# UNDERSTANDING THE MOTIVATIONS OF OPEN-SOURCE HARDWARE DEVELOPERS: INSIGHTS FROM THE ARDUINO COMMUNITY

Alexander Jaspers

Dissertation submitted in partial fulfilment of requirements for the degree of

International MSc in Business Administration

at

Universidade Católica Portuguesa

7 August 2014

## Abstract

Following a change in the paradigm of innovation, much research has been focused on open innovation and user entrepreneurship. Most of this research is centered on open-source software communities as a main driver and tool for innovation initiatives. Moving towards physical objects instead of merely considering software, the emerging topic of open-source hardware deserves closer attention.
This study aims to elicit the motivations of open-source hardware developers, in order to develop an understanding of how to manage innovation initiatives in this field. Focusing on the most prominent open-source hardware prototyping platform, Arduino, the results of a survey among community members indicate that developers are mostly motivated by intrinsic and internalized extrinsic factors. Moreover, an interview with a practitioner in the field of open-source hardware reveals interesting insights that can be used to develop a corporate environment conducive to innovation. Focusing on innovation both from a corporate and individual point of view, this study presents a number of strategies that can be employed to optimize undertakings in the field of open-source hardware.

*I would like to express my sincere gratitude to my advisor Prof. Prandelli, who has supported me throughout the entire process of writing this thesis with great insights into the exciting world of open innovation.*

*I owe my deepest gratitude to my family, for always supporting me on all my endeavours.*

# 1. Introduction

With the advent of new technologies, and most prominently the creation and widespread use of the Internet, both personal and business environments have changed dramatically over the last decades. With regard to the business environment, we can now witness a change in the paradigm of innovation. While innovation in the past has mostly been fostered behind the walls of corporate research and development departments, these walls have now disappeared due to the new logic of open innovation (Chesbrough, 2003).

On the forefront of open innovation is the development of open-source software (OSS), which is software that is free to all and mostly developed in Internet-based communities. Open-source software has a large impact on economy and society alike (Von Hippel & Von Krogh, 2003). With the benefit of being very accessible, open-source software offers the advantage of allowing for open innovation and possible user entrepreneurship.

Whereas the development of open-source software in managing and organizing innovation has been studied extensively, less light has been shed on an emerging subject in innovation management, namely that of open-source hardware. Due to new possibilities in manufacturing hardware and the popularization of these technologies, open-source hardware becomes more and more accessible to everyone. Another factor that has led to the widespread of open-source hardware is that costs have decreased exponentially, making it accessible to a wide audience. Often times open-source enabled hardware comes in the form of enhanced everyday items, an example of which is the “Pebble”-watch, which is a fully programmable watch that interacts with smartphones, and may be considered a smart object (Swan, 2012). On a more abstract level, open-source hardware might come in the form of microcontrollers or dedicated computing platforms that serve as a platform for creating smart objects. Recent examples for these types of open-source hardware are most prominently the Arduino microcontroller platform, which is fully open-source, and the Raspberry Pi, which is open-source with minor exceptions. Both platforms allow the user to develop solutions to technological problems, and are nowadays widely used to develop audio equipment, networking equipment and wearable technology, among others (Prandelli & Verona, 2012). Selling at prices below 50 US$ these devices clearly present an affordable option for innovators around the globe.

Selling at a low price, they offer everyone a platform to start innovating and creating software that matches the users' ideas.

Open-source software and open-source hardware rely on a user base that actively exchanges ideas and solutions, which figuratively takes place on a "bazaar" (Raymond, 2001). However, they can and should not be compared as equals as they differ in a variety of aspects.

The most prominent difference between open-source software and hardware is the relation between developer and producer. While in open-source software the developer is at the same time the producer of the software, this is not necessarily true for open-source hardware. Due to possible significant production costs, the developer cannot always produce open-source hardware and hardware parts in-house. Most often, the developer has to resort to production facilities, which give reality to his ideas. Only after having physically produced hardware it becomes usable for the developer.

Another difference lies in the ability to collaborate. While collaboration does not require significant resources for software developers, it can be more arduous for collaborating open-source hardware developers. In open-source software, several users can contribute to the development of software on a bazaar by supplying snippets of code. For open-source hardware, however, the scenario is completely different as contribution has to take place physically, e.g. by the contribution of different physical parts and items, which might result in substantial costs for all parties involved.

Recognizing these differences, it becomes evident that it does not suffice to merely apply literature on open-source software to hardware of this kind. Instead, more light should be shed upon the differences between software and hardware and the implications of these on the management and organization of innovation, and especially the emerging topic of user entrepreneurship.

Most research on open-source software can be grouped into one of three categories, which deal with motivations and contributions; governance, organization and innovative processes and competitive dynamics in open-source software (Von Krogh & Von Hippel, 2006). With extensive research being available on open-source software on these topics, a lack of such research can be perceived when it comes to open-source hardware.

Due to aforementioned reason, this thesis aims to fill one gap in the research on open-source hardware, which deals with the motivations and contributions of individuals to open-source hardware development. More specifically, this study aims to explore the motivations of open-source hardware developers and how these are different from those that have be found in open-source software projects. Evaluating motivation and contribution, we hope to further clarify the difference between software and hardware development, with its specific underlying characteristics. This thesis will help in understanding what motivates developers in the open-source world and will give valuable insights on how to foster participation and performance in open-source hardware projects which can serve to solve business problems. Moreover, we will highlight the relevance of open-source hardware developers' motivations with regard to user entrepreneurship. To gain such an understanding, empirical data will be collected through an online-based survey, supplemented by archival user data. Additionally, an in-depth interview will be analysed, with a person working in the field of open-source hardware. For the present study, we have used exploratory research from literature and personal interviews to elicit common motivation patterns among developers. These patterns have been formalized and operationalized in a survey, which was distributed to the Arduino community.
In the following section, an overview of existing literature on open-source software and hardware will be provided, serving to clarify the concepts of open-source software and hardware. Following this, literature on open innovation and user entrepreneurship and applicable theories regarding motivation will be reviewed before discussing the applicability of these theories to open-source hardware. Then the method, relevant data and results of the survey on motivation and contribution to open-source hardware will be considered and discussed with regard to its managerial implications. As a conclusion, we will review the study and elaborate on its contributions and limitations.

# 2. Literature Review

## 2.1. The Open-Source Phenomenon

In order to clarify the subject matter, the open-source phenomenon and its distinctive properties should be highlighted. Therefore a historical overview on the open-source

phenomenon will be given, followed by an explanation of its distinctive characteristics. Even though this thesis predominantly deals with the phenomenon of open-source hardware, it is of use to consider the evolution of open-source software, as it is historically and conceptually closely linked to the recent development of open-source hardware. Later, however, it will be shown that not only in terms of practical considerations, open-source hardware differs substantially from open-source software. The following historical overview relies mostly on the works of Lerner and Tirole (2002) as well as Von Hippel and Von Krogh (2003), who provide great insights into the evolution of this topic.

In particular software development which involves sharing and cooperation has a long history (Lerner & Tirole, 2002). Even before the term open-source software was coined, software developers, scientists and engineers shared their source code freely, mostly in academic settings and corporate laboratories. This sharing behaviour was inherent to their community and was often considered to be vital to their culture, which is often referred to as “hacker culture” (Von Hippel & Von Krogh, 2003). Contrary to popular opinion, the term “hacker” has a predominantly positive connotation in open-source communities, and can be seen as an honorary title for being especially talented or dedicated (Von Hippel & Von Krogh, 2003). A variety of important source-codes were shared, and this sharing was accelerated with the advent of the Usenet, an early network-based discussion board standard, connecting software developers at various locations. This was also the case with the source-code for the UNIX operating system and C programming language, which were originally developed at corporate AT&T Bell Laboratories. Sharing increased as the network size of the Usenet increased steadily, making the source accessible code to a wider audience over time.

As a result of this increased sharing, before-mentioned sharing culture received a critical shock when source code originally created at MIT’s Artificial Intelligence Lab, a major hub for software developers at that time, was licensed to a commercial firm. Opposing the ideology of sharing code freely, this firm subsequently decided to restrict access to the source code. Moreover, telecommunications company AT&T Bell started pursuing its intellectual property interests, threatening litigation to enforce its rights. This development was in stark contrast to the original sharing culture,

which had no provisions for intellectual property and consisted mostly of informal sharing networks (Steven, 1984).

Due to the conflicting interests between freely sharing source-code and intellectual property rights, MIT software developers and other members of the academic community were rendered unable to further develop the code or use it for learning purposes.

A change in the informal nature of source-code sharing was called for, as a reaction to these negative developments for the software developer community. With the establishment of the Free Software Foundation in 1983 by Richard Stallman, member of the MIT Artificial Intelligence Laboratory, a new era in cooperative software development saw the light of day (Lerner & Tirole, 2002). This movement lay the foundation of what we nowadays know as open-source software, due to formal licensing agreements under which software developed collaboratively could be licensed, preserving the ability to share software freely. Software developers deciding to keep their software free, could license it under the newly created standardized General Public License (GPL), which can be found in Appendix 1. This license makes it obligatory to make the source-code freely available and bars subsequent users, who might modify or use the code for their own works, from privatizing the code. Thus, software including or built on GPL licensed code has to be made available under the same conditions as the originally licensed code, which is clearly stated in the licensing agreement. Resulting from this, the free software movement was born, setting it apart from freeware or shareware, which can be obtained at zero or nominal cost, with the source-code remaining undisclosed. One should note that ‘free’ in this context is not meant solely in monetary terms but more conceptually as in ‘freely available’ and ‘free to use as desired’. Effectively this means that free software can be modified by virtually everyone, even though in practice a core community remains committed to the development of software, as will be discussed later on.

Free software became to known as open-source software only in the late 1990s, when Bruce Perens and Eric Raymond coined the term, on the base of practical considerations rather than moral rightness motivations (Kogut & Metiu, 2001). The open-source software movement was born, even though most of the licensing agreements were only marginally different from what is known as free software.

Whereas afore mentioned movement might imply a segregation between members supporting the Free Software Foundation and profit-seeking corporations, the opposite is true. Starting in the early 1990s until now, we can see a more interwoven structure with profit-seeking firms investing in open-source software projects, which creates a puzzle as the firms are giving up part of the ownership of a valuable asset. Moreover, a variety of new licenses came into being, especially with the advent of Linux, an operating system based on UNIX, created by Linus Torvalds and its distributions, among one is the Debian distribution, which includes the GNU Public License (GPL). Organizations gave greater flexibility in licensing the software and gave options to combine open-source code with privately owned code. As a result of this new approach, the open-source definition was coined, which to this day remains the predominant definition of what marks open-source software. Some of the main considerations are, that software developers can decide at their own discretion, which parts of their software will be licensed freely and which parts will be kept privately. Opposed to the GPL license per se, which required the complete software to be packaged under the same license, developers can decide how to accommodate a variety of licenses, including the GPL license.
Summarizing above mentioned history, and acknowledging that open-source software communities are nowadays very successful, we can explain the success due to a confluence of three factors, as explained by West and Dedrick (2001): the liberation from licensing restrictions imposed by commercial firms, an ideology opposing software ownership and the advent of the Internet as a platform for efficient code sharing. Having briefly explained the history of open-source software, this paper will now proceed with an explanation of how open-source communities and software development usually work, before continuing with a review of the economic implications of open-source software.

### 2.1.1. Characteristics of Open-source Communities

Having reviewed the history of collaborative and eventually open-source software development, it is of interest to take a closer look into how these communities work. From the previous section becomes evident that open-source software is more than just a collaborative effort to create software for a specific use. Instead, open-source software development is characterized by closely-knit community where members contribute to a common goal, which is creating software either for personal use or for

use in the professional or academic environment that the developers find themselves in (Constant, Sproull, & Kiesler, 1996). Taking this notion further, we can see that open-source software development communities are usually internet-based communities with a self-governed organizational structure.

Understanding the government and organization of open-source communities is an important issue, as developers in these communities cannot rely on existing structures that are given by a firm, which manages the development effort. Therefore, we can see that in most cases communities are self-governed. Often times this kind of organizational structure require new contributors to pass an informal acceptance test, which requires the contribution of high-quality code and elaborate technical support to the community (Von Hippel & Von Krogh, 2003). It is interesting to note that this kind of tests closely resembles what would be a job interview in the corporate world. Even with the absence of trust and a corporate sponsor who might take over the role of governor, effective performance of individual agents is safeguarded by mechanics which “ensure the control, efficiency, predictability and calculability of processes and outcomes in virtual organizations” (Gallivan, 2001).

Once participating in the project, be it with or without an acceptance test, it has been shown that developers take on different roles. Research has classified these roles differently according to certain criteria, which might be overlapping and should not be considered as exhaustive. Franck and Jungwirth (2003) classified developers in two categories, namely rent-seekers, who are looking for a return on investment, and donators, who are not seeking for any reward. A different classification by Raymond (2001) subgroups contributors into those who own the project, and have mostly been concerned with writing the original code, as well as maintainers, who are taking care of bug fixing activities and the active maintenance of an existing product.

While previously mentioned classifications rely on a distinctive characteristic of developers, other studies have focused more on the degree of participation of individual contributors. Koch and Schneider (2002) find that overall there are more participants in open-source software communities than in traditional organizational forms, however only a small part of the community, a so-called “inner circle”, is responsible for the major part of the output. One explanation can be that most online communities consist to a large part of “lurkers”, who do not actively contribute to the project for a variety of reasons (Nonnecke & Preece, 2000). Moreover, Shah (2006)

finds that initially contribution initially is high for users, who require satisfying a specific need, but diminished when they do not enjoy the programming effort. Critical to open-source projects are the hobby developers, who stay committed to the project even after satisfying their needs, which is in stark contrast to gated communities (Shah, 2006).

Apart from the governance structures, that have been mentioned, one should also take into consideration the principles, which are at the foundation of open-source communities when it comes to the actual development process. One particular principle in this case is what has been described with a metaphor of the cathedral and the bazaar, which was used by Raymond (2001). Opposing traditional software development which resembles building a cathedral, Linux was developed with a bazaar approach: Instead of working in a hierarchical order behind corporate gates with dedicated developers and testers, users take on a variety of roles, developing and testing the software themselves frequently, resembling the sort of business that can be found on a bazaar. Closely linked to this principle is a policy which has been widely adopted by open-source developers which is known as “release early, release often”, which is believed to be advantageous to effective in software development, finding and fixing bugs than the policy of only releasing bug-free software which has been tested by dedicated individuals (Raymond, 2001). Using the internet as a collaboration platform, this principle is easily enforceable and increases development speed as peer control ensures that bugs will be found and fixed, which is one of the success factors of open-source software development. Moreover, Franck and Jungwirth (2003) also highlight the advantage of the principle compared to proprietary software, where users are only willing to pay for a updated version, when there is a sufficient threshold of improvements offered, whereas users of open-source software have the freedom to immediately update their software. Thus, granting the ability to monitor each update and not only major updates increases the debugging capacity due to a wider reach. In the case of proprietary software, bugs may be overlooked by developers and pile up which might lead to unwanted results when experienced by the final user and often timer customer of the software.

Effective communication is also at the heart of a successful open-source project. In order to support users, contribute and discuss new source, mailing lists and platforms are set up by those who start the project (Von Hippel & Von Krogh, 2003). On these

platforms, new and modified code will be discussed and in case of a positive contribution to the overall project in terms of use value and quality, it will be published in a downloadable version by the maintainers of the project. Certain authorized developers who usually play a major role in the development of the specific project undertake this publication of a version with an according version number (Von Krogh, Spaeth, & Lakhani, 2003).

Speaking of version numbers and published source codes, the specifics of software should be considered with what regards source code and binary code. Most of the platforms on which code is exchanged for collaborative software projects, offer versions of the source code for download purposes. For the average computer user, these source codes are of little value, as the source code does not represent a piece of final software. A source code per se does not constitute an executable program until it goes through a so-called compilation process: The compilation process translates source code into binary code which in turn is directly able to trigger actions in the computer. Due to the fact, that software developers are able to modify the code and eventually compile the code to make it executable, we can thus conclude that the platform are more targeted towards knowledgeable users who possess the knowledge to compile source code into binary code. Frequently, however, pre-compiled software versions for various platforms (such as Windows, Mac, UNIX etc.) can be found on the collaboration platforms as well, next to the open-source-code, making the software itself available freely to virtually everyone with an Internet connection. Commercial or paid software, however, can usually only be obtained as a final compiled piece of software, from which the source code cannot be obtained. Even through a reverse engineering process, translating binary code into source code, it is virtually impossible to obtain the source code. Compared to open-source software, this prohibits any unauthorized user from making modifications to the source code (Franck & Jungwirth, 2003). In particular the availability of open-source software, which is often considered free software, through various platforms and means, deserves closer inspection with respect to its economic considerations.

### 2.1.2. Economics of Open-source

It has previously been shown that open-source software is freely available and distributed at a zero or nominal cost on the Internet. Basic economic theory tells us that a good, which can be consumed by an individual without reducing the availability

to other individuals, constitutes a public good if it is available at zero cost for the consumer (Mishkin, 2008). By making software available freely on the Internet and allowing reproduction of the software at zero cost, not reducing the availability by consumption, open-source software shares many characteristics with a public good. While this might seem positive at first glance, it also brings with it some of the negative properties such as the “free-rider” problem.

Users who do not contribute to the development of the software, can still obtain the software freely, which means that there is virtually no remuneration for the developers of the software. In fact, software developers incur significant opportunity costs working on open-source software as they cannot work on other projects at the same time, which would be paid. Coupled with the absence of remuneration by people who use the software, basic economic theory would predict a market failure, which would lead to the cessation of the development project as private costs outweigh private benefits by participating in the project for individual developers. Economic theory thus does not provide for open-source software. Puzzling enough, however, open-source software communities are growing steadily and even private firms are sponsoring and endorsing open-source communities, which make their source code available freely. To find out why this is the case, theory on individual motivation will be presented later on in this study, which will be operationalized in the empirical research.

## 2.2. Open-source Hardware

Having focused primarily on open-source software, it is of interest to take a closer look at the relatively recent phenomenon of open-source hardware and its particular properties. Lock (2013, p. 13) describes open-source hardware very well, as that it “focuses on hardware — physical artifacts — rather than immaterial source code.” First of all it should be noted that the open-source hardware community mostly acts according to the same principles and beliefs as the open-source software community, using similar licensing models to ensure the freedom of the end product. As a result of this, open-source hardware can be similarly opposed to proprietary hardware as open-source software is opposed to proprietary software. For most open-source hardware initiatives users have the freedom to “run, copy, distribute, study, change and improve” designs and source-code, which is very similar to the license that it is being advocated by the Free Software Foundation (“gnu.org,” n.d.). As an addition to

programming code, open-source hardware has much more related concepts that should be shared freely, due to its physical nature. Often times open-source hardware projects are being disseminated with dedicated mechanical drawings, schematics, circuit boards etc. (“Open-source hardware,” 2014).

Whereas there is no univocal definition of what open-source hardware is, new licensing models have been proposed to capture the specific nature of open-source hardware, of which one is the TAPR open hardware license, which was created by an amateur organization. The license, which can be found in Appendix 2, delineates the requirements that have to be met for hardware to qualify as open-source. The requirements include that the interface to the hardware must be made publicly available, the design must be made publicly available and the tools to create and design the hardware should be free (Rubow, 2008). Even though this license seems to fit the purpose well, it should be noted that this license is not exhaustive and other licenses could be created and used to create open-source hardware. The license mentioned is merely an example of what a license could look like, as there is no consensus as to what exactly defines open-source hardware.

