; Finshed high school	7.0%	8.7%	0.0%	0.0%	9.8%
Education; Some university/college	34.3%	27.2%	17.6%	33.3%	41.2%
Education; Bachelor's degree	34.3%	36.9%	47.1%	50.0%	25.5%
Education; Master's degree	13.7%	17.5%	35.3%	16.7%	9.8%
Education, PhD or higher	1.8%	1.0%	0.0%	0.0%	2.0%
Age; Under 15	0.0%	0.0%	0.0%	0.0%	0.0%
Age; 15-18	11.1%	9.7%	0.0%	0.0%	9.8%
Age; 19-22	22.1%	11.7%	0.0%	16.7%	19.6%
Age; 23-27	20.7%	23.3%	23.5%	0.0%	31.4%
Age; 28-33	17.3%	24.3%	41.2%	25.0%	17.6%
Age; 34-39	11.8%	18.4%	0.0%	25.0%	3.9%
Age; 40-47	9.6%	6.8%	17.6%	8.3%	3.9%
Age; 48-57	6.3%	3.9%	5.9%	8.3%	11.8%
Age; 58-68	1.1%	1.0%	11.8%	16.7%	2.0%

Continued on next page

Question	NC User	NC Dev.	C User	C Dev.	No Exp.
Age; Over 68	0.0%	0.0%	0.0%	0.0%	0.0%
Residence; Africa	1.1%	0.0%	0.0%	0.0%	0.0%
Residence; Asia	1.1%	1.9%	0.0%	0.0%	5.9%
Residence; Australia	5.9%	6.8%	5.9%	8.3%	5.9%
Residence; Europe	42.4%	40.8%	64.7%	25.0%	29.4%
Residence; North America	46.9%	49.5%	29.4%	58.3%	49.0%
Residence; South America	2.2%	0.0%	0.0%	0.0%	7.8%

End of table

B.1 PARTICIPANT COMMENTS

Many participants of the survey chose to leave an optional comment; most of these are listed below, grouped into the same five categories as used in Chapter 4.

Due to the large number of comments received, often with very similar content, the most common comments have been grouped together into meta-comments, which summarize individual points and statements. Below each statement in B.1.1 are the number of comments whose core was summarized by the meta-comment.

More complex and nuanced comments have been included and sometimes edited; spelling or major grammatical errors are fixed, external references to other subjects are annotated, and excessive profanity has been removed. Comments have intentionally not been screened for plausibility or factual correctness; all these comments, regardless of the truth of what they propose, are listed. This is done primarily to give a balanced image of each group that is hard to convey through purely the numerical results of the study. However, comments that contain essentially no information have been removed, for example, the hypothetical comment *"I like OSHW =)"* would be removed.

All changes to comments beyond trivial spelling changes have been marked by using italicized square brackets *[like this]*.

B.1.1 *Meta-comments*

- The ability to study the source files of OSHW devices (such as a schematic or board layout file) is one of the main benefits of OSHW. This simplifies debugging and increases the useful lifespan of the device as compared to a proprietary device, where the end of support from the developing company may effectively be a death sentence for the device.

  16 noncommercial users

  4 noncommercial developers

  3 Inexperienced users

- The "boring" parts of the development process (documentation, debugging, polish) can be hard to motivate people to do and are often left lacking.

  8 noncommercial users

  1 noncommercial developer

- OSHW devices often require more work on behalf of the user to make things work, which is sometimes a reason to use a proprietary device instead.

  5 noncommercial users

  1 noncommercial developer

- The ability to modify or re-use parts of OSHW devices in other projects is very important.

  6 noncommercial users

- OSHW devices in a better way include the features that the users actually find useful, which is not always the case for proprietary devices. The overlap between the users and developers and the community helps this to a large extent.

  2 noncommercial users

  2 noncommercial developer

- The ability for companies to take an OSHW design and turn it into a proprietary design, "ripping off" the initial developers, is a falling point of OSHW.

  4 noncommercial users

  1 noncommercial developer

- As OSHW devices have reduced marketing, development, and support costs the device cost can often be far lower.

  4 noncommercial users

- The community that develops OSHW is very important.

  4 noncommercial users

- Standardization between different OSHW projects can often be very low, making it difficult to combine different devices.

  3 noncommercial users

  1 inexperienced user

B.1.2 *Noncommercial Users*

- One major drawback is *[the]* stability and availability of OSHW compared to proprietary solutions. TI[a] is not that likely to go out of business compared to SparkFun.
  If all the guts are out there, community support can continue, but I always worry about making OSHW responsible for something mission critical.

a Texas Instruments, a large producer of IC's and electronic devices.

- Proprietary solutions are most of the time aimed at companies that make large quantities of products. They don't provide the kind of service a hobbyist is looking for: cheap, small volume, free supporting software, community to ask questions, lots of examples available on internet...
- Open systems often have better documentation than propriety systems if the propriety systems do not come from a mature and dedicated team. This is getting worse in recent years. I also find that Open systems tend to have a lot more chatter around them that makes finding real information hard. Arduino is a good example of this, 98% of the content on the web is too light to be informative and seems to drown out much of the insightful information.
- *[OSHW is]* sometimes difficult to use in *[a]* professional environment because there is no "official support". OSHW is sometime difficult to source in quantity.
- *[OSHW devices]* are usually not made in China as much (Arduino, Raspberry Pi, RepRaps...) and I can even make them myself from parts I choose to use, which is important to me.
- We need everyday gadgets like TV's and set top boxes to becomes OSHW. If I dont like the way my TV interface works, I should be able to change it. My wife has an electric car (Reva) and I would love to get my hands on a schematic.
- OSHW helps people (who are not necessarily engineers) to create new uses for technology. Their ideas eventually become a proprietary solution.

- OSHW currently lacks the volume purchasing power to resell at a price point for the end consumer *[that is]* low enough. *[As it is today,]* purchasing from the OSHW organization (person/group/company) *[that develops the device is more expensive]* than hunting around places like eBay for the parts and boards.
  I do understand that both have to make a profit to "keep the lights on" so to speak, neither the proprietary nor *[the]* OSHW *[developers]* have *[the]* means to *[keep someone like me]* come back for a second or third when the cost point is prohibitive. OSHW needs an option for selling that does not raise the price point over the cost of buying an item used and pulling out the parts needed to get an equivalent. A nice example is how cheap original Palm Pilots are used and instantly I have access to a pre-programed ARM based processor, touch screen. IR tranciever and web searched software base for almost any project to add to or use it as the center of the project with more documentation than I could ever want.
- They both can be good, but the engineering *[of OSHW devices]* needs to be good.
- OSHW tend to be easier to combine with other hardware and software, *[open source]* or proprietary. Usually *[this is]* because someone else has had the same or a similar idea. I think that is sort of the added value of OSHW.
- OSHW is better in the sense that it is free for anyone to build and use and therefore can help communities or students that don't have the money to buy proprietary solutions. The drawback is that with proprietary solutions for a business they provide a guarantee and fallback. For example if a controls system for a factory has a problem and kills someone. If it is OSHW the family would likely get no compensation as there is no one to sue due to the fact that it's open source. With proprietary you could sue the maker of the controls machine (Siemens, allen bradley etc) for making a product that is sub par.
- OSHW helps greatly in many ways, but I see too many projects avoid any organization, so they get little done, and what is accomplished isn't very useful. None of the big ones suffer from this (they wouldn't be big if they did) but I've seen many small projects with potential that die in this way.
- Much quicker response to feature change requests (e.g. RPi and the mounting holes)[a].

a The Raspberry Pi, a very small bare-bones computer-on-a-board, initially shipped without any screw holes for mounting the device; they were relatively quickly added after a discussion in the community where the benefits and broad desire of mounting holes became apparent.

- With OSHW I know I'm using all the power available in the hardware; unlike where a manufacturer intentionally cripples functionality in software unless you pay a fee to "unlock" features.