Given that the theoretical foundations for open-source hardware already existed for a relatively long time due to its close relation with open-source software, it should be noted that open-source hardware only recently has become a widespread concept. Therefore, it is interesting to have a closer look at the developments that have led to the widespread use and dissemination of open-source hardware projects. Probably the most important catalyst to accelerating the dispersion of open-source hardware lies in the cost consideration of electronics. As production technologies have become more efficient over the decades, and the development of new technologies has increased, electronics are becoming ubiquitous and more widely available than ever before. A good example of this are digital watches, which when initially launched cost an equivalent of thousands of euros whereas they are now available as free gifts or at prices below five euro. Moreover, changes in the methodology of how tools are designed and due to the standardization of interfaces, which will be discussed at a later stage, highlighting the Arduino project, open-source hardware has become increasingly more popular (Acosta, 2009).

Attention should be paid however to one crucial difference: even though electronics and consequently open-source hardware has become more affordable over time, it

still comes with some upfront investment that need to be taken by the user to create physical objects. Therefore, we can conclude that there is a clear distinction not only in the nature, but also in the cost associated with developing hardware. Whereas one would only need a suitable computer to program software, additional parts are needed to create and test hardware. However, these costs are reduced by dedicated platforms, which will be presented in the following paragraph.

### 2.2.1. Open-source Hardware Platforms

To facilitate the collaborative efforts that are undertaken in open-source hardware communities, several platforms have emerged that create a common space for people to work on. Even though designs to build open-source hardware are freely available on the Internet, and individuals could build shared platform from scratch, this requires a significant initial effort: Parts have to be bought and assembled, and for some people difficult soldering work is required to create working platform. To remove these barriers of entry to open-source hardware communities, this initial effort has been reduced with the advent of dedicated platforms. These dedicated platforms remove the initial barriers in two ways: First, due to economies of scale common platforms are cheaper, as individuals need not buy electronic components at a relatively high price from specialty shops. These parts are bought in bulk by the entity that creates the platform, thus significantly reducing costs. Second, the assembly of these dedicated platforms is performed by machines, which create large amounts of the boards in an automated process. It should be noted, however, that in order to guarantee feasibility, open-source hardware should fulfil the following requirements, as indicated by Davidson (2004, p.456): “First, there must be a big audience for OSH. Open-source requires a pool of developers, and few will want to work on an obscure project. Next, the design must have an easily exchangeable form. It must also have clear and easy-to-understand specs, so potential customers can tell whether or not it’s useful for them. And finally, there must be a simple way of verifying the design.”

To get a better understanding, it is useful to go through each of these requirements in turn. Similar to open-source software development, open-source hardware requires a great amount of developers that work together collaboratively in order to exchange their work. To make this possible, a shared platform has to be given, which gives legitimacy to specific open-source hardware platforms. As noted, the most important

part of the shared platform that allows exchange is a design that can be exchanged and is understandable by all parties involved. Creating such a platform and selling a dedicated platform per se fulfills this requirement, and it becomes evident from user groups and other websites whether the platform is of use or not to the potential future user. Small batch production of these platforms, as has been described above, also allows verifying the design and the correct functioning of the platform immediately. Having mentioned the Arduino platform before, we can conclude that this platform fulfills all the requirements mentioned above. For a better understanding of what the Arduino platform actually is, we will now highlight the specifics of this platform, and what makes it so successful in the open-source community.

### 2.2.2. Arduino Platform

The open-source hardware platform is best described in its own words, which can be found on the Arduino website: “Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments” (“Arduino - HomePage,” n.d.).

While this definition already explains to a great deal what Arduino is, the platform specifics should be further clarified to obtain a better understanding. To do so we will clarify above-mentioned definition in parts. From this thesis it should be evident what open-source is, so the definition and explanation of open-source need not be further clarified. The term “prototyping platform” means that the Arduino platform can be used to create prototypes of physical, interactive objects. More precisely, the Arduino is a microcontroller. Microcontrollers are made up of a microchip on a circuit board with read-write capabilities, memory, inputs and outputs, and are mostly used for low-power and low-memory purposes (Gibb, 2010). This means that the user can write a program that for instance processes inputs and generates certain outputs. To remove abstraction, an example would be as follows: The Arduino receives input from a temperature sensor that is connected to the microcontroller and measures temperature. It is programmed in such a way that it will light up a green light emitting diode (LED) when the temperature is below 30 degrees Celsius, or a red LED when the temperature exceeds 30 degrees Celsius. Noting that this is a relatively simple example, the possibilities are sheer endless and there are ample sensors, which can be used as inputs. Examples of sensors include ranging sensors that can measure

distance, light sensors that can measure the level of illumination or even sensors that react to humidity. As such it makes it a powerful tool to develop and innovate, as it is very accessible and easy-to-use.

The intended target group is very wide, as the explanation from the website shows. In fact, we could observe projects that were based on Arduino in a wide variety of fields. Some people used Arduino to create innovative musical instruments, whereas other inventors created robots that indicated whenever they received an e-mail or text messages on the Arduino platform, again other users built a cat-feeding station which provided the right food to the right cat, given the specific dietary restrictions of cats in a household with multiple cats. The amount of projects that can be found on the Internet is innumerable, and it is beyond the scope of this thesis to highlight these projects in more detail.

What is especially interesting is that most of the projects are published on the Internet including circuit design and software code. Users can take existing code and use it for their own inventions, which makes it very easy to start – even though this results in initial projects being merely some form of *bricolage*. The programming language used to control the Arduino directly is the processing language, which was intended primarily for designers or artists (Noble, 2009). As it is not catered to a technological crowd, it makes the platform even more accessible which is considered to be beneficial for fostering open innovation.

In fact the model of innovation that the Arduino platform enables, is similar to the way innovation is done at innovation labs like IDEO, where people from a variety of backgrounds sit together to invent new products. Making the most out of the skillset of every participant can help to create beneficial products. Gibb (2010) describes this cross-disciplinary approach very well: “The skill sets of the teams expand by collaborating with people of diverse backgrounds on the common platform, which the Arduino microcontroller provides.” Moving away from corporate R&D centers, the platform is especially useful for open innovation, as that it allows people all over the world to connect and exchange ideas to create new products.

The Arduino open-source hardware platform became and still is a huge global success, with more than 50,000 units sold only two years after mass production had commenced, which is interesting taking into account that initially the market was optimistically estimated at 200 units (Thompson, 2008). The idea for Arduino came

when Massimo Banzi, teacher at a technology design school in Italy, needed to have a platform which his students could use to control their robotic installations (Thompson, 2008). It is fascinating to note, that the story of how Arduino came into being is related to many other open-source software and hardware projects: Often times, developers experience a need to solve a problem, for which there is yet no solution available. Using the community to initiate a project, they find other people who experience similar problems and create a solution to solve this problem. In this case, the problem was the lack of an affordable and easy-to-use microcontroller, which was solved by creating an open-source hardware microcontroller.
Due to the flexibility of the platform to address different problems, a variety of Arduino platforms came into being, each addressing specific needs of developers. In addition to a wide range of basic platforms for different needs, the platform can also be customized by using so-called shields. Shields are printed circuit boards that can be attached to the platform to extend its functionality, and are also open-source. They facilitate the development process by extending the capability of the platform without the need of the developer having to build the functionality him- or herself, which would add an additional significant amount of resources invested to ensure functionality. All in all, this makes Arduino an attractive platform for developers, as the platform is immediately ready for use. Due to the fact that this makes the platform also a widely used prototyping platform, it is of further interest to note why people actually participate in the platform from an open innovation and user entrepreneurship point of view.

## 2.3. Innovation Literature

Following the discussion on the specifics of open-source software and hardware, it is of importance to consider the position of this seemingly very technological field in business and management literature. To do so, the literature on *open innovation* and *user entrepreneurship* will be reviewed, which gives explanation for the link between open-source projects and innovation literature.

### 2.3.1. Open Innovation

The term open innovation has been coined by Chesbrough (2003) and describes a scenario in which firms open their corporate boundaries to be receptive for external innovation initiatives. Nowadays, more and more firms are using open innovation to

create new offerings, since the brightest minds for innovative purposes can often be found outside of the corporate boundaries. In order to reap the benefits open innovation initiatives, firms can resort to the possibilities of the Internet to do so. Research has shown how the Internet can be used in the differing stages of new product development by taking on an open innovation perspective. According to Prandelli, Verona and Raccagni (2006), the Internet offers unique possibilities to integrate users in the development process at different stages, ranging from idea generation to the product launch. Various tools are at the hand of the company to include the user in the innovation process in each respective product development step.

Often times these tools are in the form of Internet-based collaboration mechanisms. Mapping these mechanisms according to the nature of the collaboration and the stages of the new product development, open-source mechanisms can be classified as having a high richness in collaboration and can be found at the back-end stages of the new product development process, especially in product design and testing (Sawhney, Verona, & Prandelli, 2005).

From the literature we learn that it can be profitable to include users in the innovation process of the firm, as it helps to create value by working with the customer (Prahalad & Ramaswamy, 2004). Therefore, it is interesting to note that open-source hardware communities might be leveraged to do so, which already happens in the context of open-source software.

### 2.3.2. User Entrepreneurship

As opposed to classical entrepreneurship, which is aimed at creating an offering to create profits, user entrepreneurship often happens accidentally (Shah & Tripsas, 2007). Having discussed the workings of open-source communities, parallels can be drawn to the literature on user entrepreneurship, which becomes especially evident if we consider the stages of user entrepreneurship as described in the model by Shah and Tripsas (2007).

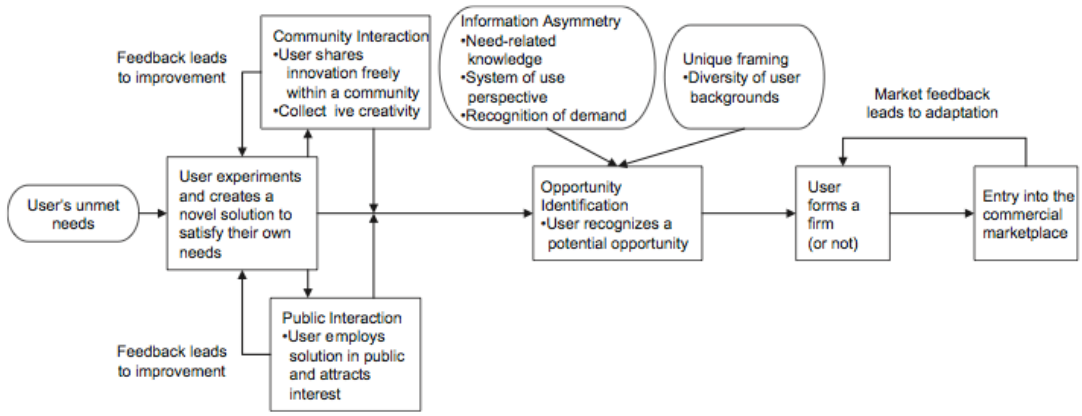


Figure 1. User-entrepreneurship model. From: Shah and Tripsas (2007).

Comparing this model to the stages previously described in the workings of open-source communities, we can see an abundance of parallels: an unmet needs is the starting point to engage in open-source projects, experimentation and interaction with the community leads to improved solutions for the initial problem. It is interesting to note that the term *community* in this context is described as a social entity, due to the presence of social structures (Ashforth & Mael, 1989). We have seen previously seen, that these social structures are indeed very present, and the exchange in the community does take place on a social level.

Another interesting trait of user entrepreneurship is the dimension of creativity, which will also be discussed in the theoretical overview on motivation below. The community that revolves around the problem, initially posed by the user entrepreneur, has all traits that are conducive to a high level of creativity: the need for support (help-seeking), giving help, a variety of background looking differently at the problem (reflective reframing) and reinforcement of the three previously mentioned dimensions due to the social nature of the community (Hargadon & Bechky, 2006).

An important aspect in the model also shows us that community and public interaction is very important, as it helps to create a solution for problems. Only at a later stage, after the product development process has been finalized, the user might use his newly obtained solution for entrepreneurial purposes, so as to generate profits. Knowing this potential, we can consider open-source hardware platforms as very beneficial for user entrepreneurship.

It should be noted that user entrepreneurship falls under the category of user innovation, and should be seen distinctively different from open innovation: User innovation originates from the unmet needs of an individual user, rather than from a

company that offers specific benefits for fostering innovation by its user base, which is the case in open innovation (Bogers & West, 2012).
We should note that user entrepreneurship is very applicable to the Arduino community, as users often times start developing to solve a personal problem. However, if the solution can be extrapolated to solve the problems of a multitude of users, the possibility for an entrepreneurial endeavour exists. Therefore, the relevance between research on user entrepreneurship and open-source hardware initiative is highly relevant. In order to elicit on how to motivate users to participate in open innovation or how to elicit user entrepreneurship, we will now consider motivational categories that have been used in the empirical part of this study.

## 2.4. Theories on Motivation

### 2.4.1. Overview

Knowing that open-source hardware developers seemingly work for free and share a good publicly at zero cost opens up the question what motivates these developers. Literature offers several explanations for why open-source developers do what they do, ranging from personal needs to looking for belongingness to a social group (Raymond, 2001). The literature on motivation is abound, and a variety of definitions for motivation exist. However, the most commonly mentioned definition of motivation is that somebody who is motivated undertakes action towards an end (Ryan & Deci, 2000). On the opposite, somebody who is not motivated does not undertake any action towards an end can be considered unmotivated.
Keeping this definition for motivation in mind, it is also noteworthy that there is a vast amount of literature dealing with the classifications and distinction of types of motivation. These classifications and distinctions are an attempt to create a model of motivation. While it is beyond the scope of this paper to explain all models, one should note that there is no theory that explains motivation perfectly, and most papers highlight certain aspects of perspectives of motivation, ranging from incentive theories to cognitive-dissonance theories and drive-reduction theories, among others. Due to their widespread use in academia and the business context, some theories should be mentioned nevertheless.
Most prominent is Maslow's Hierarchy of Needs, which was proposed as early as 1943. This theory classifies the needs in a pyramid, where the most basic needs

such as physiological needs are at the bottom of the pyramid and only as the most basic needs are satisfied more specific needs can be addressed, with the top-level need being that of self-actualisation (Maslow, 1943). Another theory, proposed by Herzberg classifies motivation according to two factors: motivation and hygiene, which refer to factors that drive positive satisfaction and need to be fulfilled to avoid demotivation, respectively (Herzberg, Mausner, & Snyderman, 2011).

Abstracting from the latter theory, we can explain motivation also in terms of intrinsic and extrinsic motivation, which is a widespread classification of motivation. These two types of motivation are often used to describe motivation. Intrinsic motivation in this case means that an individual is motivated by doing a task in itself, whereas extrinsic motivation refers to motivation that comes from an external source, e.g. by being compensated monetarily for performing a specific task (Ryan & Deci, 2000). Therefore, extrinsic motivation is oriented towards an outcome, which can be separated from the task itself. In the context of open-source hardware developers intrinsic motivation would be the enjoyment of developing hardware itself, whereas extrinsic motivation would be the prospect of earning money by developing open-source hardware.

Often times, however, open-source developers have found to be motivated by a third factor, which falls beyond the scope of the intrinsic-extrinsic dichotomy. While a developer might enjoy the task of developing open-source software or hardware itself, there might also be a motivating factor that the development process leads to personal improvement, which in turn might lead to a job offer in the future. As it is clear to see, the lines between extrinsic and intrinsic motivation is blurred in this case. Due to the interplay of both intrinsic and extrinsic motivations, we suggest adding a third category of “internalized extrinsic motivation” as proposed in psychology literature by Deci and Ryan (1987). We will now in turn elaborate on these three categories of intrinsic motivation, extrinsic motivation and internalized extrinsic motivation, which have been operationalized in the survey to elicit open-source hardware developers’ motivations.

### 2.4.2. Intrinsic Motivation

Deriving utility (joy, pleasure etc.) from performing a task itself is widely regarded as intrinsic motivation. The field of intrinsic motivation has not been extensively studied until the 1970s, however early research on this type of motivation can be found from

the late 1950s on. Even though it was not studied in human beings first, White (1959) noticed how animals and other organisms engaged in exploratory, playful and curiosity-driven behaviours, even though there was no reward or reinforcement. In fact, the saying “curiosity killed the cat” can be derived as a form of intrinsic motivation, as White (1959) noted. Applied to humans, the theory of classifying motivation into intrinsic and extrinsic motivation arose from a need to distinguish from previous theory. Operant theory stated that all behaviours are motivated by rewards, whereas learning theory stated that behaviours are driven by physiological needs (Ryan & Deci, 2000). Parting from this theory, the classification intro intrinsic and extrinsic motivation resulted in a psychological theory called Self-Determination theory, which determined the psychological needs that make up intrinsic motivation as being competence, relatedness and autonomy (Deci & Ryan, 2002). These categories have been used to measure intrinsic motivation by grouping them in two sub-categories, referring to enjoyment-based as well as community-based intrinsic motivations, which address the three needs discussed above directly. To clarify these two categories, they will be discussed individually.

#### *2.4.2.1. Enjoyment-based Motivation*

Performing a task itself and deriving utility from doing so is considered to be necessary for being intrinsically motivated. In the absence of rewards there should thus be other factors that drive human beings to do a certain task. As such, it becomes obvious that enjoyment is such a factor. In fact, we can see ourselves doing tasks that might not guarantee a rewarding outcome, yet we still perform these tasks. Often times, even though there is an outcome, which is positive, we merely do the task due to the nature of the task itself. A good example of this is long distance running, which may lead to health benefits but is often done solely because of the joy of running or to achieve a so-called runner’s high. Speaking of the runner’s high, which is a state of happiness which comes from running, we can take this as an example of what Csikszentmihalyi (2000) calls the state of flow, which is a state in which enjoyment and other relevant factors such as concentration and challenge concerning intrinsic motivation are maximized, and often occurs when the skillset of the person involved matches the challenge. Discussing this topic with developers in an informal setting, this state of flow was clearly observable: Often times, after writing a considerable amount of program code, programmers find themselves being highly

concentrated when the code is not working properly. To fix the code, the programmer engages in so-called debugging activity, which deals with finding errors in the code and resolving them. During this state, concentration is paramount and one developer described it “as solving a puzzle and finding the missing piece”. After finding the afore-mentioned missing piece, developers were observed to be especially euphoric and joyful, which is clear evidence that there is some sort of enjoyment derived from developing software and hardware. In fact finding a challenging task becomes easy as the nature of open-source software developments leads to an abundance of projects, which can be found, on the Internet, ranging from very simple to extremely complex.

#### *2.4.2.2. Community-based Motivation*

Noting that intrinsic motivation has been sub-grouped into three categories, which are competence, relatedness and autonomy, it can be concluded that the community aspect of open-source hardware can play an important role to motivate developers. As described in an earlier section, the community and the multi-perspective approach to developing hard- and software is vital to the success of projects. In fact, community based motivation originates from the increased perceived levels of competence and psychological relatedness. However, community based motivation can also be understood in terms of acting on principles (Lindenberg, 2001). From the historical overview discussed before, it became clear that the open-source software community was founded due to a need for free software, where free refers to a philosophical mind-set rather than meaning zero cost. Talking to open-source hardware developers, it became clear that the willingness to act according to principles is still a predominant factor in the open-source community. One developer, which was interviewed, explained that for him it a vital part of the community feeling is to share code freely and receive code in return, as part of a reciprocal process. Some theories go as far as to consider the open-source community a social movement, which have political and social goals in common (Hertel, Niedner, & Herrmann, 2003). As the open-source community is solving a problem together socially by communicating through forums or designated communication platforms, the social aspect becomes very clear. On top of that, goal-oriented behaviour in an organized structure, which is often created by the members of the group itself, shows many similarities to social movements. Noticing that community-based motivations

play an important role, it is of great importance to test these motivations and see their impact on developers' motivations.