- *[Regarding]* OSHW, *[one could]* for example, bring *[a]* group of people that are capable of designing a WiFi router. Make sure the router is the best in the world and easily (like really easily) "hackable" *[to be able to]* go over *[the legal WiFi signal strength limit]* for example. Add a GPS sensor, GSM, tons of RAM, SD, USB, or RAMDRIVE storage backed by a battery or a super capacitor. Make the router very reliable, *[with a]* removable fan, *[replacable]* casing, *[with]* solar panels on the casing. Then together we can manufacture routers like that and kick Cisco's[a] *[profanity]* :-) *[Add]* DD-WRT[b] and we are ready to go! Give each router very good antennas and create a mesh network. Also make a mobile router unit that are installed in cars, so the cars are connected together, *[giving you]* internet anywhere you go and also *[the]* ability to dump music from home onto *[the]* car's music library through WiFi, or tap into someone's music while on highway who is near you; perhaps listen to what they listen, or have *[two-way]* communication between cars, and the gps and accelerometer *[could]* alert other cars with a sound or *[some]* other notification that the car ahead stopped really fast or simply state an emergency.
  OR! My very best and *[favorite]* idea that I would love to turn into a reality is a cellular network. *[Currently, the]* hardware that is used in *[cell-phone base stations]* is REALLY expensive. But if you get down to it it is really not that fancy and certainly doesn't cost as much as manufacturers ask for it. So by designing and manufacturing our own cellular network hardware we can set up *[the United States]* and Canada *[with a high-bandwidth,]* very cheap network with unlimited internet and calling! This will change the world!!! Then expand into Europe and make ONE global cellular network! With unlimited everything! I would LOVE to do this project!

  a A manufacturer of networking equipment, routers among others.
  b A popular OSS firmware that can replace the default firmware of some routers

- Many times I need a certified solution for a project. The available OSHW solutions are generally fewer *[than proprietary solutions]*.

- Proprietary products tend to stick with original mistakes. OSHW tends to change more freely, for any reason the community deems necessary, with less attachment to prior poor decisions.

B.1.3 *Noncommerical Developers*

- Security; it's a blind trust vs. truly knowing situation. Which applies to the whole range. From the private/consumer use, to security for the general population such as; voting machines, power plants, healthcare devices, IT infrastructure.

- It really depends on the domain. Even classifying projects as "simple" or "complex" is still vague. There is also the software infrastructure that allows the hardware to be used effectively, so that even if a (potentially open) hardware product is technically superior, if the supporting software isn't there or isn't as mature, it will not be as effective as a tool.
  For example, which one is "better" or "cheaper", an Arduino or a TI MSP430? The MSP430 is 1/4 the cost of an Arduino, but the supporting infrastructure, community, Linux toolchain and documentation pales in comparison to the Arduino.
  Stepper drivers are on the other end of the spectrum, where many quality OSH solutions exist that are very reasonably priced, but the software for a CNC toolchain is probably much better for Windows/proprietary workflow.
- I see the model as operating outside of the traditional venture capital channels for developing new products. This will make VC and the traditional finance community less important and return some control of the economy to users.
- Both OSHW and proprietary *[devices]* have their drawbacks, but I think they're pretty similar. It all comes down to support, whether you get any at all is down to the whims of the community, as much as it is down to the whims of a proprietary company, however, those that provide decent documentation and some kind of support network generally tend to fare better than those that don't.
- *[The]* biggest drawback is it can be hard to get support at times *[for OSHW devices]*, but even then it's usually easier to get support *[for]* them than through a proprietary vendor...
- With OSHW I can control the quality of the parts for a reasonable price, whereas consumer, off-the-shelf hardware tends to be mass-produced so they *[have to follow applicable]* standards, so it can *[become]* very expensive to get better quality stuff.
- With proprietary products, you get what you buy. With OSHW, you may not always get what you buy, but you always get what you make.
- OSHW is a great place to get ideas, and to improve ideas.
- OSHW typically has better documentation because the community won't allow undocumented progress.
- *[OSHW]* benefits society more and increases self-driven inventors/hackers to create, share and absorb all the appreciation from others who find their creation useful and adaptable to their specific needs. We need more OSHW on the market.
- *[OSHW]* has minor drawbacks with *[regards to]* documentation if the project is not adopted by many, but *[most often]* anything that has happend to one person *[has]* happened to *[someone else]*; that is the case with communities based on OSHW.

- You can always run open source software on OSHW, but closed hardware is often locked down so you can't modify the software on it. So from a software point of view, open hardware is often way better to work with.
- Although I like to see open source hardware, most of the time it only works on a very small scale — just a few units. Issues crop up that should've been addressed earlier. I regularly see incredibly poor pcb layouts lauded as great, short-sighted designs praised as commercial challengers, all only because they were released as open source. Releasing something as OSHW does not give one an excuse to release shoddy code and get praised. Meanwhile, another person designing a solid product but wanting to sell it to pay for huge time investments and closing source to prevent chinese clones is ignored. In more advanced designs (say 8 layer pcb with several 500+ ball BGA devices and controlled impedance) there may be literally nobody taking the source files and modifying it to build a result from that. There are many more practical considerations with hardware that the community in their blind acceptance of ad-nauseum breakout boards and glorified LED flashers don't see yet.
- OSHW is useful, but it seems *[to be]* only useful to those who possess a backround of related education or to those willing to struggle to acquire the education. In the many instances I have been directly involved, and especially when trying to get others interested, the lack of plain clear information available to a layman is disappointing and discouraging.
- I hope one day all things will be open, software, hardware, etc. I think if people really care about environment and waste, this should be the way to go. Fixing and upgrading existing devices, instead of throwing them away and buying new ones.
- In general, if the OSHW solution has a large user base, the support and documentation is quite good. The proprietary products mostly are clearer in what documentation is available, already before you buy the product, because it can all be found in one spot (*[the]* manufacturer's website), and when the manufacturer is known, you have some feeling for its quality. OSHW is more fragmented, which gives greater uncertainty of quality.

B.1.4 *Commercial Users*

- The hacker community isn't driven by a commercial model, so it's possible to find unique devices unserved by commercial means. It does mean a somewhat steeper learning curve, but the "price of entry" for virtually any technology nowadays includes that. Additionally, the OSHW community is where the future's engineers are being motivated and trained RIGHT NOW. These kids amaze me.

- IP can *[be]* an issue with OSHW if the licencing strategy isn't well thought out. The fact that you are not tied down a licence restriction is really useful (I.E. the Libre side of OSHW).
- At the basic end, OSHW is generally more likely to provide a very specific solution to any given problem where a large commercial venture would not be able to justify the effort of making multiple very specific devices.
  At the complex end, OSHW is adaptable to specific needs where a commercial solution relies on it being worth their while doing the R&D to cover your use-case. As an example, we previously used a Connii I2C tester which is quite expensive, now we have a BusPirate which is far more flexible and we can bend it to our needs.
  The flipside is that OSHW/SW documentation is often absolutely awful or non-existent, and in a commercial operation you cannot afford to spend weeks trying to get something to work - it's worth paying the premium for something closed-source that will work out of the box and is documented.
- In a professional environment, where time is money, my employer is more prepared to buy a relatively expensive solution that works from the start, than having me spending hours getting OSHW working just right. Buying a proprietary, well supported product is often the quickest way forward.

B.1.5 *Commercial Developers*

- OSHW is great for the community but has pros and cons for the developers. On the positive side it buys goodwill and exposure from the users but on the negative side you are giving away your labor for someone else to exploit. At the end of the day you have to feed yourself and family.
- I think Open Source and Open Hardware is the only way to go. It is the most efficient use of people's time and resources. The ability to accelerate innovation is so great that I think companies that embrace open source can leverage the crowd to be able to iterate and innovate quicker and cheaper than any other development path. If companies lose their ego and just make good products that are open source people will still stick with *[their]* brand because they are the originators *[of the device, developed]* it first and know why things work. However, I think there needs to be a new type of licences that *[is]* essentially open source but *[gives]* protections from someone copying it wholesale and undercutting the originators. There needs to be a way to ensure open hardware is being used in the spirit of open hardware.

- The question of OSHW in a commercial context is whether it can actually make a company viable. While there are a very very few companies that have made it work, there are those, like Makerbot, that have not. Makerbot likely wrapped up the Replicator 2 at the insistence of those funding them. While places like Adafruit and DIY Drones publish their designs, their commercial viability is in the little impetus the buyer has to go out and wait for their own board to be produced and sent back to them, or in soldering SMD components, or in having to buy the needed components when they could instead just buy the kit.
  Getting to the place of allowing everyone to have your design for free from the beginning is risky if you are trying to build a business from it. It takes a fair leap of faith; it's important to remember that the patent system is actually closely related to the Open movement. Patents must list prior art, and after the owner's right expire anyone is free to use the design. *[OSHW]* relies on the idea that everyone *[should]* be able to use your idea immediately to create their own. If their idea is better, they can be more successful than you with that product even though the idea was originally yours.
  This can cripple the newcomer if their idea is great but they don't get the word out (marketing). But it certainly pushes the state of the art more quickly in what could be argued a more socially beneficial manner.