### 2.4.3. Extrinsic Motivation

Extrinsic motivation is best described as the motivation that comes from an external influence, which can be in the form of a reward or punishment to make an individual perform a certain task. The most common example of extrinsic motivation is paying somebody to exert a task. At the same time this also presents a good example of a purely extrinsic motivation. Other forms of extrinsic motivation are less distinctive of intrinsic motivation, as there is a reward following the performing of a task, which is not as obvious or measurable as a direct monetary reward. These extrinsic motivations are described as *internalized extrinsic* motivation (Roberts, Hann, & Slaughter, 2006). To get a better picture, we will now in turn explain these motivations with some examples.

#### *2.4.3.1. Direct Rewards*

Following the approach taken by Hars & Ou (2001), the first category of extrinsic motivation describes purely extrinsic motivations. In order to measure these motivations questions as to whether the task performed is a paid task should be asked. While for most tasks the motivations can be ambiguous, consisting of a blend of intrinsic and extrinsic motivation, the performing of some tasks is clearly motivated extrinsically. Typical examples of solely extrinsically motivated sorts of tasks are tasks, which require little effort and are of a repetitive nature. Works at conveyor belts or in call-centres that are always executed according to a pre-defined script are good examples of this. Since these tasks are usually quantifiable and measurable, extrinsic rewards present a motivational driver that can even be tied to certain goals, such as processing a certain amount of parts or calls in a given amount of time. The monotony of the task itself only allows for extrinsic rewards and people are assumed to be more productive when reaching goals due to a reward.
In the field of software and hardware development we could transfer this situation to a goal of needing to write a certain amount of lines of code which is then tied to a specified financial compensation. A more usual extrinsic reward though might be the monthly salary that is paid to a developer who works in a corporate environment or who is self-employed.

#### 2.4.3.2. Personal Needs

Originally software was written to address personal or corporate needs. In fact, if we have a look at the first applications that have been written we can see that they were clearly need-based. Text processors were written to reduce the work needed to create and duplicate documents as well as to reduce printed information. Spreadsheet software was developed to solve business problems and facilitate complex calculations, which contributed to the success of companies who adopted these kinds of software. And even games can be seen as an example of fulfilling a need, namely the need for entertainment. As the amount of applications increased, ever more needs were addressed by software, up to the point were we now have millions of apps for the most diverse purposes.

Taking this into consideration, it becomes clear that personal needs play an important role as a motivational driver to develop software and hardware. Many of us had simple problems, which could effectively be solved by a piece of code. In the literature on motivations of open-source software development, this type of motivation has also been described as use value (Roberts et al., 2006). Writing a piece of software, or developing hardware creates value to the user as it helps to reduce time to perform a certain tasks, removes information cost as it might compile information needed. Eventually this means that the use value represents also a kind of monetary value, if we consider that time is money. Thus, it is fair to use this measure as an extrinsic reward, however the fulfilment of creating such software to solve a personal problem makes the line between extrinsic and intrinsic unclear. Personal-needs motivation is thus considered an internalized extrinsic motivation, as it is not completely distinguishable from intrinsic motivation.

#### 2.4.3.3. Future Returns

In the course of the research for this study many developers indicated that developing open-source hardware presented a great motivation to develop or improve a skill, namely the skill of software and hardware development. Some of the developers interviewed had no formal education in development, but still enjoyed developing. A particular example of this was a developer who graduated with a degree in Economics only to pursue a career as a developer in a corporate setting. A self-taught developer, he described that he had learned the skills out of pure interest in the subject matter. As with many tasks, developing software requires practice and

experience and accumulating more of these usually results in better development skills. Due to the property of receiving immediate feedback on the work presented, which is very present in open-source communities, self-improvement is facilitated heavily. As it raises the skill level, we can assume that there might be some future return from participating in open-source software and hardware development. Motivation to participate in communities and to virtually “work for free” might thus be derived from possible future returns. Studies on open-source software participation have shown that this is indeed the case, with developers answering positively to questions that measured whether developers engaged in open-source communities to raise their skill level in order to receive some kind of return in the future.
Given that these kinds of return most likely result in finding a better job, with more complex tasks or distinguishing the individual from others when it comes to finding a job or task, an increase in financial compensation is to be expected. As this presents an extrinsic reward it is right to include this category under extrinsic rewards. However, since the extrinsic reward may not be the only motivation, we have to consider this category as externalized intrinsic motivation. Improving their skillset, developers also increase their competence and possibly also their status, which might be associated to intrinsic motivations, which have been described above. Due to this, we sub-group future returns as internalized extrinsic motivations.

### 2.4.4. Additional Considerations

#### *2.4.4.1. Creativity*

Whereas the afore-mentioned categories all deal with the measurement of motivation resulting from existing theory, it was decided to add additional considerations due to the special properties of the topic at hand. In contrast to traditional software development, hardware development has the property that the end result is a physical object, machine or apparatus. As such, the results of the work and time invested in the project are not only visible on a screen but in real-life.
In addition, we have noted that the intended purpose for the Arduino platform originated in the creative space: artists who needed an easy-to-use and accessible platform for digital objects, which serve as art, used it. Taking these considerations into account, we can conclude that one specificity to the platform and the community might be creativity. Historically and ideologically, creativity and developing do not

present a clear linkage. Over time, however, artists have discovered the potential that software and hardware have as an outlet for creativity. In fact, creative coding communities have emerged, which use programming code such as C++, as a way of expressing themselves. A very recent example of such a community is the Cinder community, which describes its product as a “community-developed, free and open-source library for professional-quality creative coding in C++” (Cinder, n.d.). The results that can be obtained by using these libraries are aesthetically impressive and show very well the possibilities that software development offers even to artists. Whereas the example above again refers to software, we can observe how Arduino as a platform can be used to express oneself creatively. In fact, Arduino has successfully been used for creative expression, and it is relatively easy to identify projects, which are meant to be creative. In fact, there is a dedicated Arduino platform, which is designed for creating electronically enhanced fashion, also known as “wearables”, the LilyPad Arduino (Buechley & Eisenberg, 2008). The use of Arduino in fashion is just one of many examples of where the platform can be used for creative expression. Due to its flexibility and sheer endless possibilities, Arduino presents a valuable tool for artists and designers, given that it requires little formal programming knowledge.

Considering this, it was decided to cover this additional category in the conducted research. Going beyond the classic continuum of extrinsic and intrinsic motivation, it was considered to be useful to include a category named “Creativity” that measures the degree to which developers are motivated by creativity considerations. Due to the various ways of expressing oneself, be it through music, poetry, visual arts or other art forms, it was decided to include questions, which compare the end result of an Arduino project to some of these art forms.

#### *2.4.4.2. Research on Motivation*

Motivations of open-source software developers have been studied rather extensively, by using a variety of models to describe motivation. Apart from the dimensions described above, which have been adapted from Hars and Ou (2001) and Lakhani and Wolf (2003), the literature on motivations in open-source software also offers some different approaches. According to an overview by Von Krogh and Von Hippel (2006), the studies focus on individual incentives, how firms’ and

community participation in projects impacts on individual motives, and the relationship between incentives and technical design, among others.
It should be noted, however, that all of this research focuses solely open-source software, and does not include any open-source hardware projects. So far, no research has been performed on the motivations of individuals and groups that work in the open-source hardware space. Motivations can be measured in a variety of ways, and there is no definite way of measuring motivations. In line with literature overview presented previously, it was decided to evaluate the motivations of developers in the hardware space in a continuum ranging from intrinsic to extrinsic motivations from a more psychological point of view. The research design will be explained in the following section, followed by the results of the study, which fills the gap of understanding open-source hardware developers' motivations.

# 3. Methodology

## 3.1. Research Design

In order to get a better understanding about open-source hardware developers and their motivations, this study uses a research design that combines exploratory and empirical research. The motivations of open-source hardware developers will be evaluated empirically, analysing survey data and user data that has been obtained from a forum. In addition, an interview with a professional working with these developers will highlight the importance of understanding hardware developers' motivations to ensure positive outcomes in a corporate environment that uses open-source hardware for innovation purposes.
The empirical analysis consisted of a questionnaire that was sent out specifically to a singular open-source hardware community. It consists of a variety of questions, which are aimed at measuring open-source hardware developers' motivations, and are based upon the dimensions described in the theoretical framework under "Theories on Motivation" above. In addition more general questions are asked in order to obtain a better understanding of what a typical open-source hardware developer in this specific community looks like.
For the exploratory part of the research, an interview with the Global Director of Creative Technology of media agency Maxus, Nico Abbruzzese, was conducted.

Working with open-source hardware developers on a daily base to come up with creative solutions to business problems by means of rapid prototyping, Nico was able to shed more light into how open-source hardware developers work in a corporate environment. This research was conducted in the form of a semi-structured interview that allowed for a broad perspective and a variety of insights.

### 3.2. Data Collection

The empirical part of the study consists of data obtained from the Arduino user community through questionnaires and the forum statistics itself, which can be found at forum.arduino.cc. The Arduino user forum is the official forum and was created by the founders of Arduino to enable communication among users. The forum appears to be the largest forum focused on building hardware projects on the Arduino platform, and as of writing counts a user base of 205,205 users, which in total accumulate 1,707,774 forum posts and 211,579 topics. On average this means that every user accounts for roughly 8.3 posts and 1 topic started each. Another indicator for the popularity of the platform is the amount of average registrations per day, which as of writing accounts to 163.21 new users per day. Another interesting statistic is the male to female ratio, which indicates that for every woman registered, 16.5 male users are registered, clearly indicating a masculine trend.

The study questionnaire was distributed by means of a forum post, providing a link to the questionnaire. In addition, randomly distributed private messages were sent to users of the community, requesting their contribution to the research study. Due to a limitation in the allowed number of private messages that could be sent, these private messages were sent at various times of the day throughout a period of 40 days, after which the survey was closed. Even though the survey was publicly available to an audience of virtually more than 200,000 users, the response rate was relatively low and was driven mostly by respondents who were receptive to the private messages. This is consistent with findings that electronically, web-based questionnaires generally yield a low response rate (Couper, 2000). In total n=124 completed responses were collected, of which one had to be deleted due to offensive remarks directed at the author of the study in open-ended questions, effectively setting the number of completed responses to 123. The mean age of the respondents was 35.54 (SD=14.58), which appears to be higher than the mean age found in comparable studies by Hertel et al. (2003), Lakhani and Wolf (2003) and Oreg and Nov (2008),

where the mean age 30.0, 29.8 and 31.6, respectively. Taking a look at the ratio of male to female participants in the study, we can confirm the fact that the field currently is still dominated by male users, in line with previous research in the field of open-source software: Compared to the previously mentioned three studies, which show a proportion of male respondents accounting to 91.5%, 97.5% and 97.8%, the study at hand has 96.4% male respondents of all respondents that indicated their gender (n=121).

The survey questionnaire, which can be found in Appendix 3, consisted of general questions to elicit user profiles, as well as questions directed at measuring the motivations. Respondents had to indicate to what extent they agree to certain statements on a seven-item Likert scale. Examples of the motivational categories that have been operationalized in these statements are as follows: “Developing hardware is fun.” in the *enjoyment-based* category, “I am paid to work for the project.” in the *direct rewards* category and “The hardware is critical for my business or my work.” in the *personal needs* category. Respondents could indicate the level to which they agreed to these statements, ranging from “1 – Strongly Disagree” to “7 – Strongly Agree” with a neutral value of “4 – Neither Agree nor Disagree”. The questions were sub-grouped in five categories measuring motivation in terms of the motivational groups described in the theoretical framework. Additionally, creativity considerations were also measured on Likert scale items, as has been described in the theoretical overview previously.

In addition to the survey items that elicit responses from the participants in the study, users could voluntarily mention their username on the Arduino forum, which allowed for further data to be collected. The overall response was positive, with 80 respondents indicating their username voluntarily. User data could be matched to the profiles, which serve as a base for the additional analysis. The additional data obtained was used to gain further insights into the activity level of the users of the sample and included the following variables: *posts_number* indicating the amount of post for the individual user; *registration_date* indicating the registration date, which was converted into *days_registered* for statistical analysis purposes; *karma* which are points that can be given to a user for a good contribution by other users; *spent_online* indicating the total amount of time spent on the forum in days; and

*topics_started* indicating the amount of topics started by the respective user in the Arduino.cc user forum.

The data was analyzed in three ways: First, a descriptive statistical analysis to obtain a better understanding of the motivations in general was undertaken. Second, we ran an exploratory factor analysis (EFA) to identify whether there are latent constructs in the dataset, especially with regard to intrinsic and extrinsic motivations. Third, we tested whether the supplementary quantitative data obtained from the forum has any relation with the motivations tested of the users in the Arduino forum. The data was analyzed with an open-source statistics package called PSPP, which is similar to the proprietary statistics software package SPSS, which is developed and distributed by IBM. Performing crosschecks between PSPP and SPSS, identical results could be obtained, which gives validity to the data obtained from the open-source software package. Additionally, the factor analysis was undertaken by using SAS analytical software, to get a clearer understanding of the data.

# 4. Analysis and Results

## 4.1. Survey Findings

### 4.1.1. Descriptive Analysis

In order to get a preliminary understanding of the motivations of open-source hardware developers in the Arduino community, the questionnaire will be analysed by means of a descriptive analysis. The descriptive statistics help to get a better understanding of the characteristics of open-source hardware developers as well as to get a preliminary evaluation with regard to the motivations of the open-source hardware developers. Before going into more detail on the motivations of open-source hardware developers, we will first extend the analysis of overall characteristics of open-source hardware developers. Having already discussed some basic demographics in the data collection part, we will continue our analysis in more depth.

#### *4.1.1.1. Sample Characteristics*

Taking a closer look at the characteristics of the sample, we can paint a fairly clear picture of what characterizes the Arduino forum user. To do so, we first have a look

at the time that users spend working on their Arduino projects and how often they work on their projects, to gauge the activity level and to a certain extent the level of involvement. Overall we evaluate the level activity as being very high: 73.2% of the respondents indicated to work on their Arduino project at least 2-3 times a week, of which 30.1% even work on their projects daily. Taking into consideration that only 8.1% of the respondents get paid to do so, this indicates that they spent their free time throughout the week working on Arduino projects. This result is even more striking if we take a look at the time spent on projects: The largest part of the respondents spends between 6 to 12 hours a week on the project, which coincides with almost one full working day per week, 19.5% spend even more time, between 13 and 20 hours a week, on the project. The findings are confirmed by taking into account the supplementary data, which was collected from user data. The ratio of time spent online in days to days registered on the forum is on average 2.07%, which translates to users spending almost half an hour a day (3.5 hours/week) on the forum alone, which includes all users, even those who are completely inactive. The other part of the hours can possibly be accounted for due to the time spent working on coding and creating the project itself.

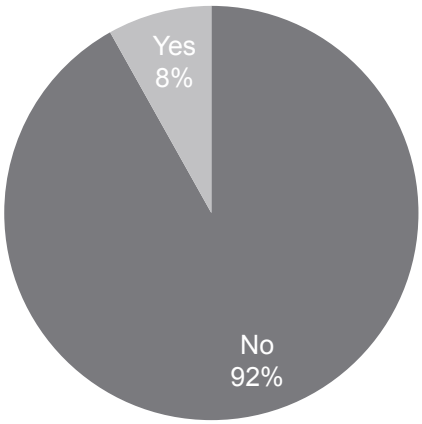


**Figure 2. Do you receive direct compensation (e.g. salary, contract) for your participation in the project?**

It is interesting to note that open-source hardware developers in the Arduino community indeed occur some sort of cost. When asked what their most important cost was for participating in the project, ranking them from least important to most important, most developers indicated that hardware parts were the most significant cost, with 86 respondents claiming so. What is especially striking, is that open-source

hardware developers also give up on sleep, social relationships and social time, with 61, 61 and 53 respondents respectively indicating so, having the choice to give multiple answers. Again, we can also see that money plays only a minor role, with only 39 of the 123 respondents indicating that time that could otherwise be used make money represents a cost to them. Overall, this represents the lowest number of responses to this question, indicating that money is not a primary driver for participation in open-source hardware projects in the Arduino community.

Another interesting consideration is the experience level of the users surveyed, when it comes to hardware and software development. Looking at the numbers we can see that the average software development experience in years is higher than the average hardware development experience, which gives some evidence to the fact that open-source hardware is a more recent phenomenon. Respondents indicated to have on average three years more software development experience than hardware experience, with the averages being 9.28 years (SD=9.78) and 6.13 years (SD=9.23). At this point it should be noted, that some users indicated to have even more than 30 years experience in both fields, however the scale on the questionnaire only allowed for values not exceeding 30. Including even higher numbers as a response for development experience, we assume the averages to be higher as well. Nevertheless, the averages weakly indicate a very experienced user base, which should be taken with caution due to a relatively high standard deviation.

From the data on occupation, we can see that many of the participants have a job which somewhat related to electronics, engineering or development. Even though only 21.1% of the respondents indicated to be a programmer, system administrator or IT manager, this number increases significantly if we classify adequate responses in the ‘Other’ category as related to the field. Of the 42.3% in this category, half had a job that was classified by the researcher as related (e.g. electronic engineer, CEO of a software company, mechanical engineer). In total we can thus conclude that there are 42.3% of the respondents that are working in a field related to development or engineering, which seems to be somewhat associated with the nature of the Arduino platform. Originally intended for students and artists, it is also noteworthy that while 27.6% are students and 6.5% are member of the academic community, and 2.4% of the respondents are artists. This might be an indicator that Arduino is used as a learning tool for students, and that also artists gain utility from using the platform.

To conclude the analysis of characteristics, it is of use to have a look at the educational and occupational profiles of the sample. This data is useful to estimate the accessibility of the platform and whether it is open to a large audience or not. Most of the respondents indicated holding a Bachelor's degree as their highest achieved educational degree (32.5%) followed by respondents holding a high-school diploma and professional degree, with 28.5% and 15.4% respectively. An aggregated 17.9% of the respondents, however, indicated holding a Master's or doctoral degree, which speaks for the heterogeneity of the sample. Judging from these percentages we can conclude that people participating in open-source hardware come from a variety of educational backgrounds, albeit that the majority has some form of formal education. This could indicate that some form of knowledge is necessary to partake in the project, even though this cannot be directly derived from the data.

#### *4.1.1.2. General Findings*

One of the first questions of the survey was aimed at getting a preliminary understanding of the motivations of the respondents, by generally asking, "Why do you participate in the open-source hardware project?" Respondents could indicate as many options as desired, with the answer options being characterized approximately by the categories that have been operationalized later on in the study. From this question, we can derive some valuable insights: First, we note that a very high amount of people responded to partake in the Arduino project out of fun considerations, with 90.2% indicating that "developing is fun". Moreover, 77.2% of the respondents used Arduino to improve their development skill, which becomes especially interesting given that a relatively high amount of the respondents are students, whose main aim is to enrich their knowledge. Apparently, the amount of people partaking in the Arduino community that have a need to create, change or extend their own hardware is also very high, with 95 respondents mentioning that this is one of the reasons for their participation. This might be seen as initial evidence for a needs based used of the platform. Having mentioned a 'creativity' dimension previously, which is due to the platform being originally intended for artists and creative users, special attention should be paid to the people indicating to use Arduino as an outlet for their creativity. In fact, 83 respondents, representing 67.5% of the sample, answered that this is one of the considerations why they participate in open-source hardware projects. It is especially interesting, considering that one does

not intuitively link development to creativity, even though it appears to be a large driver within the Arduino community.