B.1.6 *Inexperienced Users*

- OSHW is still in its infancy. If these projects can get past the initial *"nobody uses it, so nobody contributes"* problem, they'll be vastly better than proprietary solutions. Note Wikipedia's early years, for a similar situation.
  Benefits *[of OSHW]*:
  - Greater understanding
  - Easier to use in own projects / products
  - Cheaper
  - Sharing is caring

  Drawbacks *[of OSHW]*:
  - Sometimes not as polished when released as proprietary solutions are (altho OSHW eventually grows better)
  - Uses usually proprietary solutions like MCUs etc itself (which is usually reasonable though)

- Typically the challenges I face when trying to implement OSHW is the belief *[held]* by others that because it is open source it is somehow less secure and unreliable. While there are many projects out there *[where]* this may be true, I feel that the same research needs to be done no matter which route is chosen.
- There are no benefits and no drawbacks — it's all about lazyness. Why compose my own code for a microcontroller (of my *[choice]*), when there is *[the]* Arduino? Not because I want to save time, but because I'm a lazy *[profanity]*. You should know I'm against Open Source.
- *[Getting]* funding for OSHW projects seems like it could be difficult, which makes me truly sad.
- Everyone should have to work for a living, not sit back for the rest of their lives after one good idea. Copyright/patent law should give an individual (or *[a corporation]*) a set *[time]* limit to recover *[research and development]* costs, *[and]* then require ALL design schematics be place in multiple public libraries. Work for yourself, but think for mankind.
- I owned a Dingoo A320, which is not open source, but got hacked faaaar open very soon It died after 3 years of use, not bad for a 70$ device. The community is awesome and gave so many new uses to the device. And the hackers developed their own "Dream-Dingoo", the GCW Zero, which afaik is fully open source.
- The fun of building OSHW beats buying some mass produced part every single time.
  Much of the joy comes from the pride of building something with your own hands and know-how, even when following step-by-step instructions.
  That being said, you should probably administer this survey to non-hardware hackers to get a more accurate perspective on you your market.
- My only negative experience with OSHW is trying to fix an error and 20 people each have a different idea how it should work, due to their own modifications, and sometimes, none of the suggestions work.

C

PROPOSED DOCUMENTATION STYLE FOR OSHW DEVICES

RABID MANTIS

Cooling Controller

Very flexible field-configurable temperature regulator with hysteresis for heaters or coolers.

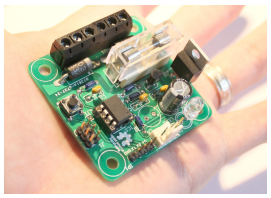
Features

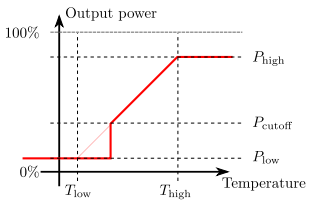
Controls both heating or cooling loads.
Directly switches up to 7A loads and/or relays.
Output switching rate controllable from 2kHz *to 60 seconds.*
Support for nearly any NTC thermistor and temperature range.
All system parameters configurable in-situ.
Hysteresis reduces perceived noise in load due to output power level changes.
Minimal functioning setup with only 15 through-hole components.
Failsafe mode that activates on thermistor or system failure.

Description

Cooling Controller is a general-purpose temperature regulator with one sensor and one output for either heating or cooling elements. All system parameters are configurable without the use of any external hardware, and are set with Morse code over a 1-button interface.
The device is directly capable of switching inductive loads of up to 7A, with an adjustable output rate varying from 2kHz to 60 seconds, and supports virtually any NTC thermistor and temperature range the thermistor is capable of surviving.
A guaranteed absolute temperature accuracy of up to $\approx \pm 1$°C is achievable, with a worst-case absolute error over a large range of ± 2.5°C.
A piecewise-linear control scheme with hysteresis allows for configuring the device in a wide range of schemes; from a simple ON/OFF thermostat with hysteresis to a linear ramp in output power with hysteresis and a minimum power cutoff level.

Overview





Output power over temperature when used with a cooling element (increased temperature results in increased power).

Contents

Copyright Notice

Hardware Revision History

1.1 Initial distributed version.

1.0 Initial version. Major flaws found, unusable.

Documentation Revision History

1.0.1 Minor grammatical changes.

1.0.0 Initial version, applies to hardware revision 1.1.

Absolute Maximum Ratings

PARAMETER	SYMBOL	RATING
Input voltage	$V_{in,max}$	30V[a]
Output current (continuous)	$I_{out,max}$	10A[b]
Operating temperature		-50°C to $+105°C$
Peak current to load, single 10ms pulse.	$I_{L,pk}$	100A[c]

[a] Assuming load tolerant of $V_{in,max}$.
[b] Output current is limited by PCB trace width.
[c] **This is untested!** Assuming F1 does not trigger, peak current is limited by Q1 and PCB trace heating.

Assembly

Cooling Controller is made to be easy to assemble with limited resources — only through-hole components are used — making it possible to assemble with only the most basic tools. The microprocessor does need to be programmed before use, which can be done using any programmer capable of programming the ATTINY85 over the ISP interface[1].

See the Bill of materials section for the components used and figure 7 for the component placement. The default component values will give a reasonable temperature resolution (better than 0.6°C over the entire range) and accuracy (ranging from ±1.2°C at 40°C to ±2.5°C at 100°C) at ranges of −35°C to +100°C (see figure 3 and figure 4). By changing two components the resolution and accuracy can be tuned to other operating ranges, as described in the Operation section. See the Pin Description section for a list of all input/output connections.

See the Changing Settings section for a description of how to set the firmware to the desired settings. Additionally, the Operation section describes the functionality of the unit (in both hardware and software) in more detail.

When assembling the unit, note that not all components are strictly required for use. In particular, the components that can safely be left unmounted are;

JP1 a pin header only used for debugging purposes during development. Allows for connection UART device for serial terminal support. *Note: the released version of the microprocessor firmware does not output any data to the header, nor does it accept any data input, due to code size limitations.*

J1 the 6-pin ISP header used for programming the ATTINY85. As this header is only used for programming the microprocessor once there is no need to solder the header to the board permanently — instead connect the pin-header to the programmer, insert the lower section of the header into J1's socket and lightly apply torque to the connector. This will usually give ample contact with all pins and allow for

[1] Programmers such as (among many others) the AVR ISPMKII (`www.atmel.com/tools/AVRISPMKII.aspx`), the USBtinyISP (`http://www.ladyada.net/make/usbtinyisp/`), or the Arduino as an ISP programmer (`http://arduino.cc/en/Tutorial/ArduinoISP`).

the one-time programming of the microprocessor. If J1 is mounted it is easy to accidentally reset the device during operation as the reset pin is exposed and easy to touch; if J1 is mounted avoid making contact with it.

C5 an additional decoupling capacitor that is not needed except for in extremely electrically noisy environments. If Cooling Controller spuriously restarts this component may be needed. Note that if mounted there is risk of damaging IC2 (the voltage regulator) if the input voltage supply is shorted during operating.

LED1/R2 a power status LED and current regulation resistor, these are not required if power status indication is not needed.

JP2 a standard 4-pin fan header. Not needed if the load is connected through the screw terminal. (Typically this pin header is only used if the load is a standard 3 or 4-pin computer fan). Note: The Bill of materials specifies two parts for this item, as the nominal part (Molex 47053-1000) is not commonly available. Instead a standard 3-pin header with plastic molding and a 1-pin header is specified, which together allow for 3 and 4-pin fan support, even though this is a kludge of a solution.

D1 acts as a reverse-polarity protection diode and can be left unmounted if this safety function is not needed. **WARNING: If left unmounted the system will be damaged by applying a reverse input voltage!**

Note that F1 may be any fast fuse rated for $\leq$ 8A.

Programming the microprocessor

Programming the microprocessor can either be done when mounted on Cooling Controller or in an external programmer. If programming the device when mounted on Cooling Controller, disconnect any loads loads nominally drawing over 1A[2] and any fan with a tachometer signal connected to JP2. Depending on the platform and programmer used the actual commands to program the device vary. If using AVRDUDE, a package available on most platforms (`http://www.nongnu.org/avrdude/`), the command to program the device's flash memory and set the device's fuse bytes is;

```
avrdude -pt85 -c<programmer> -P<port> -u -Uflash:w:device.hex:a
    -Ulfuse:w:0xc1:m -Uhfuse:w:0xcc:m -Uefuse:w:0xff:m
```

(Note that it is not necessary to write the EEPROM data, as the default values will be loaded from flash memory on the initial startup).

If using other programming tools, the important actions to perform are uploading the .hex file containing the software and specifying the fuse bytes as follows;

LOW FUSE	HIGH FUSE	EXTENDED FUSE
0xC1 (PLL clock, 14 CK + 4 ms startup time)	0xCC (SPI enabled, Watchdog always on)	0xFF

[2]The output pin driving the MOSFET transistor is tri-stated when in reset (and when being programmed) and may rise to an intermediate voltage, bringing the transistor into the current-saturation region and potentially heating the transistor beyond acceptable levels.

Pin Description

All connections to Cooling Controller are as follows;

HEADER	PIN	DESCRIPTION
X1	V+ IN	Positive voltage input.
	GND	Ground input.
	FAN +	Load positive voltage output. (Fan, heater, or otherwise)
	FAN -	Load negative voltage output. (Fan, heater, or otherwise)
	THERMISTOR A	Thermistor lead input.
	THERMISTOR B	Thermistor lead input. Connect any applicable cable shielding to this pin.
JP1	GND	Ground reference.
	TX	Device UART output.
	RX	Device UART input.
	V+	Internally regulated logic voltage, V_{logic}.
JP2	-	3pin/4pin fan connector.
J1	-	6-pin ISP connector.

Usage

In order to use Cooling Controller, assemble the device as per the directions in the Assembly section and connect incoming power, the load to control, and a thermistor to the device as per the Pin Description section.

Though great effort has been spent to ensure that Cooling Controller works as intended[3] there is always the potential for the device to operate erroneously and drive the load in a constantly on or off state. **Whenever severe damage may occur by the output being constantly on or off be sure to implement some type of secondary output protection — such as using thermal fuses for a heating load!**

Cooling Controller uses the load as a loudspeaker when altering the device settings, both on startup and if the internal settings become corrupted. **As such, the load must be able to withstand a 50% duty cycle square wave output at frequencies of 200 - 500 Hz.**

See the Electrical Characteristics section for typical input voltage ranges and load capability. As there is a flywheel diode integrated into the device relays can be directly connected to the load output without any additional components.

Before the device can be used it must be configured using the 1-button interface. See the Changing Settings section for instructions as to how to configure the firmware settings.

There are other parameters that may be of interest to modify in some applications, see the Operation section for a description of the electrical and software implementation and a list of parameters that may be useful to modify.

[3] Cooling Controller has two redundant copies of settings with independent CRC checksums stored in EEPROM, uses a permanently enabled hardware watch-dog timer, and has sturdy voltage stabilization with a brown-out detector.

Changing Settings

Cooling Controller's settings are set with a 1-button interface using the load as a loudspeaker to give feedback. If using a load that does not make any switching noise (such as a purely resistive load) attach a load that makes sound, such as a 3-pin PC fan.

All user settings are controlled by programming the device with Morse code over a single button (SW1). To enter programming mode; power up Cooling Controller , press and hold the button when the load starts making a sound, and finally release the button when the sound stops. If the device has correctly entered programming mode a short beep will be output after a pause. (Cooling Controller can not enter programming mode once it has started in its ordinary operating mode, it must be powered down and started up again with the button depressed during the startup tone).

Valid commands are listed in table 2 and described in more detail in the Parameter Description section. The codes listed in table 2 consist of three characters; "•" a dit character, "–" a dah character, and " " a pause. These are input to Cooling Controller by pressing the button briefly for a dit character, for a longer time (until the output tone changes) for a dah character, and by pausing and waiting for a brief beep for a pause character. Most commands have a parameter following the command which is input in the same way. All commands must be terminated with a period (".") character before Cooling Controller will attempt to interpret the data.

As an example, the full command to set the low temperature point to 5°C, "TL5.", is – • – • • • • • • • • • – • – • – in Morse code. This corresponds to one long press (*T*); a pause (waiting for a short beep from Cooling Controller); one short press, one long press, two short presses (*L*); a pause again; five short presses (*5*); a final pause; and three repetitions of a short press followed by a long press (.).

When a valid command has been input it is immediately saved and Cooling Controller will output the data it interpreted in order to ensure that the command was correctly parsed. Any invalid command, such as an incorrectly specified header and/or Morse input that doesn't match any letter, is ignored entirely. *If Cooling Controller does not output anything after a period "." symbol has been input then no action has been taken and all previous characters have been ignored.*

If the stored settings become corrupted[4] Cooling Controller will reload the default settings on startup and warn the user that the default settings have been loaded by outputting a constant tone[5] until the button has been pressed. If Cooling Controller warns about corrupted data there is a high probability that there are other problems in device, that may for example have been caused by exceeding the absolute maximum ratings! *Note: after programming the ATTINY85 from a host PC and starting up for the first time, Cooling Controller will warn that the settings are corrupted, this is normal behavior and occurs only on the first startup!*

After setting the system parameters, restart the Cooling Controller (and be sure not hold the button pressed while starting) to bring it into its temperature control mode.

[4] This is very unlikely in normal operation, see the Operation section.
[5] A $500Hz$, 50% duty cycle square wave output to the load.

Table 2: List of valid input command headers.

PARAMETER	CHARACTERS	MORSE CODE[a]	PARAMETER[b]
Reset to defaults	RS	• – • • • •	N/A
Low temperature point	TL	– • – • •	(x) °C
High temperature point	TH	– • • • •	(x) °C
Temperature hysteresis	TY	– – • – –	$(x/10)$ °C
Low temperature power level	PL	• – – • • – • •	(x) %
High temperature power level	PH	• – – • • • • •	(x) %
Cutoff power level	PC	• – – • – • – •	(x) %
4-pin fan mode enable	FN	• • – • – •	See Parameter Description
Failsafe mode	FS	• • – • • • •	See Parameter Description
Minimum switch period $[ms]$	SM	• • • – –	(x) milliseconds
Minimum switch period $[s]$	SS	• • • • • •	(x) seconds
Bias resistor R_b value	RB	• – • – • • •	(x) Ω
Thermistor R_0 value	R0	• – • – – – – –	(x) Ω
Thermistor β value	BT	– • • • –	(x)
Help	XX	– • • – – • • –	N/A

For example, setting the low temperature point to 5°C corresponds to the command "TL5.", which is "– • – • • • • • • • – • – • –" in Morse code.

[a]Here, a • corresponds to a dit symbol and is input with a short press, – is a dah and is input with a long press.

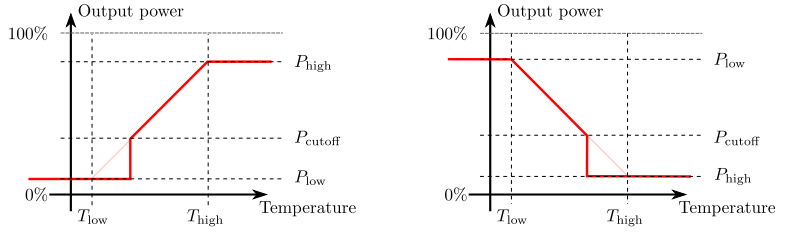
[b]Note: Most commands take an additional numerical argument, all commands are terminated with a "." character, • – • – • – in Morse code.

NUMBER	MORSE CODE	NUMBER	MORSE CODE
0	– – – – –	6	– • • • •
1	• – – – –	7	– – • • •
2	• • – – –	8	– – – • •
3	• • • – –	9	– – – – •
4	• • • • –	- (negative)	– • • • • –
5	• • • • •	. (period)	• – • – • –

(a) Other Morse code symbols.

Table 3: Default parameter settings.

PARAMETER	DEFAULT	PARAMETER	DEFAULT	PARAMETER	DEFAULT
TL	40	PH	100	SH/SM	10 (ms)
TH	60	PC	20	RB	33000
TY	50	FN	0	R0	33000
PL	0	FS	1	BT	4090



(a) Temperature and power relationship for a cooling load.