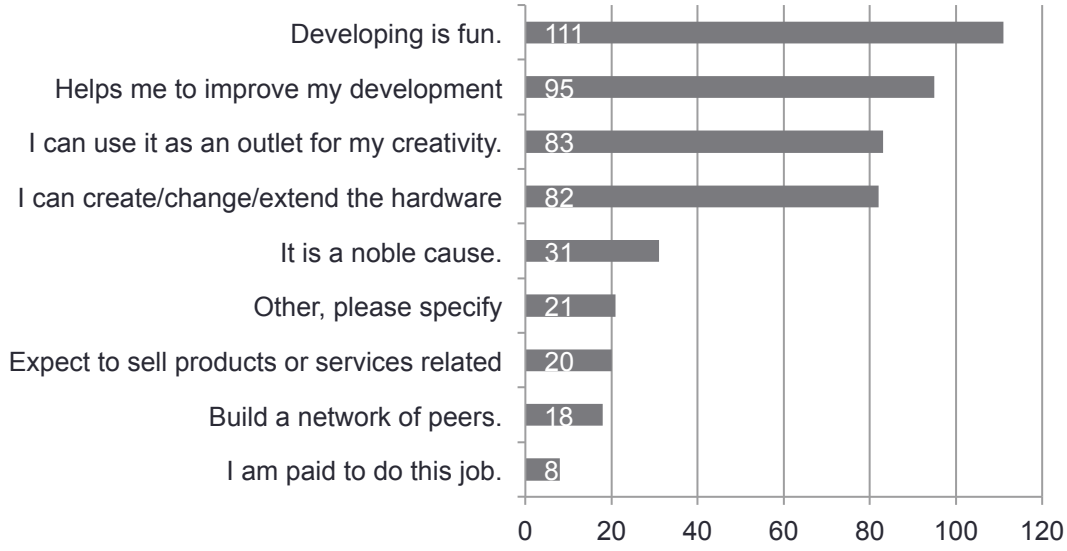


**Figure 3. Why do you participate in open-source hardware projects? Multiple answers possible.**

To further assess the creativity dimension, it was decided to also include a question on how users judge their working experience with Arduino with other creative experiences, in terms of creativity. The question at hand received an overwhelming positive response, with 91.1% of the respondents indicating that working with Arduino is equally as creative (66.7%) or even their most creative effort (25.2%) compared to their most creative experience. Clearly, this indicated that participants do make a link between building open-source hardware and creativity. Going even further and considering the nature of the project, focusing on physical objects rather than software alone, a large percentage indicated that this particularity made it much more enjoyable to develop hardware compared to just programming software, with 40.7% of the respondents indicating so. However, the majority of respondents, 52.8%, indicated no preference.

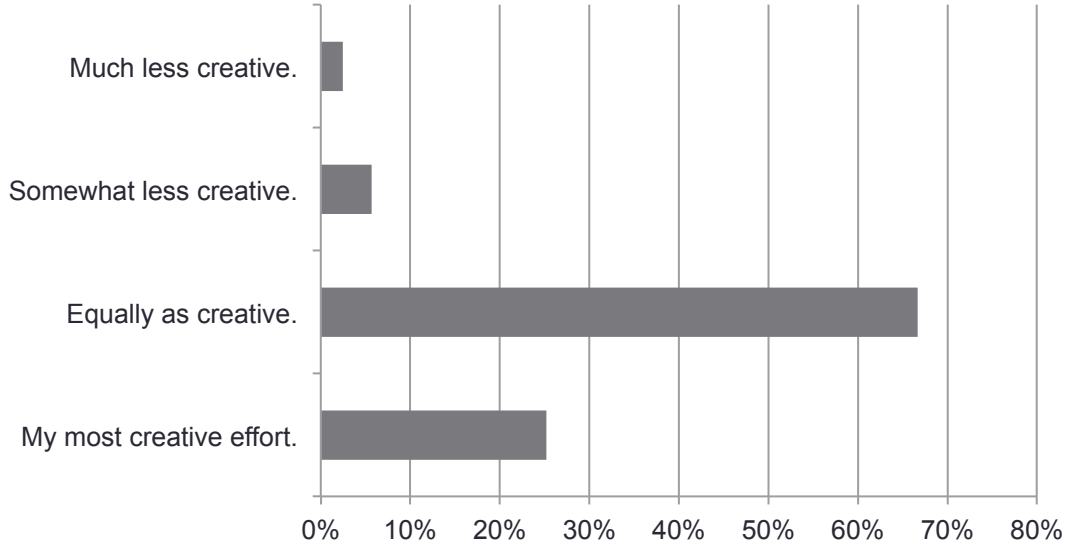


**Figure 4. Compared to my most creative experience, working with Arduino is…**

Concluding the preliminary analysis, the question of what people do with a finished project will be answered. Multiple answers to this question were possible; due to the fact that what can be done with a finished project is not mutually exclusive. As was expected in an open-source community, a major part of the respondents reflected a voluntary sharing behavior of the source code and circuit plans, with 67.5% of the respondents indicating that they would freely share their project design and code upon completion. Interestingly enough, this sharing behavior goes beyond merely sharing schematics or the code itself, 49.6% of the users also indicated to share videos and/or pictures of their creations with the community. Combining this finding with the fact that 40.7% of the respondents indicated that they asked for feedback from the community to further improve their projects, the sharing behavior could be interpreted as a way of looking for feedback. However, it might also mean that users are trying to gather recognition from their peers, a dimension which was not explicitly tested. 34.1% of the respondents deconstruct their projects upon completion and start a new project, which indicates that they derive more utility from the process of developing a project itself, than from the final outcome. Nevertheless, there are 28.5% of users who state that they have their prototypes printed, which is a means of manufacturing a circuit board on the base of their prototype, which has all the wirings and circuits printed on a board. Interestingly enough, even though users state that their boards get printed, only a marginal amount of respondents indicate that they sell their works to companies or private persons, with only 7 respondents (5.7%) stating

to do so. This gives further evidence for the non profit-seeking character of the Arduino community.

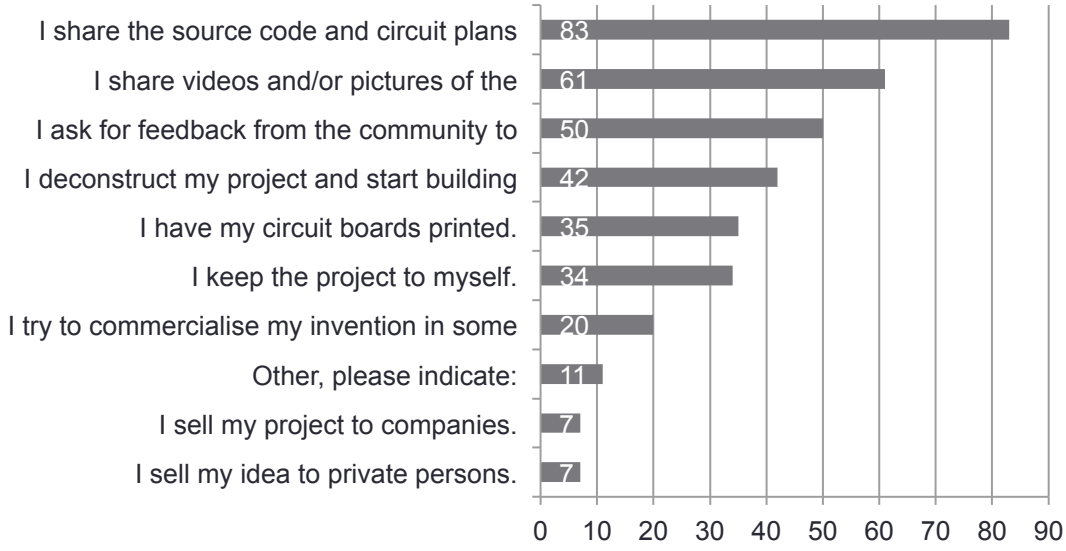


**Figure 5. Usually, what do you do you do with a finished Arduino project? Check all that apply.**

Summing up, the preliminary analysis of dimensions related to motivation show interesting results: Members of the Arduino community seem to partake in the community due to fun, self-improvement, needs-based and creativity considerations. They seem to value creativity, while not putting a great importance of monetary compensation for their efforts. While these preliminary insights give a first impression of the community itself, the following analyses will give a more in-depth insight into the motivational categories.

#### *4.1.1.3. Motivational Categories*

Having gained a better understanding of the overall characteristics of the Arduino forum user community, with regard to their demographics and behaviour, a closer look should be taken at the categories that were used to gauge the motivations of the community. Generally speaking we can say that the user community overall seems to be very motivated by a variety of aspects, which becomes evident when comparing the means of the categories. Combining all variables of a category, calculating their means and creating aggregate mean scores for the six categories, we find that the means all show a positive deviation from the neutral value, with the exception of direct rewards, which indicates an opposing deviation.

| | N | Mean |
|---|---|---|
| Enjoyment-based | 123 | 5.86 |
| Community-based | 123 | 5.38 |
| Direct rewards | 123 | 2.61 |
| Personal needs | 123 | 4.98 |
| Future rewards | 123 | 4.19 |
| Creativity | 123 | 5.66 |

**Table 1. Comparison of means of overall categories.**

Confirming our preliminary considerations above, the enjoyment-based and creativity categories together with the community-based category seem to be the dimensions among which respondents seemed to be most motivated, with mean scored of 5.86, 5.66 and 5.38 respectively. Giving a clear indication that these categories seem to be a driving force for motivation, as opposed to direct rewards, some interesting findings of the respective categories will be highlighted.

*Enjoyment-based motivation*

| *Variable* | *N* | *Mean* | *Std Dev* | *Minimum* | *Maximum* |
|---|---|---|---|---|---|
| *Developing hardware is fun.* | 123 | 6.21 | 1.02 | 1.00 | 7.00 |
| *I enjoy developing hardware.* | 123 | 6.11 | 1.01 | 1.00 | 7.00 |
| *Participating in the project gives me a feeling of accomplishment.* | 123 | 5.80 | 1.17 | 1.00 | 7.00 |
| *Participating in the project gives me a strong feeling of competence.* | 123 | 5.41 | 1.32 | 1.00 | 7.00 |
| *Participating in the project gives me a feeling of effectiveness.* | 123 | 5.36 | 1.38 | 1.00 | 7.00 |
| *Participating in the project is intellectually stimulating.* | 123 | 6.28 | .96 | 1.00 | 7.00 |
| *I rate my participation as an important activity for myself.* | 123 | 5.82 | 1.16 | 1.00 | 7.00 |

**Table 2. Results of the enjoyment-based motivation category.**

Within the enjoyment dimension, especially high scores with respect to the individual questions were observed, which results in a high overall mean as described above. In line with the general question asked in the preliminary analysis, users rate the statement “Developing hardware is fun.” highly, with a 6.21 (SD=1.02) score on the Likert-scale, which corresponds to a level that would be slightly above “Agree”. Similarly, users indicate to find the project intellectually stimulating (6.28, SD=0.96), and they genuinely enjoy developing hardware (6.11, SD=1.01). A relatively low score was obtained on the question “Participating in the project gives me a strong feeling of effectiveness.” which on average was answered with 5.36 points on the Likert-scale (SD=1.38)

*Community-based motivation*

| *Variable* | *N* | *Mean* | *Std Dev* | *Minimum* | *Maximum* |
|---|---|---|---|---|---|
| *You can always trust an open-source developer.* | 123 | 4.43 | 1.44 | 1.00 | 7.00 |
| *Recognition from others is my greatest reward.* | 123 | 4.87 | 1.46 | 1.00 | 7.00 |
| *Open-source developers should help each other out.* | 123 | 6.26 | .82 | 4.00 | 7.00 |
| *I deeply enjoy helping others - even if I have to make sacrifices.* | 123 | 5.63 | .94 | 3.00 | 7.00 |
| *Open-source developers are a big family.* | 123 | 5.41 | 1.22 | 2.00 | 7.00 |
| *I am proud to be part of the open-source community.* | 123 | 5.71 | 1.01 | 3.00 | 7.00 |

**Table 3. Results of the community-based motivation category.**

Dealing with a user forum, which is a community of users discussing the Arduino project, the community dimension deserves a closer inspection. In this category the most striking finding is that users almost equivocally agree to the statement that open-source developers should help each other out, receiving a score of 6.26 with a relatively low standard deviation of only 0.82. In line with this is the finding that users state that they deeply enjoy helping others, even if they have to make sacrifices (5.63, SD=0.94). Also, it can be stated that users are proud to be part of the open-source community and the community is also seen as a big family (5.71, SD=1.01; 5.41, SD=1.22). Recognition from others is not necessarily described as being a developers' biggest reward, as users on average indicate to only somewhat agree (4.87, SD=1.46). Trust among developers seems to be lower, with users on average indicating to neither agree nor disagree to the statement "You can always trust an open-source developer." (4.43, SD=1.44).

*Direct Rewards*

| *Variable* | *N* | *Mean* | *Std Dev* | *Minimum* | *Maximum* |
|---|---|---|---|---|---|
| *I am paid to work for the project.* | 123 | 2.28 | 1.71 | 1.00 | 7.00 |
| *I receive some form of explicit compensation (e.g. salary, contract) for participating in the project.* | 123 | 2.11 | 1.55 | 1.00 | 7.00 |
| *For me, working for the project is not profitable / extremely profitable.* | 123 | 3.27 | 2.04 | 1.00 | 7.00 |
| *Comparing to other hardware development jobs, working for the project is: Very poorly paid / Very well paid.* | 123 | 2.78 | 1.56 | 1.00 | 6.00 |

**Table 4. Results of the direct rewards motivation category.**

Having already observed that the direct rewards category as an indicator for a user's motivation, is signified by a low overall mean, it is of importance to understand why this is the case by taking a closer look at the individual questions that were used to measure this category. The question used were mainly aimed at determining whether user are paid to work on their projects, which seems to be the most adequate

question to evaluate whether this is a motivation for participating on the forum. In response to this users indicated that they were neither paid (2.28, SD=1.71) nor received some sort of explicit compensation for the project (2.11, SD=1.55) and that it was relatively poorly paid (3.27, SD=2.04) on average. It is noteworthy, however, that users do experience the project at least as somewhat profitable, even though it is not clear what exactly makes the participation profitable (3.27, SD=2.04).

### *Personal Needs*

| *Variable* | *N* | *Mean* | *Std Dev* | *Minimum* | *Maximum* |
|---|---|---|---|---|---|
| *The hardware is critical for my business or work.* | 123 | 3.50 | 2.02 | 1.00 | 7.00 |
| *My participation in the open-source project ensures that the hardware provides functionality that matches my specific needs.* | 123 | 5.11 | 1.53 | 1.00 | 7.00 |
| *It is hard for commercial hardware to meet my ever-changing needs.* | 123 | 4.59 | 1.55 | 1.00 | 7.00 |
| *Being able to fix problems with the hardware myself is one of the great advantages of open-source hardware.* | 123 | 5.80 | 1.18 | 1.00 | 7.00 |
| *Members of the community are valuable in fixing problems that arise when developing.* | 123 | 5.89 | 1.02 | 2.00 | 7.00 |

**Table 5. Results of the personal needs motivation category.**

Having described earlier that users might use the platform to fulfil some sort of personal needs, this dimension gives us insights on whether this is indeed the case. Overall the category is explained by a relatively high mean, and the questions used to measure the item show a contrast. Whereas users do not state that the hardware is critical for their business or work (3.5, SD=2.02), users do create the hardware as it fits their specific needs (5.11, SD=1.53), which might indicate a personal use motivation. Most striking is the fact that users participate in the community to receive support in fixing their problems (5.89, SD=1.04), which obviously indicates a personal needs based motivation.

### *Future Rewards*

| *Variable* | *N* | *Mean* | *Std Dev* | *Minimum* | *Maximum* |
|---|---|---|---|---|---|
| *Experience from the project raises my skill level of developing.* | 123 | 6.21 | .80 | 4.00 | 7.00 |
| *Because of my involvement in the project, I will be able to get a better job.* | 123 | 4.47 | 1.61 | 1.00 | 7.00 |
| *In one way or another I will make money from my participation in the project.* | 123 | 3.46 | 1.73 | 1.00 | 7.00 |
| *Participating in the project makes me more marketable.* | 123 | 4.43 | 1.71 | 1.00 | 7.00 |
| *I will sell products related to the project.* | 123 | 3.41 | 1.81 | 1.00 | 7.00 |
| *I will sell consulting, training, implementation or customization services related to the project.* | 123 | 3.16 | 1.76 | 1.00 | 7.00 |

**Table 6. Results of the future rewards motivation category.**

The future rewards category evaluates whether users participate in the project in the hope of receiving some sort of future reward, and was classified as an externalized intrinsic motivation. The biggest motivational driver in this category was found to be the fact that participating in the Arduino project will help to raise the individual's skill of developing (6.21, SD=0.8). Surprisingly, the prospect of getting a better job (4.47, SD=1.61) or being more marketable by participating in the project (4.43, SD=1.71) seems to be lower. However, these questions did not yield a score as low as the questions that were aimed at whether the user is expecting to make money in the future (3.46, SD=1.73), selling products (3.41, SD=1.81) or services (3.16, SD=1.76) related to the project.

*Creativity*

| *Variable* | *N* | *Mean* | *Std Dev* | *Minimum* | *Maximum* |
|---|---|---|---|---|---|
| *When I work on a project, it is just like composing poetry or music.* | 123 | 5.20 | 1.49 | 1.00 | 7.00 |
| *A finished project is like a piece of art to me.* | 123 | 5.81 | 1.28 | 1.00 | 7.00 |
| *Building objects makes open-source hardware development especially enjoyable.* | 123 | 5.98 | 1.02 | 2.00 | 7.00 |

**Table 7. Results in the creativity motivation category.**

The creativity category measures whether the creative aspect of the Arduino project can be seen as a motivational driver for users participating in the project. This category indeed seemed to be scoring rather high on the questions that were asked within the category: Users indicated that the nature of building objects made open-source hardware especially enjoyable (5.98, SD=1.02). Additionally they agreed that their finished projects were comparable to pieces of art itself (5.81, SD=1.28) as well as art forms such as poetry or music (5.2, SD=1.49).

### 4.1.2. Discussion of Descriptive Analysis

Having comprehensively elaborated on the results of the descriptive analysis, these results will now be discussed and interpreted in light of the reviewed literature on open-source communities and motivations of members hereof. In terms of general characteristics, our sample is similar to the ones found in comparable studies focusing on open-source software. Apart from the basic demographics discussed earlier, the usage behaviour on the project also shows a degree of similarity: Comparing the hours spent on open-source hardware projects, to that of hours spent on open-source software projects, as evaluated by Lakhani and Wolf (2003), most

users spent between 1 and 12 hours on the projects. Interestingly though, the amount of hours spent tends to be higher in our study with the majority spending between 6 and 12 hours, as opposed to 1 to 5 hours in the software study. This finding might be interpreted in such a way as that it may require more time to build physical objects, due to the assembly of parts in addition to only software development.

In terms of occupation we can also witness a difference, compared to previously conducted studies. As opposed to the study by Lakhani and Wolf (2003), less people participating in the Arduino community were working in a job related to the field itself. Only 42.3% of the users are working in a related field, which seems especially low considering that the open-source software studies count a higher amount for programmers alone, leaving out professions like IT managers or system administrators. In previously mentioned study, 71.1% of the participants worked in the field.

Comparing the percentage of artists that use the platform (2.4%) to the percentage of people in the US labour force ("Occupational Employment and Wages, May 2013, 27-0000 Arts, Design, Entertainment, Sports, and Media Occupations (Major Group)," 2014), we can see that there is an overrepresentation of artists, as the percentage of actual people working in the field in the US in only 0.07%. Thus, there is a higher percentage of artists in the Arduino community, than there is in the overall US labour force, giving some more evidence of the composition of the Arduino user group. This number shows that the platform indeed seems to be popular among artists.

In terms of experience in the field, the sample in the study at hand seems to have less experience, measured in years. Given that the study by Lakhani and Wolf (2003) found a software development experience of 11 years on average, the values in this study, which are ca. 9 years for software development and 6 years for hardware development, seem to be low. However, if we consider that Arduino is made to be accessible for everyone, these numbers still can be considered to be high. Given Arduino's nature of being a beginner platform, it was expected that the numbers would be even lower. Especially noteworthy is the fact that there seems to be a great variety in terms of experience, with large standard deviations indicating so.