(b) Temperature and power relationship for a heating load.

Figure 1: Temperature and power parameters for cooling and heating loads.

Parameter Description

Control settings

Shown in figure 1 is the fundamental behavior of Cooling Controller. At temperatures below T_{low} the output power level is P_{low}; at temperatures above T_{high} the output power level driving the load is P_{high}, and at temperatures in-between the power level is linearly interpolated. In some cases (such as when controlling a fan) there is a minimum useful power level which can be set by P_{cutoff}, which sets the output power to the lowest of P_{low} and P_{high} when the linearly interpolated value is below P_{cutoff}.

An additional parameter T_{hyst} sets the measured temperature's hysteresis, which adds a degree of delay to the system. For example, if T_{hyst} is set to one degree and $T_{low} = T_{high} = 25°$ the output will need to reach 25.5° before the output power level is changed to P_{high} and fall below 24.5° before the output power level is changed to P_{low}. At temperatures in the range $24.5° - 25.5°$ the output level will be constantly P_{high} if the temperature was most recently above 25.5° and constantly P_{low} if the temperature was most recently below 24.5°.

Parameters can be "mangled" to a large extent, P_{high} can be lower than P_{low} (which is what is needed for a load that heats the sensor, as shown in figure 1b). If P_{cutoff} is not needed it can be set to zero. If $T_{low} \geq T_{high}$ the output will be P_{high} for temperatures greater than T_{high} and P_{low} for temperatures lower than T_{high},; IE. the system will behave like a thermostat, without any linearly interpolated region (in this case a nonzero T_{hyst} is useful to reduce the switching frequency).

Temperature values are expressed in degrees Celsius except T_{hyst} which is expressed in tenths of degrees Celsius; IE. an input value of 15 corresponds to a hysteresis of 1.5°. Valid values are whole degrees Celsius from absolute zero to several tens of thousands of degrees, making the useful range limited only by the thermistor specification. Note that the default component setup and tolerances results in an accuracy range as shown in figure 3 and figure 4.

The output power levels to the load are expressed in %, where 0% corresponds to the load being completely off and 100% corresponds to the load being completely on.

Load settings

These settings relate to the type of load connected to Cooling Controller. **FN** sets the type of load; if using a four-pin PC fan with a PWM input connected to JP2 set this parameter to one. If using any other type of load set this parameter to zero. (A four-pin PC fan will typically make slightly less noise than an equivalent 3-pin fan). **FS** sets the failsafe mode for the load — if the sensor fails or Cooling Controller crashes the output will be set to a failsafe state. *This is not normal behavior and will typically never occur.* Set this to zero to turn off the load when in the failsafe state and set it to one to turn on the load when in failsafe mode.

The two parameters **SM** and **SS** control the minimum output periods for the output. Due to the output modulation scheme used the load will experience a varying PWM frequency depending on the power level[6]. These parameters allow for setting the minimum time between load on/off transitions. **SM** specifies the time in milliseconds, while **SS** specifies the time in seconds. The two commands are otherwise identical and effectively control the same system parameter — the most recently specified value of either **SM** or **SS** sets the minimum switching time. Set to zero to switch at T_{min}, which is as fast as possible (see the Electrical Characteristics section). The maximum allowable value is 60 seconds (60000 milliseconds). Keep in mind the minimum time between load on/off transitions is strictly enforced — regardless of any temperature change the output power level will not change until the minimum time has expired. Conversely, if there is a short minimum time between load on/off transitions and low or no hysteresis there is the potential for the *average* power level to be below P_{cutoff} if the temperature fluctuates close to the cutoff point.

Sensor settings

As Cooling Controller supports nearly any thermistor[7] the sensor used must be specified. **RB** sets the bias resistor R_b (R3) used, expressed in a whole number of Ohms [Ω]. Similarly, **R0** sets the thermistor reference resistance[8], also expressed in a whole number of Ohms [Ω]. **BT** specifies the thermistor's β parameter, expressed in whole numbers.

Miscellaneous settings

RS resets the entire device to the default settings in table 3, and **XX** gives generally helpful advice to the user.

Examples

See examples 1,2, and 3 for a few usage-cases for Cooling Controller and the settings used for them. The initial reset is not strictly needed, but ensures that the default settings are used for the unspecified parameters.

[6]High and low power levels give a low PWM frequency, while a 50% power level gives a high frequency.
[7]See theElectrical Characteristics section for limitations.
[8]Defined as the resistance at 25°C.

Setting Mnemonics

All the input settings to the device consist of two letters to define the parameter to change, with an optional argument, followed by a period (a "." character). All the valid input commands are;

RS Reset the entire unit to the default parameters. No additional argument.

TL Sets the lower temperature control point. At temperatures below the argument the output power will be **PL**.

TH Sets the upper temperature control point. At temperatures above the argument the output power will be **PH**.

TY Sets the temperature hysteresis **expressed in tenths of degrees Celsius**. After a temperature change in one direction the temperature must change by the argument's amount in the other direction before the output power level will change.

PL Sets the output power at the low temperature level. At temperatures below **TL** the output power will be the value specified.

PH Sets the output power at the high temperature level. At temperatures above **TH** the output power will be the value specified.

PC Sets the power cutoff level. When the temperature is between **TL** and **TH** the power level is linearly interpolated. When the interpolated power output level is below the argument the power level will be set to **PL**. Set to zero to disable this functionality.

FN Sets the load type, set to zero if using a load connected to the terminal block or if using a 3-pin fan. Set to one if using a 4-pin computer fan (with PWM input) connected to JP2.

FS Sets the failsafe mode. Set to zero to turn off the load when in the failsafe state, set to one to turn on the load when in the failsafe state.

SM Sets the minimum time between output on/off transitions in milliseconds.

SS Sets the minimum time between output transitions in seconds. This is identical to specifying **SM** with an argument 1000 times greater.

RB Set the bias resistor (R_b) value used expressed in Ohms [Ω].

R0 Set the thermistor reference resistance (R_0) value used expressed in Ohms [Ω].

BT Set the thermistor beta (β) value used.

XX Use only when in dire need, no arguments needed.

Example 1 Set the device to use a 4-pin fan connected to JP2 (switching at as high a frequency as possible), turning on when the temperature exceeds 35°C, reaching maximum power at 50°C, with a 3°C hysteresis, and otherwise using the default settings.

```
RS.
TL35.
TH50.
TY30.
FN1.
SM0.
```

Example 2 Set the device to drive a refrigerator (through a relay connected to the screw terminal X1), turning it on when the temperature rises above 5°C and turning it off when falling below 3°C, switching at most every 30 seconds, and otherwise using the default settings.

```
RS.
TL4.
TH4.
TY20.
PL100.
PH0.
PC0.
SS30.
```

Example 3 Set the device to drive a resistive heater directly connected to the screw terminal, with maximum power when under 50°C, linearly ramping down to 25% power when at 60°C without any hysteresis, using a thermistor / bias resistor pair with parameters $R_0 = 22k\Omega$, $\beta = 4220$, $R_b = 8k2\Omega$, and otherwise using the default settings.

```
RS.
TL50.
TH60.
TY0.
PL100.
PH25.
PC0.
R022000.
BT4220.
RB8200.
```

Bill of materials

Note; A value of 4n7 corresponds to a value of 4.7n, and in the case of a capacitor corresponds to 4.7nF. The suggested part number is only that — a suggestion — and may be replaced with any other equivalent matching the specifications listed under value, rating, and type. Note that high-accuracy resistors are only chosen for R2,R4,R5 due to (typically) reduced minimum order quantities.

COMPONENT NAME	VALUE	RATING	TYPE	SUGGESTED PART NO.
C1,C3,C4	100n	50V	X7R ceramic, 7.5mm pin grid	K104K15X7RF5TH5
C2,C5	100u	50V	8mm can, 3.5mm pin grid	50ZLH100MEFC8X11.5
D1,D2	SR303	30V, 3A	Generic power Schottky diode	SR303
F1	$\leq 8A$	Fast acting fuse	5x20mm fuse holder and fuse.	MCHTC-15M & MCF05G-8A
IC2	LM317LZ	20mA	TO-92 package	LM317LZ
IC1	ATTINY85-20PU		Atmel AVR Tiny series	ATTINY85-20PU
J1			2.54mm x 2-row, 6-way pin header	M20-9980646
JP1			2.54mm x 1-row, 4-way pin header	M20-9990446
JP2		4-pin fan header	2.54mm x 1-row, 3-way pin header & 2.54mm x 1-row, 1-way pin header	MC34631 & 90120-0761
LED1	Generic LED	$\geq$5mA	5mm body, 2.5mm grid	L-9294SYCK
Q1	IRL7833PBF	4mΩ, $V_{DSS} \geq 30V$	Low RDS_{on} at 4.5V, TO220 package	IRL7833PBF
R2,R5	2k8	1%, 250mW	7.5mm grid	YR1B2K8CC
R3	Default: 33k	0.1%, 250mW	7.5mm grid	YR1B33K2CC
R4	1k	1%, 250mW	7.5mm grid	YR1B1K0CC
SW1	B3F1020	NO tactile switch		B3F1020
Thermistor	Default: NTCLE100E3333JB0 $R_0 = 33k \pm 5\%$, $\beta = 4090 \pm 1.5\%$		Generic leaded thermistor	NTCLE100E3333JB0
X1			5mm x 6-way terminal	CTB5202/6

Mechanical Description

Cooling Controller is takes up a space of 50mm by 50mm, with a build height of 23mm, limited by the transistor Q1. Mounting holes are listed in table 6 following the coordinate system shown in figure 2. PCB manufacturing requirements are shown in table 5 and should be generally achievable at any PCB house.

If using a high-powered load be sure to use cabling with sufficient conductor area for both the load and incoming power. If Cooling Controller is fused for 8A a wire of $1mm^2$ should be sufficient for most insulation types. If the distance from Cooling Controller to the thermistor is large and the ambient environment is electrically noisy[9] a shielded cable may be useful. If so attach the cable's shield to the negative thermistor input terminal.

Place the device in a position so that SW1 can be accessed, as this button is used to configure the system settings.

The power MOSFET transistor (Q1) does not require any heat-sinking, however the metal tab is electrically connected to the negative load terminal, which switches between GND and $V+$. Due to this, **ensure the metal tab on Q1 does not make contact with any other electrically conductive parts.**

Though the device does not require any heat-sinking or active cooling, some passive air-flow must be allowed when driving large loads (as the power MOSFET transistor (Q1) and the traces on the PCB may warm slightly).

Table 5: PCB manufacturing requirements.

PARAMETER	REQUIREMENT	UNIT
PCB thickness	Any (nominal 1.6)	mm
PCB layers	2	-
Copper fill thickness/density (minimum)	35/1	µm / oz/ft^2
Trace isolation (minimum)	0.254/10	mm/mil
Trace width (minimum)	0.4064/16	mm/mil
Trace to board edge (minimum)	0.25	mm
Drill to board edge (minimum)	0.504	mm
Drill diameter (minimum)	0.3	mm
Via annular ring (minimum)	0.254/10	mm/mil

[9] Where a noisy environment in this contect induces a ripple of over 5mV on the positive thermistor input terminal when measured with an oscilloscope.

Table 6: Mounting hole locations.

HOLE DIAMETER [mm]	POSITION X [mm]	POSITION Y [mm]
4.1	5	5
4.1	5	45
4.1	45	5
4.1	45	45

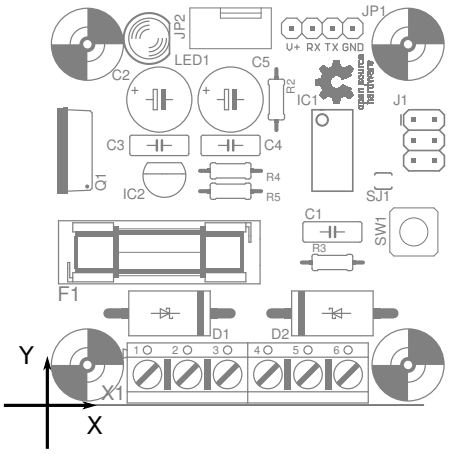


Figure 2: Coordinate system for mounting holes.

Electrical Characteristics

$V_{in} = 12$V, $T_{ambient} = 25°C$, Q_1 mounted with IRL7833PBF transistor unless otherwise specified.

PARAMETER	SYMBOL	TEST CONDITIONS/-COMMENTS	MIN	TYP	MAX	UNIT
Input Characteristics						
Power Input voltage	V_{in}		6		28	V
Quiescent supply current	I_q			14		mA
Output Characteristics						
Continuous current to load	I_L	Limited by PCB trace heating	8			A
Maximum switch rate		**SM** parameter set to zero		2		kHz
Sensor Characteristics						
Bridge output resistance		Resistance of thermistor and bias resistor (R3) in parallel[10].			100	kΩ
Sensor failure threshold, low[11]	$V_{t,l}$		$0.008 \cdot V_L$		$0.011 \cdot V_L$	V
Sensor failure threshold, high[12]	$V_{t,h}$		$0.989 \cdot V_L$		$0.992 \cdot V_L$	V
Internal Characteristics						
Logic supply voltage	V_{Logic}	Voltage supplied to ATTINY85 and analog reference voltage.		4.75		V

[10] Note that the thermistor's resistance changes with temperature, though as long as $R_{bias} < 100k\Omega$ this condition will always be satisfied.
[11] At bridge voltages below $V_{t,l}$ Cooling Controller will enter failsafe mode.
[12] At bridge voltages above $V_{t,h}$ Cooling Controller will enter failsafe mode.

Typical Performance

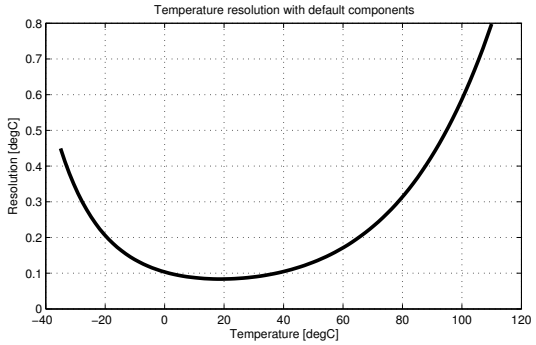


Figure 3: Temperature resolution with default component values.

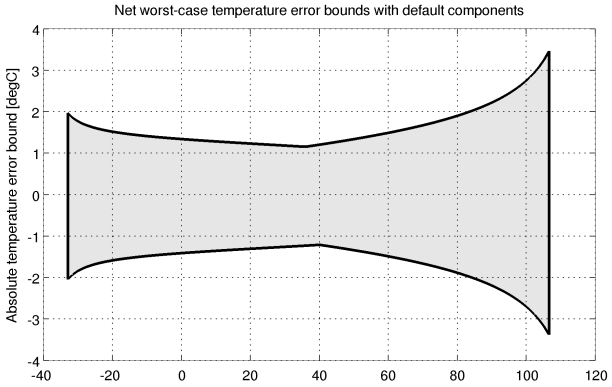


Figure 4: Worst-case absolute temperature error with default components.

Operation

Cooling Controller is electrically very simple, and has only the very most fundamental components — nearly all of the complexity of the device is in its software. Cooling Controller's full schematic is shown in figure 5, the PCB in figure 6 and component placement in figure 7.

Electrical operation

The screw terminal (X1) supplies the device with power through F1 which protects the device from over-current. D1 protects the device from reverse-polarity by shorting the input and triggering F1 in the event of a reversed polarity input. C2 and C3 decouple the incoming voltage, which is linearly regulated down to V_{Logic} by IC2, R4, R5, C4, and optionally C5.

The regulated voltage V_{Logic} supplies the microcontroller IC1 and drives the status indicator LED1, R2. IC1 is programmed through J1, and is directly connected to several components. SW1, the user interface button, is directly connected to the input and biased with an internal pull-up resistor. R3, C1, and the thermistor form a simple half-bridge with low-pass filter, which is fed to an ADC input. As can be seen in figure 6, a star topology is used both for ground and power for the half-bridge as well as the high-power and signal processing sections of the PCB, reducing the level of interference caused by transient voltage differentials.