Concerning the educational level, we compare data to the study performed by Hars and Ou (2001). Non-college degrees made up 24% of the sample in the previous study evaluating the motivations of open-source software developers, and if we compare this number to the amount of people who have hold a high-school diploma as their highest achieved educational degree, the difference is only marginal. People holding a master's degree make up a lower percentage in the present study, whereas PhD holding people make up a slightly higher percentage. Overall, the differences between the two studies seem to be relatively low however. Therefore, no interpretation for this topic can be undertaken.

To take part in open-source hardware projects, users also invest a lot of time that could be used otherwise, which shows clearly how much they value their own participation. In line with earlier research by Lakhani and Wolf (2003) users are willing to give up sleep and social time. However, we have also found that a significant cost incurred by the hardware community is the cost for hardware parts. This is especially interesting, considering that in previously mentioned study only 23.4% indicated the cost of materials for development an important factor, compared to 69.9% of the participants in the present study. While this might be an evident finding, it gives us evidence to assume that different mechanisms are necessary to facilitate open-source hardware creation compared to the mere creation of open-source software.

Overall it is striking that users seem to be motivated by a variety of aspects, which is in line with the findings by Hars and Ou (2001), who find that both internal and external factors play an important role, in their research on open-source software. It should be noted, however, that our sample shows overall very high scores without a clear distinction that can be made, with the exception of the pay category.

A closer look should be taken at the creativity findings, especially with regard to how users judge their experiences compared to their most creative experiences. Even though we did not include a control group of people only participating in open-source software projects, we can compare the level of creativity to the data collected in the study by Lakhani and Wolf (2003), which gives preliminary confirmation to our assumption that the creativity dimension plays an important role in the Arduino community. Earlier we saw that 91.1% of the respondents judged their experience of working on an Arduino project as their most creative experience, or at least equally

as creative. Compared to 61.7% in the study on open-source software this number gives a clear hint at the particularity of Arduino being targeted at a community that uses the platform as a creative outlet. In fact, creativity seems to play a major role in motivating people to work on Arduino projects, which is confirmed by a very high mean score when it comes to the creativity motivation category.
All in all, the results of the present study speak a clear language: Participants offer a significant amount of time and incur costs to work on Arduino projects. Clearly they seem to be motivated by a variety of factors, which are ranging from purely intrinsic motivational factors to externalized intrinsic motivational factors. Users are taking part because of the enjoyment they derive form working on projects, from being part of the community, in order to make their creations fit to their personal needs and to improve themselves. However, they seem not to be motivated equally as much by direct rewards, meaning by being paid to work for the project, based on the descriptive analysis at hand. To get an even better understanding, we will now proceed to an exploratory factor analysis, which will give us insights into wheter there is an underlying structure among the six categories chosen for this study.

### 4.1.3. Exploratory Factor Analysis

In order to give a better understanding with respect to the categories in which motivation matters in the study at hand, the data was subjected to an exploratory factor analysis. So far there is no research, which evaluates certain patterns among variables measuring the motivations of open-source hardware developers. By performing this analysis we hope to find factors that describe the underlying data of the dataset, which was obtained from the questionnaire.
In a first step we checked whether the scores for each category could be aggregated into a single variable for each category to remove complexity, through a reliability test, which measures the internal consistency of the questions that were used to measure the categories. By determining a statistic, Cronbach's Alpha, we deemed that the scores overall had an acceptable to high internal consistency with respect to the category that they were intended to measure (Kline, 2013). Therefore, it was decided to aggregate the scores and to create six latent variables, in line with the categories, that represent an average of all the questions measured per category.

| | *Cronbach's Alpha* | *N of Items* |
|---|---|---|
| ENJOYMENT | .84 | 7 |
| COMMUNITY | .81 | 6 |
| PAY | .74 | 4 |
| NEEDS | .62 | 5 |
| FUTURE | .84 | 6 |
| CREATIVITY | .78 | 3 |

**Table 8. Internal consistency test: Cronbach's Alpha.**

The following step included taking a closer look at the amount of factors that should be taken into consideration in the case at hand. A preliminary analysis aimed at determining the number of factors, yields a satisfactory result with two factors. Two factors are chosen by the Kaiser criterion, stating that only factors with eigenvalues of the correlation matrix higher than one should be chosen for a subsequent factor analysis (Kaiser, 1960). Using this rule, our factors will explain a total of 60.6% of the variance. Additionally a visual representation, the scree plot, showed a clear elbow point, which yields a satisfactory result with two factors. The two factors obtained will explain 38.4% and 22.1% of the variance, respectively.

| Num | Eigenvalue | Difference | Proportion | Cumulative |
|---|---|---|---|---|
| 1 | 2.305211 | 0.9766227 | 0.3842 | 0.3842 |
| 2 | 1.3285883 | 0.6032131 | 0.2214 | 0.6056 |
| 3 | 0.7253752 | 0.0925788 | 0.1209 | 0.7265 |
| 4 | 0.6327964 | 0.085153 | 0.1055 | 0.832 |
| 5 | 0.5476434 | 0.0872577 | 0.0913 | 0.9233 |
| 6 | 0.4603857 | . | 0.0767 | 1 |

**Table 9. Eigenvalues of the correlation matrix.**

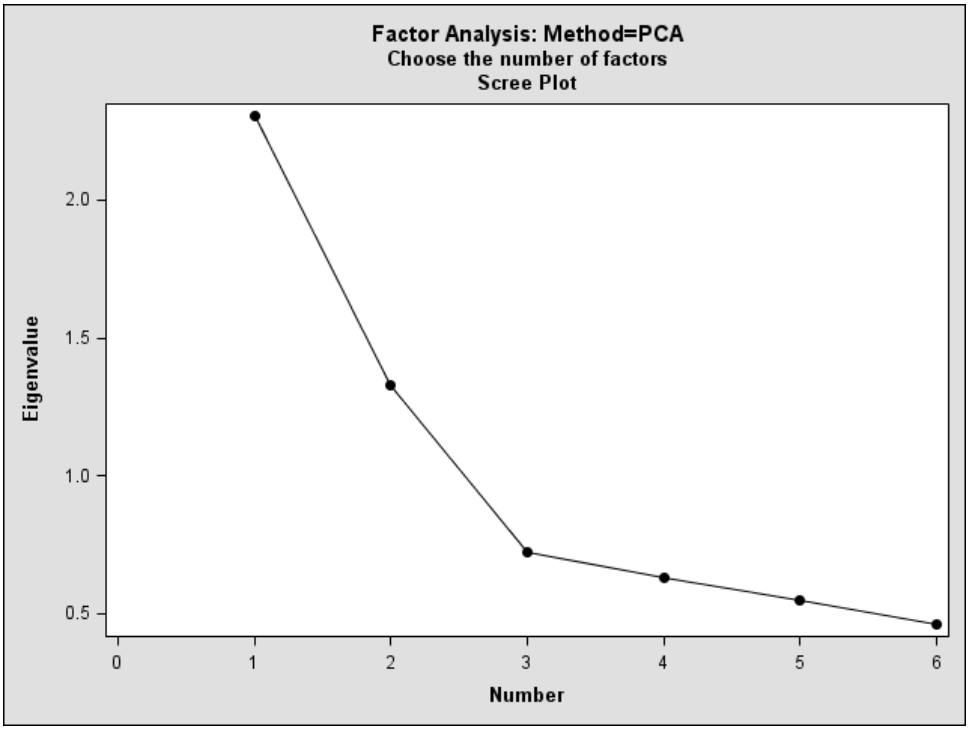


Table 10. Scree Plot.

To obtain a more detailed understanding with respect to the proportion of explained variance, it is of use tot take a look at the cumulative proportion of variance explained by the factors, which will give us insights into how well captured the variables are within the two factor model. The two factor model shows us that all variables are relatively well explained, with the exception of the *ENJOYMENT* dimension, which appears to be slightly below a cumulative 50% of variance explained threshold in the two-factor model. Noticing, however, that it is reasonably well explained at a 36% level in the first factor we decide to proceed with the two-factor model, also taking into consideration that all other variables' variance are explained at a 50% level.

| Obs | _NAME_ | PROP_PCA_1_1 | PROP_PCA_1_2 |
|---|---|---|---|
| 1 | ENJOYMENT | 0.36 | 0.46 |
| 2 | COMMUNITY | 0.42 | 0.62 |
| 3 | PAY | 0.23 | 0.74 |
| 4 | NEEDS | 0.54 | 0.56 |
| 5 | FUTURE | 0.41 | 0.69 |
| 6 | CREATIVITY | 0.33 | 0.56 |

Table 11. Cumulative proportion of variance explained with regard to the original variables.

Proceeding with a two-factor solution, the rotated factor loadings, using a Varimax rotation were obtained, which give evidence of an underlying latent factor structure in the dataset. From the factors, we can see that there are two latent variables, with their own specifics: The first factor explains the *COMMUNITY, CREATIVITY* and *ENJOYMENT* dimensions, which were previously mentioned to be purely intrinsic motivations (*COMMUNITY, ENJOYMENT)* as well as the additional measurement of *CREATIVITY* which we have found to be important in the case at hand. The other factor gives good explanation for *PAY* and *FUTURE*, which refer to purely extrinsic and internalized extrinsic motivations, which mostly revolve about receiving some monetary benefit. *Needs-based* motivations are almost equally well explained in both factors, with a rotated factor loadings of .50 and .55 for the factors, respectively.

| Rotated Factor Pattern | | | | |
|---|---|---|---|---|
| | **Factor1** | | **Factor2** | |
| **COMMUNITY** | 79 | * | . | |
| **CREATIVITY** | 75 | * | . | |
| **ENJOYMENT** | 67 | * | . | |
| **PAY** | . | | 86 | * |
| **FUTURE** | . | | 81 | * |
| **NEEDS** | 50 | | 55 | * |
| **Printed values are multiplied by 100 and rounded to the nearest integer. Values greater than 0.550288 are flagged with an '*'. Values less than 0.3 are not printed.** | | | | |

Table 12. Rotated factor pattern after Varimax rotation.

### 4.1.4. Discussion of Exploratory Factor Analysis

To uncover the underlying structure of the variables in the study at hand without any *a priori* hypotheses, the exploratory factor analysis gives us important insights into what main underlying concepts of motivation can be found in the field of open-source hardware development, and especially the Arduino community. Through this dimension reduction procedure, it was discovered that there are two main factors that can be used to describe the motivations among open-source hardware developers. The first factor describes very well the motivations that are mainly of an intrinsic nature: community-based and enjoyment-based motivations as well as the creativity dimension that was included in the study. Considering that these variables can be grouped into one factor, this factor could be labelled “fun and creativity” as it describes these dimensions sufficiently. Identifying this latent construct helps us to understand that there is indeed an underlying construct which influences responses

on the variables. It can be inferred that these dimensions, which are conceptually similar in nature have a big influence on the motivations of developers.
The second factor yields an equally interesting insight: It appears that there is a factor, which influences responses that deal with the reward-based motivational categories: the purely extrinsic *direct rewards* category and the internalized extrinsic variable of *future-rewards.* Uncovering this structure, we can infer that there is a pattern whose factor can be labelled “rewards”. The results show that the response patterns in these two categories are similar, which gives additional insights with regard to the descriptive analysis earlier, in which the *direct rewards* and *future rewards* category were among the lowest in terms of to what extent respondents agreed to the statements provided in these categories.
Overall the factor analysis and the subsequent labelling of the categories shows us that there are indications for a dichotomy between “fun and creativity” and “rewards” considerations. This leads us to the conclusion that developers are indeed motivated differently within these two categories, a finding which has been confirmed loosely in the descriptive analysis earlier. Careful note should be taken on the variable that is virtually equally explained in the two factors that we derived from the analysis: *needs-based* motivation. It can be inferred that this dimension has equal importance, due to the fact that open-source hardware development originates from the fact that people see a need to create something that addresses a certain need they, which is at the very core of the open-source movement, as described earlier. Given its importance in both factors, we can interpret this result as that it is an overall driver that both influences the “fun and creativity” and the “rewards” dimension, which is in line with the discussion on how open-source communities come into being in the first place. It is especially relevant as that the Arduino microcontroller was invented to address a need and that it is now used both for the purpose of building artistic objects as well as to generate money or receiving some other form of reward by building smart objects.

### 4.2. Supplementary Data

Having collected additional quantitative data by matching the respondents to their user profiles, supplementary data was obtained which can be employed to get a better picture of the forum usage behaviour and a possible relation with the motivation of the users in the Arduino forum. In total we obtained five additional

variables: *posts_number,days_registered, karma, spent_online, topics_started.* Therefore, we will first have a look at the supplementary data in general before proceeding to combine the data with the survey responses.

| | | posts_number | days_registered | karma | spent_online | topics_started |
|---|---|---|---|---|---|---|
| N | | 80 | 80 | 80 | 80 | 80 |
| Mean | | 1251.64 | 323.98 | 18.64 | 13.034957879 | 34.94 |
| Median | | 12.00 | 51.00 | .00 | .298969445 | 2.00 |
| Std. Deviation | | 4711.253 | 456.534 | 68.375 | 38.8281645415 | 187.714 |
| Percentiles | 25 | 2.25 | 20.00 | .00 | .091840278 | 1.00 |
| | 50 | 12.00 | 51.00 | .00 | .298969445 | 2.00 |
| | 75 | 304.75 | 528.00 | 3.75 | 3.994270834 | 12.75 |

**Table 13. Statistics of Supplementary Data.**

With regard to the number of posts, the dataset is characterized by a high variance, resulting in a high standard deviation. Even though the mean number of posts tends to be high with approximately 1252 posts per user, this high number is mainly due to some extreme values, which represent very active users that show a disproportionately high number of posts. This finding is especially striking if we take a look at the median as a measure of tendency, which has a value of 12, and is thus very different from the mean, further indicating a high dispersion. Correcting for this, if we trim the mean by eliminating the most extreme five per cent of observations, we get a more realistic estimate of 345 posts per user, which still appears to be high, even though it is significantly lower than the previous mean. Similarly, we observe that these outliers also exist in the other supplementary data variables, however they appear not to be as pronounced as in the variable measuring number of posts. Since we do want to get a realistic picture and the community seems to be characterized by a high dispersion of posts number in general, it was decided to keep the outliers in the dataset for later analysis.

| | | | *Statistic* | *Std. Error* |
|---|---|---|---|---|
| *posts_number* | *Mean* | | 1251.64 | 526.73 |
| | *95% Confidence Interval for Mean* | *Lower Bound* | 203.20 | |
| | | *Upper Bound* | 2300.08 | |
| | *5% Trimmed Mean* | | 345.08 | |

**Table 14. Trimmed mean of posts_number.**

The other variables, listed in Table 13 are also characterized by having a relatively high standard deviation in all variables. Given the high dispersion and high ranges of all of the variables, there seems to be some evidence that there are different sorts of users, which use the forum to a different extent and possibly for different purposes. In line with the theory described earlier, we assume that many of the users, which are on the forum, are only there to obtain information and observe conversations, not to participate. Therefore, we assume that there is a large number of so-called “lurkers”. As expected, there is a high correlation between the supplementary data variables, all of which are significant at a .01 level in a one-tailed test, as it was assumed that there is only a positive correlation between the variables. From the dataset, we obtain the information that the most striking correlations are those between the number of posts and the karma given, which are virtual points given for valuable contributions, as well as the amount of posts and time spent online on the forum. Interestingly enough, a relatively low correlation has been found between topics started and days registered as well as karma, even though this correlation still has been found to be significant.

| | | posts_number | days_registered | karma | spent_online | topics_started |
|---|---|---|---|---|---|---|
| posts_number | Pearson Correlation | 1 | .562** | .975** | .866** | .645** |
| | Sig. (1-tailed) | | .000 | .000 | .000 | .000 |
| | N | 80 | 80 | 80 | 80 | 80 |
| days_registered | Pearson Correlation | .562** | 1 | .561** | .575** | .373** |
| | Sig. (1-tailed) | .000 | | .000 | .000 | .000 |
| | N | 80 | 80 | 80 | 80 | 80 |
| karma | Pearson Correlation | .975** | .561** | 1 | .760** | .476** |
| | Sig. (1-tailed) | .000 | .000 | | .000 | .000 |
| | N | 80 | 80 | 80 | 80 | 80 |
| spent_online | Pearson Correlation | .866** | .575** | .760** | 1 | .860** |
| | Sig. (1-tailed) | .000 | .000 | .000 | | .000 |
| | N | 80 | 80 | 80 | 80 | 80 |
| topics_started | Pearson Correlation | .645** | .373** | .476** | .860** | 1 |
| | Sig. (1-tailed) | .000 | .000 | .000 | .000 | |
| | N | 80 | 80 | 80 | 80 | 80 |

**. Correlation is significant at the 0.01 level (1-tailed).

**Table 15. Correlations between Supplementary Data Variables.**

Now that we have a better understanding of the supplementary variables in general and the correlations between the variables, it becomes interesting to see whether there is any relationship between these data and the motivational categories. It will be interesting to see whether highly active users show different motivational patterns, as opposed to users who are not as active on the forum. As has been described before, it was decided to keep the outliers in the dataset. To be able to compare highly active users versus users with a low activity level, as derived from the supplementary data, grouping variables were created. These grouping variables put the observations in the respective supplementary data variables into either a high or low category, depending on whether the value observed is above or below the median of the respective variable. The grouping variables equalize extreme values by putting the most active users in one category, irrespective of their activity level. Therefore we obtained five new grouping variables, which indicate whether the number of posts, days registered, karma, time spent online and topics started is considered “high” or “low”. These grouping variables were used to perform an independent-samples t-test for each supplementary data variable, testing for a difference in the means with respect to the motivational categories. The output of this analysis can be found in Appendix 4.

From the output of the independent-samples t-test, it can be seen that none of the results are significant ($p > 0.05$). We have tested all supplementary data variables on the motivational categories, with no significant results indicating a difference in the means of the five motivational categories. Therefore, we can state that there is no difference in the motivations of open-source hardware developers with respect to their activity level. To give an example, this means that users who are characterized by a very high level of activity on the user forum are not motivated more on a community-based motivational level than users with a low activity level on the forum. Overall, the outcome of the supplementary data analysis shows that there is high dispersion in activity level among users of the Arduino user community. Also, it could be observed that users, who are very active in terms of number in posts, spend a great amount of time on the user forum as indicated by the correlations. With respect to the topic of the thesis at hand, we observed that the activity level has no impact on the motivational categories, meaning that all users, irrespective of their usage behavior, show similar motivational characteristics.

### 4.3. Interview Findings

The interview with the Global Director of Creative Technologies, Nico Abbruzzese, who works at *Metalworks by Maxus*, a business dealing with media planning and advertising, proved to be very insightful when it comes to looking at a management perspective of dealing with open-source hardware developers. Part of the media agency *Maxus*, *Metalworks* is a company that focuses on rapid prototyping to build physical objects that solve business problems of a client, mostly in the field of marketing. The developers primarily involved with developing objects by means of open-source hardware platform, are called creative technologists in the firm, which shows that the creativity dimension plays an important role. Following their work for a period of time, it was interesting to see how these developers used the Arduino platform to build objects that could solve specific problems.

Interviewing Nico Abbruzzese who was in charge of managing these creative technologists on a daily base, helped to gather valuable insights that can be used to learn what makes an optimal corporate environment, which is conducive to innovation. In the following paragraph, these insights will be highlighted, before proceeding to the discussion of the interview.

One of the first insights gathered from the interview was the comparison of open-source hardware developers to craftsmen, i.e. people who specialize on a certain craft. Through the interview, we learnt that craftsmen take great pride in their work and are highly specialized on performing a single task. Nico Abbruzzese mentioned as an example craftsmen who build model ships from scratch, a task that takes a long while to achieve perfection. According to Nico, for these people most joy can be derived from the actual process of building the ship instead of merely the outcome. From the interview it became clear, that this intrinsic motivation also appears to be a strong driving force in open-source hardware developers, and the most successful of these have a motivation which is mostly focused on the process, rather than just the outcome.

Taking the notion of comparing open-source hardware developers further, Nico Abbruzzese elaborated that open-source hardware developers are the craftsmen of modern times. While it was usual for craftsmen in the past, e.g. carpenters, shipbuilders and even artists, to travel the world to perfect their skill and learn more about the craft they engaged in, this phenomenon has been replaced in our time with

the Internet. Facilitating communication and being able to find communities that are interested and specializing in the same topic, which for example is the case in the Arduino community, focusing on one platform, helps craftsmen to find like-minded people. In the case of open-source hardware development, these like-minded people are fellow open-source hardware developers who like to exchange ideas and concepts in order to improve their skills. It was interesting to note, how Nico Abbruzzese emphasized the importance of these communities in order for people to meet. In the absence of the Internet, he stated, there was no other possibility than to travel long distances to meet people who shared the same interest, passion or profession. From the interview, it was clear that the Internet creates a platform, which is invaluable to developers who want to find like-minded people.

Nico Abbruzzese continued to elaborate on the community aspect, which is very important to open-source hardware developers. He stated that the communities are not only important to exchange ideas and for educational purposes, but that they are especially important for developers as they present an environment in which the developers feel comfortable. In explaining this, Nico Abbruzzese expanded, that to his knowledge open-source hardware developers seldom find people who are equally interested in the topic in their private social environment and social circles. He went on to explain that if developers bring up topics dealing with hardware development, these topics are often disregarded and considered not interesting or even strange by their social peers, e.g. at their workplace. Of course this is only valid, if these developers do not work at a place where they use their development knowledge for professional purposes. Contrasting to this, open-source hardware developers feel very comfortable discussing these topics in their dedicated communities. He mentioned that it therefore is very important to create a community or environment that is dedicated to a certain topic, to make the developers feel most comfortable and in turn to create an environment conducive to innovation.

On a more general base, the interview provided also interesting insights into how the open-source hardware development movement is motivated historically. Nico Abbruzzese continued to explain that he saw a lot of parallels between the people who employ the Arduino community and people that are historically called “tinkerers”. Interestingly enough the concept of tinkering is closely related to the concept of *bricolage,* which has been mentioned earlier in this thesis. Tinkerers divert objects

from their intended use, either improving or modifying them to achieve a certain goal, according to Nico Abbruzzese. From the interview it became clear that Nico saw a clear parallel between this early tinker movement, which would for instance modify simple objects like teapot, to open-source hardware developers, who modified smart objects to add additional functionality. As an example, Nico mentioned a project of his own, which he considered a modern form of tinkering. Living in Singapore, where rainfall can be very heavy, and impeding the ease of finding a taxi due to increasing demand, Nico adapted his alarm clock by means of the Arduino platform: Attaching Arduino to a standard commercially available alarm clock, he programmed the Arduino board in such a way as that it checks the current weather conditions half an hour before the wake-up time. In case of incoming rain, it would automatically trigger a taxi call, which helped him to ensure getting a taxi on time.
Overall, the interview was very fruitful to gain insights of what it means to work with open-source hardware developers and to gain a deeper understanding of the psyche of open-source hardware developers, albeit from a single perspective. Therefore, we will now proceed to elaborate on the insights from this interview, and how we can use the insights to better understand the motivations of open-source hardware developers.

### 4.3.1. Interview: Key Insights

From the interview we can gain important insights that will help to create an environment, which is motivating for open-source hardware developers. Especially in a corporate context, creating such an environment is very important, in order to obtain innovative solutions to business problems by means of open-source hardware. Even though most innovation in the field takes place outside of corporate research centres itself, it is important to notice that these lessons can be implemented in virtually any environment in order to elicit innovation from open-source hardware developers.
To do so, it is important to create an environment in which developers feel comfortable. Taking the notion of the craftsmen and drawing an analogy to open-source hardware developers, we note that the aspect of being able to broaden one's horizon and to learn more is very important. Therefore a platform, in which open-source hardware developers discuss and exchange ideas, should be kept open so as to allow for all sorts of discussion. Ideas should not be limited, and unrelated topics

might be of use for the community to learn. For a company willing to use outside knowledge, this might mean that they should create a platform, which allows for discussion of all sorts, not restricting discussion that might appear irrelevant on first sight. Only by acting in such a way they can address the needs of learning, showing and being shown new projects and ideas, which are vital to open-source hardware developers.

Given the finding that open-source hardware developers often times lack the social circles to discuss their projects and ideas, it is important to include a strong social component when creating a platform to harness ideas. Therefore, we propose to create online platforms, which do not only provide for a strictly topic-related discussion platform but also for a social platform. Assuming that people interested in the field have an interest in sharing other ideas or just chatting for fulfilling a social need, due to their similar interests that creates a common ground, the social aspect of a community becomes very important. These communities should, however, not only be restricted to the virtual space. In fact, it is a good idea to bring open-source hardware developers physically together at dedicated meet-ups. Due to the physical nature of the projects, it would be conducive to initiate such meet-ups where developers can work collaboratively on projects, exchanging ideas and creating a common solution. Discussing this, we should take note of so-called “hackathons”, which are aimed at exactly this: bringing together developers and people from a multitude of disciplines sharing a common interest to develop and innovate. Learning from the interview, we propose these sorts of events as a useful tool to further innovation beyond virtual and corporate boundaries.

As for the tinkering movement that was discussed in the interview, and for the open-source hardware development movement in general, we have seen that the drive to start working on a project comes from a specific problem. Therefore, it is important to constantly spark interest in developers by presenting specific problems. Taking the insight that developers are more interested in the process of developing itself, rather than the outcome, platforms could be instated that present problems in clear way, which would spark the interest of developers willing to solve that problem, while at the same time pursuing their greatest interest. While many companies already have some sort of platform to present problems, it would be an advancement to create these problem presentations for specific platforms, which would allow developers to

use their particular knowledge: for instance by creating a search contest which has to be solved with a singular platform, e.g. the Arduino platform.
As we can see, there are many opportunities to motivate open-source hardware developers and to obtain innovative ideas and products in turn. By creating an optimal environment and addressing the motivational needs of developers, corporations can gain valuable ideas and innovative products. The interview showed that this is indeed true, speaking to a practitioner in the field personally. Combining the interview with the quantitative findings, we can thus proceed to the managerial implications, elaborating on how the insights from this study can be used to foster innovation in a corporate environment as well as user entrepreneurship.

## 5. Managerial Implications

The study at hand has focused on the motivations of open-source hardware developers, in particular those developers that engage on the user forum of the Arduino prototyping platform. From the analysis, we have garnered a variety of insights, which can be used in a managerial context to foster innovation. The following managerial implications can be used to make the most out of collaboration with open-source hardware developers, which is especially interesting as it is an evolving field of interest.
First of all it is important to mention that there is no isolated motivational factor that describes the motivations of open-source hardware developers, therefore we propose a multi-perspective approach in creating an environment that is conducive in fostering the innovation of physical objects and artefacts. The descriptive analysis has shown that the participants in the study are rating a multitude of motivational categories positively. Therefore, we assume that isolated efforts focusing solely on category will not yield the positive effects required. Managers should be aware not to focus overly on one category, for instance by creating a great community feeling without making the hardware development task itself enjoyable.
Moreover, we have seen that pay seems to be the least motivating driver in the study as compared to the other categories. This result can be applied in such a way as to not communicating a possible award prominently to gain innovative ideas from open-source hardware developers, in company sponsored innovation initiatives. In contrast, we suggest putting the focus on the enjoyment aspect and community

aspect of such an initiative. Another idea would be to invite peer-recognized hardware developers that possess special knowledge onto innovation platforms, so as to enrich the knowledge of the participants, which was considered to be an important motivational driver.

Additionally, we suggest companies to take a rather passive role in creating innovation platforms for open-source hardware developers. Even though not specifically tested, we have learned form the interview that the freedom to try things and “tinker” with physical objects is a very important consideration in the open-source community. In addition, the historical overview has shown, that many open-source developers call for free software and hardware. In this case, free should be seen not only in monetary terms but especially in conceptual terms. Not intervening too much with innovation platform initiatives from a corporate side will maintain this notion of freedom while at the same time giving more possibilities to hardware developers to tackle business problems from various perspectives.

What concerns the nature of open-source hardware being physical objects; this dimension should be emphasized. In contrast to open-source software, where the outcome is usually merely visible on a computer screen, the physical dimension brings new challenges to using open-source communities for innovation. First of all physical objects require hardware parts which cannot be duplicated at virtually zero cost, compared to software. To foster innovation, companies or interested individuals could set up labs for open-source hardware developers to create new and smart objects. These labs will offer a variety of hardware parts to experiment with. By doing so, a barrier to entrance to this field will be removed, as users can try a variety of solutions to problems, taking different hardware parts to gain an optimal outcome, without the need of a significant investment in these hardware parts.

An additional contribution of this study was to show that users judge their Arduino projects as a very creative endeavour. While it is beyond the scope of this study to discuss how to improve and foster creativity, we would like to emphasize the importance of this dimension. In order to gain unusual solutions to (business) problems, a high degree of creativity is required. The Arduino community considered being highly creative from the study at hand, we encourage companies and individuals to partake in this community to gain unusual solutions to problems. Offering a vast variety of perspectives, as has been shown by the educational and

occupational backgrounds, ranging from engineers to artists, the Arduino community appears to be a great source of creative potential. While many companies already use a multi-perspective approach to innovate (e.g. IDEO), innovation platforms could be used to gain this multitude of perspectives without having to be physically present. Therefore, we propose to effectively use communities for idea generation so as to solve business problems. We will now proceed to the conclusion of this study, including its limitations and opportunities for further research.

## 6. Conclusion

While open-source software development has been studied extensively, less light has been shed on the emerging phenomenon of open-source hardware. This study aimed to give a better understanding on one facet of open-source hardware development: the motivations of open-source hardware developers. Using the most popular open-source hardware development, Arduino, as a starting point to obtain knowledge about the motivational considerations of developers, we have found a number of interesting insights that can help to foster innovation in the open-source hardware space.

In light of the existing literature on the motivations of open-source software developers, we have developed motivational categories aimed at understanding the motivations of hardware developers. Even though there are various ways of defining and measuring motivation, motivation in the present study was captured in five dimensions, which refer to enjoyment-based, community-based, needs based, direct and future rewards motivational drivers respectively. Moreover, an additional *creativity* category was found to be useful for further analysis, given the background of the platform.

From the quantitative analysis, we obtained the understanding that Arduino developers are motivated by a variety of aspects, with the least important one being the one of direct rewards. This dimension only plays a role to a certain degree in the open-source hardware developer community. Finding a solution for the puzzle of why people engage in open-source hardware development, despite there being no economic reasoning or theory to do so, has shown us that money is indeed not required strongly to motivate developers in their community. Instead, the other

motivational factors are important and give a good explanation as to why people engage in open-source hardware developers.

Through an exploratory factor analysis, we further obtained evidence that there seem to be two underlying latent factors, which have been labelled as a “fun and creativity” factor and a “rewards” factor. Considering these two factors, it is interesting to note that the former factor can be linked to intrinsic motivations, whereas the latter factor can be linked to more extrinsic motivations, in the light of motivation theory. This underlying variable pattern gives initial evidence for a possible dichotomy between intrinsic and extrinsic motivations in the case of open-source hardware development. It should be noted, that this is in line with some of the earlier research on open-source software development.

Analysing supplementary data on usage behaviour in the forum, it was discovered that the motivations of open-source hardware developers are not significantly different when comparing users who are highly active on the user forum compared to users who show little activity. Therefore, we assume that there is one sort of open-source hardware developers, who is motivated by a variety of factors. Irrespective of the participation in the user community, similar patterns can be found. A possible explanation for this pattern has been found, in the so-called “lurker” phenomenon, which has been described in the literature review.

From the interview, additional first-hand observations on working with open-source hardware developers showed that developers in the field indeed seem to be motivated intrinsically. Another interesting finding are the historical linkages that can be drawn between open-source hardware developers and craftsmen/tinkerers. This insight has shown that there appears to be an innate human drive to display certain behaviour: in the past people have tinkered with everyday objects to extend their functionality, whereas nowadays an entire community revolves around a platform that helps to obtain certain functionality.

Before concluding this thesis, some of the limitations and opportunities for further research should be presented. While the present study only focused on one community, further research can be used to obtain a better understanding of open-source hardware development in general by looking at a variety of communities. Moreover, a better understanding of people who only work in corporate environments using open-source hardware should be obtained by further research. Focusing solely

on a specific community and only on open-source hardware, this study does not allow for any direct conclusions between pure open-source software development and its hardware counterpart. Further research can be used to get a better understanding of the differences between the two counterparts. In addition, we encourage research that can help to highlight and formalize other facets of open-source hardware, such as the impact of hardware costs on development, the government of open-source hardware communities or the collaboration process, which is different due to the physical nature of the subject matter.

In conclusion, the present study has helped to highlight one of the many facets of open-source hardware development: the motivations of open-source hardware developers in the Arduino user community. The study at hand presents an overview of how to motivate open-source hardware developers, in order to gain innovative solutions to business problems. The results are especially interesting, given that open-source hardware can be regarded as a public good. A future research agenda to cover more facets of this emerging and increasingly more important topic should be pursued, to gain a better understanding similar to that open-source software. Due to the relative novelty of open-source hardware compared to open-source software, it is difficult to predict how this field will evolve. The future will tell, how companies and individuals will use open-source hardware to innovate, and whether open-source hardware will once again change the paradigm of innovation.

# 7. Appendices

## 7.1. Appendix 1 – GNU General Public License

Version 3, 29 June 2007
Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

**Preamble**

The GNU General Public License is a free, copyleft license for software and other kinds of works.
The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.
When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.
To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.
For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.
Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.
For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.
Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.
Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.
The precise terms and conditions for copying, distribution and modification follow.

**TERMS AND CONDITIONS**

**0. Definitions.**

“This License” refers to version 3 of the GNU General Public License.
“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.
“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.
To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.
A “covered work” means either the unmodified Program or a work based on the Program.
To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.
An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

**1. Source Code.**

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.
A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.
The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.
The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.
The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.
The Corresponding Source for a work in source code form is that same work.

**2. Basic Permissions.**

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.
You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.
Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

**3. Protecting Users' Legal Rights From Anti-Circumvention Law.**

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.
When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

**4. Conveying Verbatim Copies.**

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright

notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.
You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

**5. Conveying Modified Source Versions.**

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

**6. Conveying Non-Source Forms.**

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.
“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.
If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).
The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.
Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

**7. Additional Terms.**

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.
When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.
Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is

governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.
If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.
Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

**8. Termination.**

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).
However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.
Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.
Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

**9. Acceptance Not Required for Having Copies.**

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

**10. Automatic Licensing of Downstream Recipients.**

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.
An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.
You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

**11. Patents.**

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”.
A contributor's “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.
Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.
If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.
If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.
A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.
Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

**12. No Surrender of Others' Freedom.**

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

**13. Use with the GNU Affero General Public License.**

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

**14. Revised Versions of this License.**

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.
Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.
If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

**15. Disclaimer of Warranty.**

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

**16. Limitation of Liability.**

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

**17. Interpretation of Sections 15 and 16.**

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

## 7.2. Appendix 2 – The TAPR Open Hardware License

Version 1.0 (May 25, 2007) Copyright 2007 TAPR – http://www.tapr.org/OHL

**PREAMBLE**

Open Hardware is a thing – a physical artifact, either electrical or mechanical – whose design information is available to, and usable by, the public in a way that allows anyone to make, modify, distribute, and use that thing. In this preface, design information is called "documentation" and things created from it are called "products."

The TAPR Open Hardware License ("OHL") agreement provides a legal framework for Open Hardware projects. It may be used for any kind of product, be it a hammer or a computer motherboard, and is TAPR's contribution to the community; anyone may use the OHL for their Open Hardware project. You are free to copy and use this document provided only that you do not change it.

Like the GNU General Public License, the OHL is designed to guarantee your freedom to share and to create. It forbids anyone who receives rights under the OHL to deny any other licensee those same rights to copy, modify, and distribute documentation, and to make, use and distribute products based on that documentation.

Unlike the GPL, the OHL is not primarily a copyright license. While copyright protects documentation from unauthorized copying, modification, and distribution, it has little to do with your right to make, distribute, or use a product based on that documentation. For better or worse, patents play a significant role in those activities. Although it does not prohibit anyone from patenting inventions embodied in an Open Hardware design, and of course cannot prevent a third party from enforcing their patent rights, those who benefit from an OHL design may not bring lawsuits claiming that design infringes their patents or other intellectual property.

The OHL addresses unique issues involved in the creation of tangible, physical things, but does not cover software, firmware, or code loaded into programmable devices. A copyright-oriented license such as the GPL better suits these creations.

How can you use the OHL, or a design based upon it? While the terms and conditions below take precedence over this preamble, here is a summary:

- You may modify the documentation and make products based upon it.
- You may use products for any legal purpose without limitation.
- You may distribute unmodified documentation, but you must include the complete package as you received it.
- You may distribute products you make to third parties, if you either include the documentation on which the product is based, or make it available without charge for at least three years to anyone who requests it.
- You may distribute modified documentation or products based on it, if you:
- License your modifications under the OHL.
- Include those modifications, following the requirements stated below.
- Attempt to send the modified documentation by email to any of the developers who have provided their email address. This is a good faith obligation – if the email fails, you need do nothing more and may go on with your distribution.
- If you create a design that you want to license under the OHL, you should:
- Include this document in a file named LICENSE (with the appropriate extension) that is included in the documentation package.
- If the file format allows, include a notice like "Licensed under the TAPR Open Hardware License (www.tapr.org/OHL)" in each documentation file. While not required, you should also include this notice on printed circuit board artwork and the product itself; if space is limited the notice can be shortened or abbreviated.
- Include a copyright notice in each file and on printed circuit board artwork.
- If you wish to be notified of modifications that others may make, include your email address in a file named "CONTRIB.TXT" or something similar.
- Any time the OHL requires you to make documentation available to others, you must include all the materials you received from the upstream licensors. In addition, if you have modified the documentation:
- You must identify the modifications in a text file (preferably named "CHANGES.TXT") that you include with the documentation. That file must also include a statement like "These modifications are licensed under the TAPR Open Hardware License."

- You must include any new files you created, including any manufacturing files (such as Gerber files) you create in the course of making products.
- You must include both “before” and “after” versions of all files you modified.
- You may include files in proprietary formats, but you must also include open format versions (such as Gerber, ASCII, Postscript, or PDF) if your tools can create them.

**TERMS AND CONDITIONS**

**1. Introduction**

1.1 This Agreement governs how you may use, copy, modify, and distribute Documentation, and how you may make, have made, and distribute Products based on that Documentation. As used in this Agreement, to “distribute” Documentation means to directly or indirectly make copies available to a third party, and to “distribute” Products means to directly or indirectly give, loan, sell or otherwise transfer them to a third party.

1.2 “Documentation” includes:

(a) schematic diagrams;

(b) circuit or circuit board layouts, including Gerber and other data files used for manufacture;

1. (c) mechanical drawings, including CAD, CAM, and other data files used for manufacture;
2. (d) flow charts and descriptive text; and
3. (e) other explanatory material.
4. Documentation may be in any tangible or intangible form of expression, including but not limited to computer files in open or proprietary formats and representations on paper, film, or other media.

1.3 “Products” include:

(a) circuit boards, mechanical assemblies, and other physical parts and components;

(b) assembled or partially assembled units (including components and subassemblies); and

(c) parts and components combined into kits intended for assembly by others;

which are based in whole or in part on the Documentation.