Q1 switches the load on and off while the flyback diode D2 ensures that inductive loads can be switched without inducing large back-EMF transients. JP2 offers support for standard 3-pin and 4-pin PC fans. Though there is hardware support for the standard fan tachometer input on 3-pin and 4-pin fans (there is a trace from the tachometer output to the microprocessor input) this is not used at all in the software. JP1 is a debug header used only during development and allows for bidirectional UART transmission of data, though the released version of software neither sends nor parses any data input to this header.

Software operation

The software for Cooling Controller is written in C and distributed as an Eclipse[13] project file. Compiling and building the binary file requires the free packages avrdude, binutils-avr, gcc-avr, and avr-libc; these are available for all major platforms. Several supporting libraries are used with a relatively simple main program which performs the Cooling Controller-specific functions. The libraries used are;

./button/ performs button debouncing and is used to check the button status.

./dbg_putchar/ is only used during development and allows for basic debug status messages to be output from the device over a UART interface.

[13] A free development environment available at `http://www.eclipse.org`.

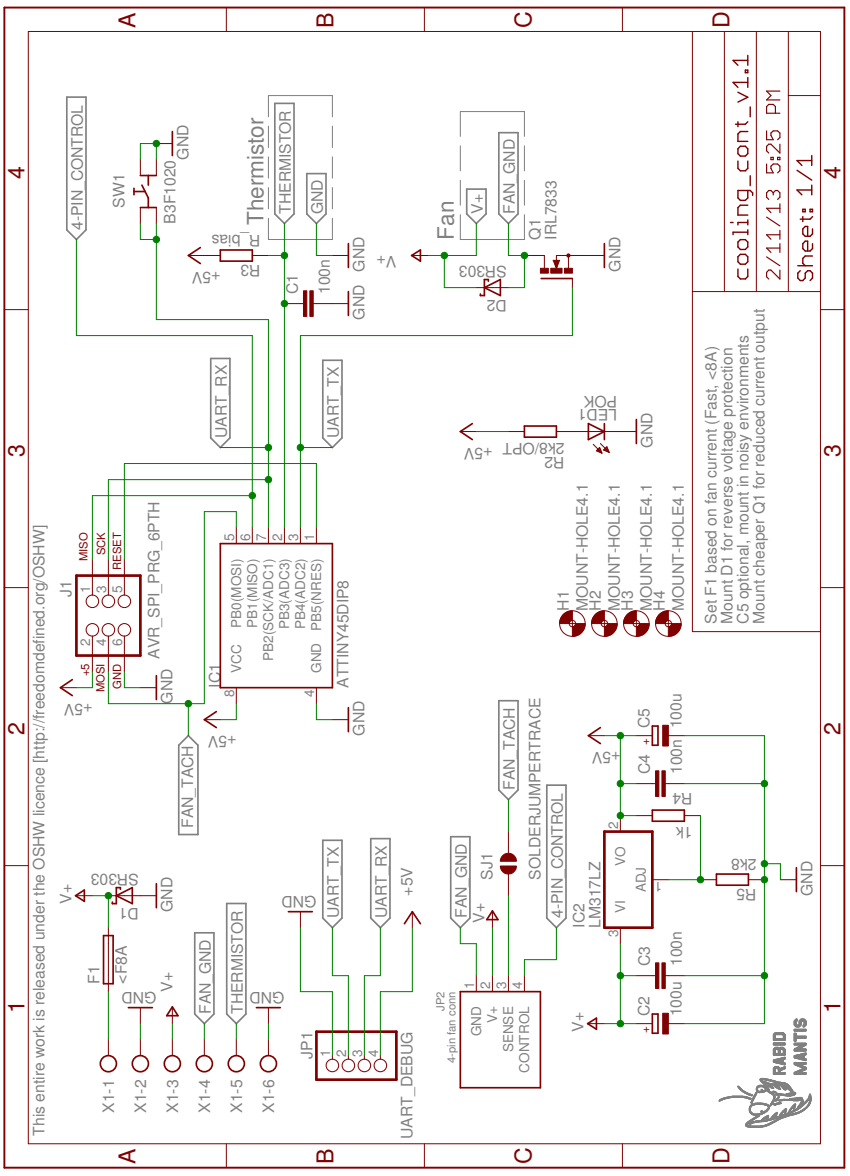
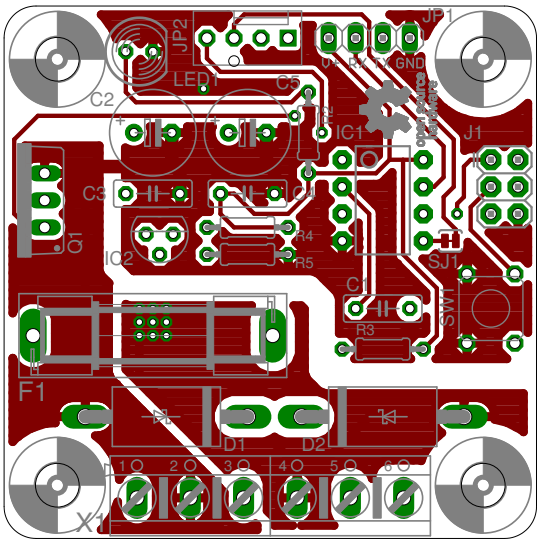
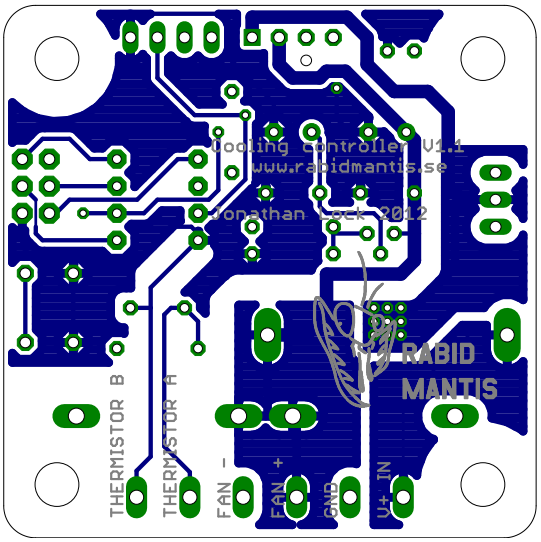


Figure 5: Cooling Controller schematic.

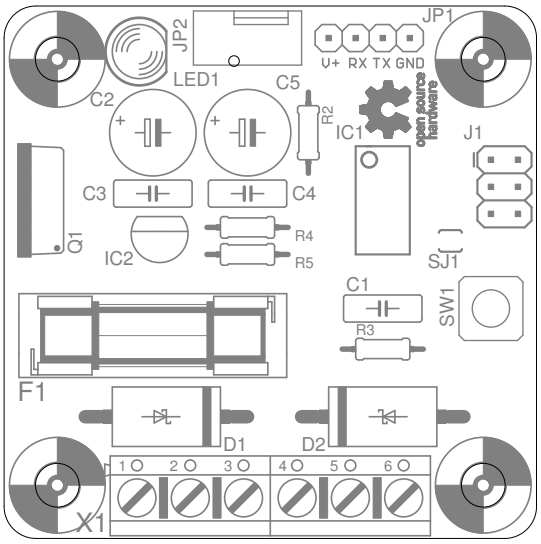


(a) Complete top layer as seen from above.

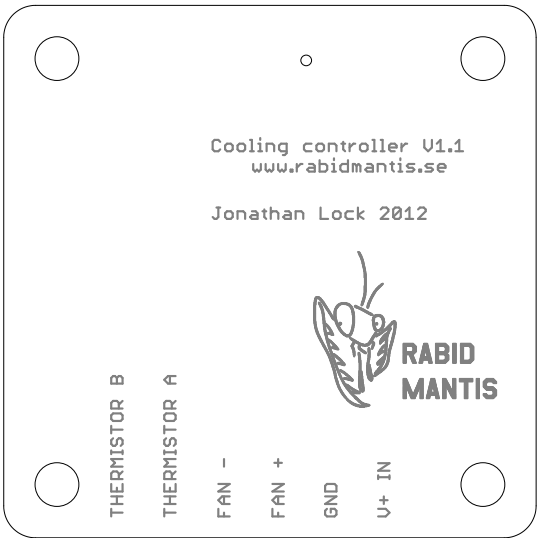


(b) Complete bottom layer as seen from below.

Figure 6: PCB details.



(a) Top layer component outline as seen from above.



(b) Bottom layer component outline as seen from below.

Figure 7: Component placement details.

./eeprom/ read/write functionality to non-volatile parameters (user settings) with CRC checksums and redundant data copies.

./magic/ does some magic.

./morse/ performs all Morse input and output parsing using streams.

./pwm/ drives Q1 (the switching transistor) with a sigma-delta modulation scheme while respecting minimum on/off times.

./ssched/ a simple cooperative multitasking scheduler that drives button, magic, Morse, and pwm functions periodically.

./thermistor/ reads the ADC voltage from the half-bridge (R3,C1, and thermistor) and calculates the thermistor temperature.

gpio.h & portpins.h gives easy bit-banging input/output functions.

The application-specific code is located in the few remaining files, as described below;

globals.h contains many of the Cooling Controller-specific compile-time constants.

input_commands.h/.c contains all the user-configurable parameter headers and their corresponding function calls.

tasks.h/.c contains all the tasks to run with the ssched library, most importantly the input parameter parsing function and the temperature-to-output-power conversion function.

main.c initializes the hardware, checks for and warns in the event of invalid EEPROM data, adds the tasks to the scheduler and checks the button status on startup. In the event of the button being pressed the device starts the Morse subsystem and enters an infinite loop where the user can configure settings; otherwise the device will start up the PWM subsystem and enter an infinite loop where the thermistor temperature is checked and the output power is set according to user settings.

The source files are compiled with GCC (4.6.3) to a file with very little free space on the microprocessor, on the order of a few free bytes. This is done with the optimization flags

```
-Os -fpack-struct -fshort-enums -mcall-prologues -fno-split-
   wide-types -funsigned-char -funsigned-bitfields
```

in order to bring the binary file size to something that will fit on the ATTINY85.

Temperature and ADC input relationship

The sensing element used for Cooling Controller , an NTC resistor, is a device whose electrical resistance varies greatly with temperature as per the generic relation;

$$R_{thermistor}(T) = R_0 e^{\beta\left(\frac{1}{T}-\frac{1}{T_0}\right)} \quad (1)$$

where R_0 is the resistance at temperature T_0, T is the current temperature and β is a device-specific parameter (with temperatures in Kelvin). The NTC resistor is connected in a simple voltage-divider, giving an input voltage to the ADC of

$$V_{ADC} = V_{supply}\frac{R_{thermistor}}{R_{bias}+R_{thermistor}}. \quad (2)$$

Solving (1) and (2) for the temperature with a measured ADC voltage gives the relations

$$T = \frac{\beta \cdot T_0}{\beta + T_0 \ln\left(\frac{R_{thermistor}}{R0}\right)} \quad (3)$$

$$R_{thermistor} = R_{bias}\frac{V_{ADC}}{V_{supply}-V_{ADC}} \quad (4)$$

which is a closed-form and relatively simple relationship between the ADC voltage and the measured temperature. This does however require the use of the $\ln(\ldots)$ function, which is relatively large in code size, as well as requiring floating point number support with the standard math library. This is one of the main reasons for the large code size of the software.

Temperature accuracy and resolution

The absolute accuracy of the temperature measurement is primarily limited by the tolerance of the thermistor used, the ADC and bias resistor only marginally contribute to the accuracy. A very crude MATLAB script for calculating the worst-case accuracy error and resolution is shown in Section A. By altering the parameters `Rb`, `Rb_tol`, `R0`, `R0_tol`, `Beta`, and `Beta_tol` the temperature accuracy and resolution can be tuned to different temperature regions.

A. MATLAB script for determining worst-case temperature accuracy and resolution

```
clc
clear all
close all
format short eng

% Calculate the temperature accuracy of Cooling Controller for a given
% setup and temperature range.

% User configurable settings

Rb = 33e3;     % Bias resistor [ohm]
Rb_tol = 0.001;  % Bias resistor tolerance
R0 = 33e3;       % NTC reference resistance [ohm]
R0_tol = 0.05;  % NTC reference tolerance
T0 = 25;         % NTC reference temperature [degC]
Beta = 4090;     % NTC beta value
Beta_tol = 0.015;     % NTC beta tolerance

ADC_err = 1.5;  % ADC tolerance (in bits)

Tl_plot = -35;  % Lower temperature point for plot
Th_plot = 110;  % Upper temperature point for plo

plot_pts = 1e2;       %number of data points in plot

% Do not modify below!

abszero = 274.15;
k2c = @(k) k - abszero;
c2k = @(c) c + abszero;
kADC = 1024;     % ADC resolution
ADC_tol = ADC_err / kADC;

temp = linspace(Tl_plot,Th_plot, plot_pts);

% Thermistor resistance at some temperature
Rtherm = @(t,R0,Beta) R0 * exp(Beta * (1./c2k(t) - 1/c2k(T0)));

% ADC input at some temperature
ADC = @(t,R0,Beta,Rb) Rtherm(t,R0,Beta) ./ (Rb + Rtherm(t,R0,Beta)) * kADC;

% partial derivative of ADC input wrt. temperature
dADC = @(t,R0,Beta,Rb) abs(-Beta * Rb * Rtherm(t,R0,Beta) ./ (c2k(t).^2 .* (Rtherm
    (t,R0,Beta) + Rb).^2) * kADC);

% ADC resolution for some temperature
res = @(t,R0,Beta,Rb) 1./dADC(t,R0,Beta,Rb);

%Super ugly and crude (not quite) exhaustive combination of all worst-case
%error situations. Rb_tol direction manually tested and set to worst sign.
ADC_1 = ADC(temp,R0*(1+R0_tol),Beta*(1+Beta_tol),Rb*(1-Rb_tol)) * (1+ADC_tol);
ADC_2 = ADC(temp,R0*(1+R0_tol),Beta*(1-Beta_tol),Rb*(1-Rb_tol)) * (1+ADC_tol);
ADC_3 = ADC(temp,R0*(1-R0_tol),Beta*(1+Beta_tol),Rb*(1+Rb_tol)) * (1-ADC_tol);
ADC_4 = ADC(temp,R0*(1-R0_tol),Beta*(1-Beta_tol),Rb*(1+Rb_tol)) * (1-ADC_tol);
ADC_nom = ADC(temp,R0,Beta,Rb);    % Ideal ADC value

% Endpoints for error computation (ADC values)
```

```
start = min([ADC_1(1),ADC_2(1),ADC_3(1),ADC_4(1)]);
stop = max([ADC_1(end),ADC_2(end),ADC_3(end),ADC_4(end)]);

start_temp = interp1(ADC_nom,temp,start,'spline');
stop_temp = interp1(ADC_nom,temp,stop,'spline');

err_temp = linspace(start_temp,stop_temp, plot_pts);

err_pts = linspace(start,stop, plot_pts);

ADC_interp = interp1(ADC_nom,temp,err_pts,'spline');
err_1 = interp1(ADC_1,temp,err_pts,'spline') - ADC_interp;
err_2 = interp1(ADC_2,temp,err_pts,'spline') - ADC_interp;
err_3 = interp1(ADC_3,temp,err_pts,'spline') - ADC_interp;
err_4 = interp1(ADC_4,temp,err_pts,'spline') - ADC_interp;

err_tot = [err_1;err_2;err_3;err_4];
err_min = min(err_tot,[],1);
err_max = max(err_tot,[],1);

figure(1)
set(0,'DefaultAxesColorOrder',[0 0 0],...
      'DefaultAxesLineStyleOrder','-|--|-.')
h = plot(temp,res(temp,R0,Beta,Rb));

grid on
title('Temperature␣resolution␣with␣default␣components')
xlabel('Temperature␣[degC]')
ylabel('Resolution␣[degC]')

set(gcf, 'PaperPosition', [0 0 7 4]);
set(h(1),'linewidth',3);

print('-depsc','resolution.eps')

figure(2)
set(0,'DefaultAxesColorOrder',[0 0 0],...
      'DefaultAxesLineStyleOrder','-|-|-.')

X = [err_temp,fliplr(err_temp)];
Y = [err_min,fliplr(err_max)];

h = fill(X,Y,'k','FaceAlpha',0.1);

grid on
title('Net␣worst-case␣temperature␣error␣bounds␣with␣default␣components')
xlabel('Temperature␣[degC]')
ylabel('Absolute␣temperature␣error␣bound␣[degC]')

set(gcf, 'PaperPosition', [0 0 7 4]);
set(h(1),'linewidth',2);

print('-dpng','temp_error.png','-r300')
```