1.4 This Agreement applies to any Documentation which contains a notice stating it is subject to the TAPR Open Hardware License, and to all Products based in whole or in part on that Documentation. If Documentation is distributed in an archive (such as a “zip” file) which includes this document, all files in that archive are subject to this Agreement unless they are specifically excluded. Each person who contributes content to the Documentation is referred to in this Agreement as a “Licensor.”

1.5 By (a) using, copying, modifying, or distributing the Documentation, or (b) making or having Products made or distributing them, you accept this Agreement, agree to comply with its terms, and become a “Licensee.” Any activity inconsistent with this Agreement will automatically terminate your rights under it (including the immunities from suit granted in Section 2), but the rights of others who have received Documentation, or have obtained Products, directly or indirectly from you will not be affected so long as they fully comply with it themselves.

1.6 This Agreement does not apply to software, firmware, or code loaded into programmable devices which may be used in conjunction with Documentation or Products. Such software is subject to the license terms established by its copyright holder(s).

**2. Patents**

2.1 Each Licensor grants you, every other Licensee, and every possessor or user of Products a perpetual, worldwide, and royalty-free immunity from suit under any patent, patent application, or other intellectual property right which he or she controls, to the extent necessary to make, have made, possess, use, and distribute Products. This immunity does not extend to infringement arising from modifications subsequently made by others.

2.2 If you make or have Products made, or distribute Documentation that you have modified, you grant every Licensor, every other Licensee, and every possessor or user of Products a perpetual, worldwide, and royalty-free immunity from suit under any patent, patent application, or other intellectual property right which you control, to the extent necessary to make, have made, possess, use, and distribute Products. This immunity does not extend to infringement arising from modifications subsequently made by others.

2.3 To avoid doubt, providing Documentation to a third party for the sole purpose of having that party make Products on your behalf is not considered “distribution,” and a third party's act of making Products solely on your behalf does not cause that party to grant the immunity described in the preceding paragraph.

2.4 These grants of immunity are a material part of this Agreement, and form a portion of the consideration given by each party to the other. If any court judgment or legal agreement prevents you

from granting the immunity required by this Section, your rights under this Agreement will terminate and you may no longer use, copy, modify or distribute the Documentation, or make, have made, or distribute Products.

**3. Modifications**

You may modify the Documentation, and those modifications will become part of the Documentation. They are subject to this Agreement, as are Products based in whole or in part on them. If you distribute the modified Documentation, or Products based in whole or in part upon it, you must email the modified Documentation in a form compliant with Section 4 to each Licensor who has provided an email address with the Documentation. Attempting to send the email completes your obligations under this Section and you need take no further action if any address fails.

**4. Distributing Documentation**

4.1 You may distribute unmodified copies of the Documentation in its entirety in any medium, provided that you retain all copyright and other notices (including references to this Agreement) included by each Licensor, and include an unaltered copy of this Agreement.

4.2 You may distribute modified copies of the Documentation if you comply with all the requirements of the preceding paragraph and:

(a) include a prominent notice in an ASCII or other open format file identifying those elements of the Documentation that you changed, and stating that the modifications are licensed under the terms of this Agreement;

(b) include all new documentation files that you create, as well as both the original and modified versions of each file you change (files may be in your development tool's native file format, but if reasonably possible, you must also include open format, such as Gerber, ASCII, Postscript, or PDF, versions);

(c) do not change the terms of this Agreement with respect to subsequent licensees; and

(d) if you make or have Products made, include in the Documentation all elements reasonably required to permit others to make Products, including Gerber, CAD/CAM and other files used for manufacture.

**5. Making Products**

5.1 You may use the Documentation to make or have Products made, provided that each Product retains any notices included by the Licensor (including, but not limited to, copyright notices on circuit boards).

5.2 You may distribute Products you make or have made, provided that you include with each unit a copy of the Documentation in a form consistent with Section 4. Alternatively, you may include either (i) an offer valid for at least three years to provide that Documentation, at no charge other than the reasonable cost of media and postage, to any person who requests it; or (ii) a URL where that Documentation may be downloaded, available for at least three years after you last distribute the Product.

**(i) 6. NEW LICENSE VERSIONS**

TAPR may publish updated versions of the OHL which retain the same general provisions as the present version, but differ in detail to address new problems or concerns, and carry a distinguishing version number. If the Documentation specifies a version number which applies to it and “any later version”, you may choose either that version or any later version published by TAPR. If the Documentation does not specify a version number, you may choose any version ever published by TAPR. TAPR owns the copyright to the OHL, but grants permission to any person to copy, distribute, and use it in unmodified form.

**7. WARRANTY AND LIABILITY LIMITATIONS**

7.1 THE DOCUMENTATION IS PROVIDED ON AN “AS-IS” BASIS WITHOUT WARRANTY OF ANY KIND, TO THE EXTENT PERMITTED BY APPLICABLE LAW. ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND TITLE, ARE HEREBY EXPRESSLY DISCLAIMED.

7.2 IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW WILL ANY LICENSOR BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, PUNITIVE, OR EXEMPLARY DAMAGES ARISING OUT OF THE USE OF, OR INABILITY TO USE, THE DOCUMENTATION OR PRODUCTS, INCLUDING BUT NOT LIMITED TO CLAIMS OF INTELLECTUAL PROPERTY INFRINGEMENT OR LOSS OF DATA, EVEN IF THAT PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7.3 You agree that the foregoing limitations are reasonable due to the non-financial nature of the transaction represented by this Agreement, and acknowledge that were it not for these limitations, the Licensor(s) would not be willing to make the Documentation available to you.

7.4 You agree to defend, indemnify, and hold each Licensor harmless from any claim brought by a third party alleging any defect in the design, manufacture, or operation of any Product which you make, have made, or distribute pursuant to this Agreement.

## 7.3. Appendix 3 – Survey Questionnaire

Questions have been sub-grouped to categories.

**General Issues**

1. How would you rate your contribution to the Arduino community? “0” represents extremely passive contribution (e.g. only reading), “100” extremely active contribution (e.g. contributing often, actively).
2. Why do you participate in open-source projects? Check all that apply.
    a. Developing is fun.
    b. It is a noble cause.
    c. I can create/change/extend the hardware to fit my specific needs.
    d. Expect to sell products or services related to it.
    e. Helps me improve my development skills.
    f. I can use it as an outlet for my creativity.
    g. Build a network of peers.
    h. I am paid to do this job.
    i. Other.
3. *I spend most of my programming time as a:*
    a. Salaried developer.
    b. Contract developer.
    c. Hobby developer.
    d. Student.
    e. Other.
4. Do you receive direct compensation (e.g., salary, contract) for your participation in the project?
    a. Yes.
    b. No.

**Intrinsic Motivation**

***Enjoyment-based***

1. Developing hardware is fun. (strongly agree/strongly disagree)
2. I enjoy developing hardware. (strongly agree/strongly disagree)
3. Developing hardware gives me a chance to do the jobs I feel I do the best. (strongly agree/strongly disagree)

4. Participating in the project gives me a feeling of accomplishment. (strongly agree/strongly disagree)
5. Participating in the project gives me a feeling of competence. (strongly agree/strongly disagree)
6. Participating in the project gives me a feeling of effectiveness. (strongly agree/strongly disagree)
7. Participating in the project is intellectually stimulating. (strongly agree/strongly disagree)
8. I rate my participation as an important activity for myself. (strongly agree/strongly disagree)

***Community-based***

1. I don't care about money. (strongly agree/strongly disagree)
2. You can always trust an open-source developer. (strongly agree strongly disagree)
3. Recognition from others is my greatest reward. (strongly agree/strongly disagree)
4. Open-source developers should help each other out. (strongly agree/strongly disagree)
5. I deeply enjoy helping others—even if I have to make sacrifices. (strongly agree/strongly disagree)
6. Open-source developers are a big family. (strongly agree/strongly disagree)
7. I am proud to be part of the open-source community. (strongly agree/strongly disagree)

**Extrinsic Motivation**

***Direct Rewards***

1. I am paid to work for the project. (strongly agree/strongly disagree)
2. I receive some form of explicit compensation (e.g., salary, contract) for participating in the project. (strongly agree/strongly disagree)
3. For me, working for the project is: (extremely profitable/not profitable)
4. Comparing to other hardware development jobs, working for the project is: (very well paid/very poorly paid)

### *Personal Needs*

1. How often do you use the hardware for yourself (excluding development or testing activities)? (always/never)
2. The hardware is critical for my business or my work: (strongly agree/strongly disagree)
3. My participation in the open-source project ensures that the hardware provides functionality that matches my unique and specific needs. (strongly agree/strongly disagree)
4. It is hard for commercial hardware to meet my ever-changing needs. (strongly agree/strongly disagree)
5. Being able to fix problems with the hardware myself is one of the great advantages of open-source hardware. (strongly agree/strongly disagree)
6. Members of the community are valuable in fixing problems that arise when developing. (strongly agree/strongly disagree)

### *Future Returns*

1. Experience from the project raises my skill level of developing. (strongly agree/strongly disagree)
2. Because of my involvement in the project, I will be able to get a better job. (strongly agree/strongly disagree)
3. In one way or another I will make money from my participation in the project. (strongly agree/strongly disagree)
4. Participating in the project makes me more marketable. (strongly agree/strongly disagree)
5. I will sell products related to the project. (strongly agree/strongly disagree)
6. I will sell consulting, training, implementation or customization services related to the project. (strongly agree/strongly disagree)

## Additional Issues

### *Creativity*

1. When I work on a project, it's just like composing poetry or music. (strongly agree/strongly disagree)

2. A finished project is like a piece of art.

(strongly agree/strongly disagree)

3. Building objects makes open-source hardware development especially enjoyable. (strongly agree/strongly disagree)
4. This project compared to my most creative experience is

(my most creative effort/equally as creative/somewhat less creative/much less creative).

5. Compared to merely programming software, hardware development is

(my most enjoyable effort/equally as enjoyable/somewhat less enjoyable/much less enjoyable).

***Effort and Cost Level***

1. Actually, how often do you work for the project? (more than once a day/not at all)
2. Actually, how many hours a week do you spend in the project? (0 / 1 to 5 / 6 to 12 / 13 to 20 / 21 to 40 / more than 40)
3. How many projects are you currently involved in? (0/1/2/3/4+ projects)
4. What is your most important cost for participating in this project? (order from most important to least important).
    a. Social time
    b. Sleep
    c. Hardware parts
    d. Software/bandwidth etc.
    e. Time to make money
    f. Academic performance
    g. Stress/health
    h. Social relationship(s)
    i. Professional/career advancement
    j. Other
5. Usually, what do you do you do with a finished Arduino project? Check all that apply.
    a. I share the source code and circuit plans with the community freely.
    b. I keep the project to myself.

c. I sell my project to companies.
d. I sell my idea to private persons.
e. I share videos and/or pictures of the project with my peers.
f. I have my circuit boards printed.
g. I deconstruct my project and start building a new project.
h. I ask for feedback from the community to further improve the project.
i. I try to commercialize my invention in some way.
j. Other, please indicate:

***Others***

1. Gender
2. Year of Birth
3. Country
4. Highest educational degree
    a. Primary school
    b. High school
    c. Professional degree
    d. Bachelor's degree
    e. Master's degree
    f. Doctoral degree
    g. Other
5. Programming experience
6. Hardware development experience
7. Year of start in open-source hardware community
8. Occupation
    a. Programmer
    b. Sys. Admin.
    c. IT Manager
    d. Student
    e. Academic
    f. Other, please indicate.
9. Please indicate your GitHub/Arduino Forum/etc. user name

## 7.4. Appendix 4 – Output of the Independent-Samples T-Test

In the output, “1” denotes the active user state, whereas “.00” indicates the passive user state, as per the analysis described.

# T-Test: posts_number

**Group Statistics**

| | hl_posts_number | N | Mean | Std. Deviation | Std. Error Mean |
|---|---|---|---|---|---|
| ENJOYMENT | 1.00 | 41 | 6.0070 | .63804 | .09964 |
| | .00 | 39 | 6.0256 | .68345 | .10944 |
| COMMUNITY | 1.00 | 41 | 5.3780 | .79147 | .12361 |
| | .00 | 39 | 5.5470 | .85579 | .13704 |
| PAY | 1.00 | 41 | 2.4390 | 1.14671 | .17909 |
| | .00 | 39 | 2.6218 | 1.47554 | .23628 |
| NEEDS | 1.00 | 41 | 4.9073 | 1.00807 | .15743 |
| | .00 | 39 | 5.0718 | .82906 | .13276 |
| FUTURE | 1.00 | 41 | 4.1748 | 1.16902 | .18257 |
| | .00 | 39 | 4.1838 | 1.28179 | .20525 |
| CREATIVITY | 1.00 | 41 | 5.6260 | 1.04933 | .16388 |
| | .00 | 39 | 5.8291 | 1.05927 | .16962 |

**Independent Samples Test**

| | | Levene's Test for Equality of Variances | | t-test for Equality of Means | | | |
|---|---|---|---|---|---|---|---|
| | | F | Sig. | t | df | Sig. (2-tailed) | Mean Difference |
| ENJOYMENT | Equal variances assumed | .377 | .541 | -.126 | 78 | .900 | -.01867 |
| | Equal variances not assumed | | | -.126 | 76.908 | .900 | -.01867 |
| COMMUNITY | Equal variances assumed | .035 | .853 | -.917 | 78 | .362 | -.16896 |
| | Equal variances not assumed | | | -.916 | 76.734 | .363 | -.16896 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| PAY | Equal variances assumed | .590 | .445 | -.620 | 78 | .537 | -.18277 |
| | Equal variances not assumed | | | -.616 | 71.717 | .540 | -.18277 |
| NEEDS | Equal variances assumed | .767 | .384 | -.795 | 78 | .429 | -.16448 |
| | Equal variances not assumed | | | -.799 | 76.431 | .427 | -.16448 |
| FUTURE | Equal variances assumed | .173 | .679 | -.033 | 78 | .974 | -.00896 |
| | Equal variances not assumed | | | -.033 | 76.453 | .974 | -.00896 |
| CREATIVITY | Equal variances assumed | .000 | .985 | -.861 | 78 | .392 | -.20304 |
| | Equal variances not assumed | | | -.861 | 77.719 | .392 | -.20304 |

**Independent Samples Test**

| | | t-test for Equality of Means | | |
|---|---|---|---|---|
| | | Std. Error Difference | 95% Confidence Interval of the Difference | |
| | | | Lower | Upper |
| ENJOYMENT | Equal variances assumed | .14775 | -.31282 | .27548 |
| | Equal variances not assumed | .14801 | -.31340 | .27605 |
| COMMUNITY | Equal variances assumed | .18418 | -.53564 | .19772 |
| | Equal variances not assumed | .18455 | -.53646 | .19854 |
| PAY | Equal variances assumed | .29463 | -.76933 | .40379 |

| | | | | |
|---|---|---|---|---|
| | Equal variances not assumed | .29648 | -.77382 | .40828 |
| NEEDS | Equal variances assumed | .20694 | -.57647 | .24752 |
| | Equal variances not assumed | .20594 | -.57460 | .24564 |
| FUTURE | Equal variances assumed | .27406 | -.55458 | .53665 |
| | Equal variances not assumed | .27470 | -.55602 | .53809 |
| CREATIVITY | Equal variances assumed | .23580 | -.67248 | .26639 |
| | Equal variances not assumed | .23585 | -.67262 | .26653 |

## T-Test: days_registered

**Group Statistics**

| | hl_days_registered | N | Mean | Std. Deviation | Std. Error Mean |
|---|---|---|---|---|---|
| ENJOYMENT | 1.00 | 40 | 6.0179 | .61907 | .09788 |
| | .00 | 40 | 6.0143 | .69970 | .11063 |
| COMMUNITY | 1.00 | 40 | 5.2792 | .82809 | .13093 |
| | .00 | 40 | 5.6417 | .78587 | .12426 |
| PAY | 1.00 | 40 | 2.4063 | 1.34175 | .21215 |
| | .00 | 40 | 2.6500 | 1.28702 | .20350 |
| NEEDS | 1.00 | 40 | 4.8200 | .90105 | .14247 |
| | .00 | 40 | 5.1550 | .92541 | .14632 |
| FUTURE | 1.00 | 40 | 4.1417 | 1.28577 | .20330 |
| | .00 | 40 | 4.2167 | 1.16036 | .18347 |
| CREATIVITY | 1.00 | 40 | 5.7500 | 1.03706 | .16397 |
| | .00 | 40 | 5.7000 | 1.08026 | .17080 |

**Independent Samples Test**

| | | Levene's Test for Equality of Variances | | t-test for Equality of Means | | | |
|---|---|---|---|---|---|---|---|
| | | F | Sig. | t | df | Sig. (2-tailed) | Mean Difference |
| ENJOYMENT | Equal variances assumed | 1.046 | .310 | .024 | 78 | .981 | .00357 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Equal variances not assumed | | | .024 | 76.859 | .981 | .00357 |
| COMMUNITY | Equal variances assumed | .218 | .642 | -2.008 | 78 | .048 | -.36250 |
| | Equal variances not assumed | | | -2.008 | 77.787 | .048 | -.36250 |
| PAY | Equal variances assumed | 1.208 | .275 | -.829 | 78 | .410 | -.24375 |
| | Equal variances not assumed | | | -.829 | 77.865 | .410 | -.24375 |
| NEEDS | Equal variances assumed | .000 | .995 | -1.640 | 78 | .105 | -.33500 |
| | Equal variances not assumed | | | -1.640 | 77.945 | .105 | -.33500 |
| FUTURE | Equal variances assumed | 1.207 | .275 | -.274 | 78 | .785 | -.07500 |
| | Equal variances not assumed | | | -.274 | 77.193 | .785 | -.07500 |
| CREATIVITY | Equal variances assumed | .914 | .342 | .211 | 78 | .833 | .05000 |
| | Equal variances not assumed | | | .211 | 77.870 | .833 | .05000 |

**Independent Samples Test**

| | | t-test for Equality of Means | | |
|---|---|---|---|---|
| | | Std. Error Difference | 95% Confidence Interval of the Difference | |
| | | | Lower | Upper |
| ENJOYMENT | Equal variances assumed | .14772 | -.29051 | .29766 |

| | | | | |
|---|---|---|---|---|
| | Equal variances not assumed | .14772 | -.29058 | .29772 |
| COMMUNITY | Equal variances assumed | .18051 | -.72186 | -.00314 |
| | Equal variances not assumed | .18051 | -.72188 | -.00312 |
| PAY | Equal variances assumed | .29397 | -.82900 | .34150 |
| | Equal variances not assumed | .29397 | -.82901 | .34151 |
| NEEDS | Equal variances assumed | .20422 | -.74158 | .07158 |
| | Equal variances not assumed | .20422 | -.74158 | .07158 |
| FUTURE | Equal variances assumed | .27385 | -.62018 | .47018 |
| | Equal variances not assumed | .27385 | -.62027 | .47027 |
| CREATIVITY | Equal variances assumed | .23677 | -.42138 | .52138 |
| | Equal variances not assumed | .23677 | -.42139 | .52139 |

## T-Test: karma

**Group Statistics**

| | hl_karma | N | Mean | Std. Deviation | Std. Error Mean |
|---|---|---|---|---|---|
| ENJOYMENT | 1.00 | 26 | 5.9835 | .58739 | .11520 |
| | .00 | 54 | 6.0317 | .69192 | .09416 |
| COMMUNITY | 1.00 | 26 | 5.2244 | .83525 | .16381 |
| | .00 | 54 | 5.5741 | .79941 | .10879 |
| PAY | 1.00 | 26 | 2.3269 | 1.26643 | .24837 |
| | .00 | 54 | 2.6250 | 1.33419 | .18156 |
| NEEDS | 1.00 | 26 | 4.9385 | .93084 | .18255 |
| | .00 | 54 | 5.0111 | .92709 | .12616 |
| FUTURE | 1.00 | 26 | 4.1859 | 1.23408 | .24202 |
| | .00 | 54 | 4.1759 | 1.22107 | .16617 |
| CREATIVITY | 1.00 | 26 | 5.6667 | .96609 | .18947 |
| | .00 | 54 | 5.7531 | 1.09924 | .14959 |

**Independent Samples Test**

| | | Levene's Test for Equality of Variances | | t-test for Equality of Means | | | |
|---|---|---|---|---|---|---|---|
| | | F | Sig. | t | df | Sig. (2-tailed) | Mean Difference |
| ENJOYMENT | Equal variances assumed | 2.108 | .151 | -.306 | 78 | .760 | -.04823 |
| | Equal variances not assumed | | | -.324 | 57.465 | .747 | -.04823 |
| COMMUNITY | Equal variances assumed | .105 | .747 | -1.806 | 78 | .075 | -.34972 |
| | Equal variances not assumed | | | -1.778 | 47.552 | .082 | -.34972 |
| PAY | Equal variances assumed | .052 | .821 | -.951 | 78 | .344 | -.29808 |
| | Equal variances not assumed | | | -.969 | 51.871 | .337 | -.29808 |
| NEEDS | Equal variances assumed | .057 | .812 | -.328 | 78 | .744 | -.07265 |
| | Equal variances not assumed | | | -.327 | 49.281 | .745 | -.07265 |
| FUTURE | Equal variances assumed | .088 | .768 | .034 | 78 | .973 | .00997 |
| | Equal variances not assumed | | | .034 | 48.989 | .973 | .00997 |
| CREATIVITY | Equal variances assumed | .740 | .392 | -.342 | 78 | .733 | -.08642 |
| | Equal variances not assumed | | | -.358 | 55.677 | .722 | -.08642 |

**Independent Samples Test**

| | | t-test for Equality of Means | | |
|---|---|---|---|---|
| | | Std. Error Difference | 95% Confidence Interval of the Difference | |
| | | | Lower | Upper |
| ENJOYMENT | Equal variances assumed | .15760 | -.36198 | .26552 |
| | Equal variances not assumed | .14878 | -.34611 | .24965 |
| COMMUNITY | Equal variances assumed | .19361 | -.73516 | .03573 |
| | Equal variances not assumed | .19664 | -.74518 | .04575 |
| PAY | Equal variances assumed | .31339 | -.92198 | .32582 |
| | Equal variances not assumed | .30765 | -.91547 | .31931 |
| NEEDS | Equal variances assumed | .22159 | -.51380 | .36850 |
| | Equal variances not assumed | .22191 | -.51852 | .37322 |
| FUTURE | Equal variances assumed | .29247 | -.57230 | .59224 |
| | Equal variances not assumed | .29357 | -.57999 | .59994 |
| CREATIVITY | Equal variances assumed | .25264 | -.58939 | .41655 |
| | Equal variances not assumed | .24140 | -.57006 | .39722 |

## T-Test: spent_online

**Group Statistics**

| | hl_spent_online | N | Mean | Std. Deviation | Std. Error Mean |
|---|---|---|---|---|---|
| ENJOYMENT | 1.00 | 40 | 5.9071 | .60829 | .09618 |
| | .00 | 40 | 6.1250 | .69172 | .10937 |
| COMMUNITY | 1.00 | 40 | 5.2958 | .84098 | .13297 |
| | .00 | 40 | 5.6250 | .77968 | .12328 |
| PAY | 1.00 | 40 | 2.5250 | 1.32142 | .20894 |
| | .00 | 40 | 2.5313 | 1.31946 | .20863 |
| NEEDS | 1.00 | 40 | 4.8450 | .97821 | .15467 |

| | | | | | |
|---|---|---|---|---|---|
| | .00 | 40 | 5.1300 | .85281 | .13484 |
| FUTURE | 1.00 | 40 | 4.1500 | 1.19221 | .18851 |
| | .00 | 40 | 4.2083 | 1.25675 | .19871 |
| CREATIVITY | 1.00 | 40 | 5.6333 | .97490 | .15415 |
| | .00 | 40 | 5.8167 | 1.12963 | .17861 |

**Independent Samples Test**

| | | Levene's Test for Equality of Variances | | t-test for Equality of Means | | | |
|---|---|---|---|---|---|---|---|
| | | F | Sig. | t | df | Sig. (2-tailed) | Mean Difference |
| ENJOYMENT | Equal variances assumed | 1.108 | .296 | -1.496 | 78 | .139 | -.21786 |
| | Equal variances not assumed | | | -1.496 | 76.746 | .139 | -.21786 |
| COMMUNITY | Equal variances assumed | .410 | .524 | -1.815 | 78 | .073 | -.32917 |
| | Equal variances not assumed | | | -1.815 | 77.557 | .073 | -.32917 |
| PAY | Equal variances assumed | .181 | .672 | -.021 | 78 | .983 | -.00625 |
| | Equal variances not assumed | | | -.021 | 78.000 | .983 | -.00625 |
| NEEDS | Equal variances assumed | .382 | .539 | -1.389 | 78 | .169 | -.28500 |
| | Equal variances not assumed | | | -1.389 | 76.577 | .169 | -.28500 |
| FUTURE | Equal variances assumed | .034 | .853 | -.213 | 78 | .832 | -.05833 |
| | Equal variances not assumed | | | -.213 | 77.784 | .832 | -.05833 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| CREATIVITY | Equal variances assumed | 1.463 | .230 | -.777 | 78 | .439 | -.18333 |
| | Equal variances not assumed | | | -.777 | 76.366 | .440 | -.18333 |

**Independent Samples Test**

| | | t-test for Equality of Means | | |
|---|---|---|---|---|
| | | Std. Error Difference | 95% Confidence Interval of the Difference | |
| | | | Lower | Upper |
| ENJOYMENT | Equal variances assumed | .14564 | -.50781 | .07210 |
| | Equal variances not assumed | .14564 | -.50789 | .07217 |
| COMMUNITY | Equal variances assumed | .18133 | -.69016 | .03182 |
| | Equal variances not assumed | .18133 | -.69019 | .03186 |
| PAY | Equal variances assumed | .29526 | -.59407 | .58157 |
| | Equal variances not assumed | .29526 | -.59407 | .58157 |
| NEEDS | Equal variances assumed | .20519 | -.69351 | .12351 |
| | Equal variances not assumed | .20519 | -.69363 | .12363 |
| FUTURE | Equal variances assumed | .27390 | -.60362 | .48695 |
| | Equal variances not assumed | .27390 | -.60364 | .48698 |
| CREATIVITY | Equal variances assumed | .23593 | -.65303 | .28636 |
| | Equal variances not assumed | .23593 | -.65319 | .28652 |

## T-Test: topics_started

**Group Statistics**

| hl_topics_started | N | Mean | Std. Deviation | Std. Error Mean |
|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| ENJOYMENT | 1.00 | 47 | 6.0213 | .61335 | .08947 |
| | .00 | 33 | 6.0087 | .72311 | .12588 |
| COMMUNITY | 1.00 | 47 | 5.4574 | .79587 | .11609 |
| | .00 | 33 | 5.4646 | .87178 | .15176 |
| PAY | 1.00 | 47 | 2.6436 | 1.29563 | .18899 |
| | .00 | 33 | 2.3636 | 1.33769 | .23286 |
| NEEDS | 1.00 | 47 | 5.1277 | .92353 | .13471 |
| | .00 | 33 | 4.7879 | .89853 | .15641 |
| FUTURE | 1.00 | 47 | 4.2518 | 1.16453 | .16986 |
| | .00 | 33 | 4.0758 | 1.30043 | .22638 |
| CREATIVITY | 1.00 | 47 | 5.6312 | 1.02206 | .14908 |
| | .00 | 33 | 5.8586 | 1.09617 | .19082 |

**Independent Samples Test**

| | | Levene's Test for Equality of Variances | | t-test for Equality of Means | | | |
|---|---|---|---|---|---|---|---|
| | | F | Sig. | t | df | Sig. (2-tailed) | Mean Difference |
| ENJOYMENT | Equal variances assumed | .794 | .376 | .084 | 78 | .933 | .01262 |
| | Equal variances not assumed | | | .082 | 61.567 | .935 | .01262 |
| COMMUNITY | Equal variances assumed | .069 | .794 | -.038 | 78 | .970 | -.00720 |
| | Equal variances not assumed | | | -.038 | 64.940 | .970 | -.00720 |
| PAY | Equal variances assumed | .013 | .910 | .939 | 78 | .351 | .27998 |
| | Equal variances not assumed | | | .934 | 67.627 | .354 | .27998 |
| NEEDS | Equal variances assumed | .002 | .969 | 1.638 | 78 | .105 | .33978 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Equal variances not assumed | | | 1.646 | 70.207 | .104 | .33978 |
| FUTURE | Equal variances assumed | .454 | .503 | .634 | 78 | .528 | .17602 |
| | Equal variances not assumed | | | .622 | 64.053 | .536 | .17602 |
| CREATIVITY | Equal variances assumed | .028 | .868 | -.951 | 78 | .345 | -.22738 |
| | Equal variances not assumed | | | -.939 | 65.906 | .351 | -.22738 |

**Independent Samples Test**

| | | t-test for Equality of Means | | |
|---|---|---|---|---|
| | | Std. Error Difference | 95% Confidence Interval of the Difference | |
| | | | Lower | Upper |
| ENJOYMENT | Equal variances assumed | .15003 | -.28606 | .31130 |
| | Equal variances not assumed | .15443 | -.29613 | .32137 |
| COMMUNITY | Equal variances assumed | .18802 | -.38151 | .36711 |
| | Equal variances not assumed | .19107 | -.38880 | .37440 |
| PAY | Equal variances assumed | .29821 | -.31371 | .87367 |
| | Equal variances not assumed | .29990 | -.31852 | .87848 |
| NEEDS | Equal variances assumed | .20743 | -.07319 | .75275 |
| | Equal variances not assumed | .20643 | -.07190 | .75147 |
| FUTURE | Equal variances assumed | .27756 | -.37656 | .72859 |
| | Equal variances not assumed | .28302 | -.38937 | .74140 |

| | | | | |
|---|---|---|---|---|
| CREATIVITY | Equal variances assumed | .23917 | -.70353 | .24877 |
| | Equal variances not assumed | .24215 | -.71087 | .25611 |

# 8. Bibliography


Acosta, R. (2009). *Open source hardware* (Thesis). Massachusetts Institute of Technology. Retrieved from http://dspace.mit.edu/handle/1721.1/55201

Arduino - HomePage. (n.d.). Retrieved from http://arduino.cc

Ashforth, B. E., & Mael, F. (1989). Social identity theory and the organization. *Academy of Management Review*, *14*(1), 20–39.

Bogers, M., & West, J. (2012). Managing distributed innovation: Strategic utilization of open and user innovation. *Creativity and Innovation Management*, *21*(1), 61–75.

Buechley, L., & Eisenberg, M. (2008). The lilypad arduino: Toward wearable engineering for everyone. *Pervasive Computing, IEEE*, *7*(2), 12–15.

Chesbrough, H. W. (2003). *Open innovation: The new imperative for creating and profiting from technology*. Harvard Business Press.

Cinder. (n.d.). Cinder | The library for professional-quality creative coding in C++. Cinder. Retrieved from http://libcinder.org/about/

Constant, D., Sproull, L., & Kiesler, S. (1996). The kindness of strangers: The usefulness of electronic weak ties for technical advice. *Organization Science*, *7*(2), 119–135.

Couper, M. P. (2000). Review: Web surveys: A review of issues and approaches. *Public Opinion Quarterly*, 464–494.

Csikszentmihalyi, M. (2000). *Beyond boredom and anxiety*. Jossey-Bass. Retrieved from http://psycnet.apa.org/psycinfo/2000-12701-000

Davidson, S. (2004). Open-source hardware. *IEEE Design & Test of Computers*, *21*(5), 456–456.

Deci, E. L., & Ryan, R. M. (1987). The support of autonomy and the control of behavior. *Journal of Personality and Social Psychology*, *53*(6), 1024.

Deci, E. L., & Ryan, R. M. (2002). *Handbook of self-determination research*. University Rochester Press. Retrieved from http://books.google.com/books?hl=en&lr=&id=DcAe2b7L-RgC&oi=fnd&pg=PP11&dq=Handbook+of+self-determination+research.&ots=dpDS0IY-1h&sig=HWDUuu-Y91n6hPTdfFLWOd1A09A

Franck, E., & Jungwirth, C. (2003). Reconciling rent-seekers and donators–The governance structure of open source. *Journal of Management and Governance*, *7*(4), 401–421.

Gallivan, M. J. (2001). Striking a balance between trust and control in a virtual organization: a content analysis of open source software case studies. *Information Systems Journal*, *11*(4), 277–304.

Gibb, A. M. (2010). *New media art, design, and the Arduino microcontroller: A malleable tool*. Pratt Institute. Retrieved from http://aliciagibb.com/wp-content/uploads/2013/01/New-Media-Art-Design-and-the-Arduino-Microcontroller-2.pdf

gnu.org. (n.d.). Retrieved May 8, 2014, from https://www.gnu.org/philosophy/free-sw.html

Hargadon, A. B., & Bechky, B. A. (2006). When collections of creatives become creative collectives: A field study of problem solving at work. *Organization Science*, *17*(4), 484–500.

Hars, A., & Ou, S. (2001). Working for free? Motivations of participating in open source projects. In *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on* (p. 9–pp). IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=927045

Hertel, G., Niedner, S., & Herrmann, S. (2003). Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. *Research Policy*, *32*(7), 1159–1177.

Herzberg, F., Mausner, B., & Snyderman, B. B. (2011). *The motivation to work* (Vol. 1). Transaction Publishers. Retrieved from http://books.google.com/books?hl=en&lr=&id=KYhB-B6kfSMC&oi=fnd&pg=PP1&dq=the+motivation+to+work&ots=nh2DUGJDYg&sig=MCDtvkspyEepaA6lwFXaK9-GpQg

Kaiser, H. F. (1960). The application of electronic computers to factor analysis. *Educational and Psychological Measurement*. Retrieved from http://psycnet.apa.org/psycinfo/1960-06772-001

Kline, P. (2013). *Handbook of psychological testing*. Routledge. Retrieved from http://books.google.com/books?hl=en&lr=&id=JggVAgAAQBAJ&oi=fnd&pg=P

P1&dq=The+handbook+of+psychological+testing&ots=KNknJ79tqq&sig=ZtY7plRu4ktUZ3Wo-0TMaUG1ibo

Koch, S., & Schneider, G. (2002). Effort, co-operation and co-ordination in an open source software project: GNOME. *Information Systems Journal*, *12*(1), 27–42. doi:10.1046/j.1365-2575.2002.00110.x

Kogut, B., & Metiu, A. (2001). Open-source software development and distributed innovation. *Oxford Review of Economic Policy*, *17*(2), 248–264.

Lakhani, K. R., & Wolf, R. G. (2003). Why hackers do what they do: Understanding motivation and effort in free/open source software projects. *Perspectives on Free and Open Source Software*, *1*, 3–22.

Lerner, J., & Tirole, J. (2002). Some simple economics of open source. *The Journal of Industrial Economics*, *50*(2), 197–234.

Lindenberg, S. (2001). Intrinsic motivation in a new light. *Kyklos*, *54*(2-3), 317–342.

Lock, J. (2013). Open Source Hardware. Retrieved from http://publications.lib.chalmers.se/records/fulltext/182348/182348.pdf

Maslow, A. H. (1943). A theory of human motivation. *Psychological Review*, *50*(4), 370–396. doi:10.1037/h0054346

Noble, J. (2009). *Programming interactivity*. O'Reilly Media, Inc. Retrieved from http://books.google.com/books?hl=en&lr=&id=STXR9vCXs5YC&oi=fnd&pg=PR3&dq=Programming+Interactivity&ots=kXVQGRJ479&sig=CYoyQqau7LFvM0fsZJvGPTdPqnk

Nonnecke, B., & Preece, J. (2000). Lurker demographics: Counting the silent. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 73–80). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=332409

Occupational Employment and Wages, May 2013, 27-0000 Arts, Design, Entertainment, Sports, and Media Occupations (Major Group). (2014, April 1). US Bureau of Labor Statistics. Retrieved from http://www.bls.gov/oes/current/oes270000.htm

Open source hardware. (2014, May 7). In *Wikipedia, the free encyclopedia*. Retrieved from http://en.wikipedia.org/w/index.php?title=Open_source_hardware&oldid=604986338

Oreg, S., & Nov, O. (2008). Exploring motivations for contributing to open source initiatives: The roles of contribution context and personal values. *Computers in Human Behavior*, *24*(5), 2055–2073.

Prahalad, C. K., & Ramaswamy, V. (2004). Co-creation experiences: The next practice in value creation. *Journal of Interactive Marketing*, *18*(3), 5–14.

Prandelli, E., & Verona, G. (2012). From Software to Hardware: The Changing Technology of Open Source. In *Advances in Strategy and Organization. Selected Papers from "Second Tuesday" Seminars*. Milano: McGraw Hill.

Prandelli, E., Verona, G., & Raccagni, D. (2006). Diffusion of Web-Based Product Innovation. *California Management Review*, *48*(4).

Raymond, E. S. (2001). *The Cathedral & the Bazaar: Musings on linux and open source by an accidental revolutionary*. O'Reilly. Retrieved from http://books.google.com/books?hl=en&lr=&id=F6qgFtLwpJgC&oi=fnd&pg=PR7&dq=open+source+bazaar&ots=k3prFDxOVb&sig=lYPUDLjVXLSavpRPCLHRBiXhsls

Roberts, J. A., Hann, I.-H., & Slaughter, S. A. (2006). Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the Apache projects. *Management Science*, *52*(7), 984–999.

Rubow, E. (2008). *Open source hardware*. Technical report. Retrieved from http://cseweb.ucsd.edu/classes/fa08/cse237a/topicresearch/erubow_tr_report.pdf

Ryan, R. M., & Deci, E. L. (2000). Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary Educational Psychology*, *25*(1), 54–67.

Sawhney, M., Verona, G., & Prandelli, E. (2005). Collaborating to create: The Internet as a platform for customer engagement in product innovation. *Journal of Interactive Marketing*, *19*(4), 4–17.

Shah, S. K. (2006). Motivation, governance, and the viability of hybrid forms in open source software development. *Management Science*, *52*(7), 1000–1014.

Shah, S. K., & Tripsas, M. (2007). The accidental entrepreneur: The emergent and collective process of user entrepreneurship. *Strategic Entrepreneurship Journal*, *1*(1-2), 123–140.

Steven, L. (1984). *Hackers: Heroes of the computer revolution*. Doubleday, New York.

Swan, M. (2012). Sensor mania! The Internet of Things, wearable computing, objective metrics, and the Quantified Self 2.0. *Journal of Sensor and Actuator Networks*, *1*(3), 217–253.

Thompson, C. (2008). Build it. share it. profit. Can open source hardware work? *Wired Magazine*, *16*(11), 16–11.

Von Hippel, E., & Von Krogh, G. (2003). Open source software and the "private-collective" innovation model: Issues for organization science. *Organization Science*, *14*(2), 209–223.

Von Krogh, G., Spaeth, S., & Lakhani, K. R. (2003). Community, joining, and specialization in open source software innovation: a case study. *Research Policy*, *32*(7), 1217–1241.

Von Krogh, G., & Von Hippel, E. (2006). The promise of research on open source software. *Management Science*, *52*(7), 975–983.

West, J., & Dedrick, J. (2001). Open source standardization: the rise of Linux in the network era. *Knowledge, Technology & Policy*, *14*(2), 88–112.

White, R. W. (1959). Motivation reconsidered: The concept of competence. *Psychological Review*, *66*(5), 297–333. doi:10.1037/h0040934