H-node.org – Hardware Database for Free Software Community

Analysis and Development

Matti Juhani Lammi

Thesis of Information Technology Degree Programme
International Software Development
Bachelor of Engineering

KEMI 2013

PREFACE

I would like to thank *Tapani Ruokanen* for acting as my instructor while I was writing this final project. He gave me many advices and provided valuable guidance. I would like to thank *Pertti Ylimäinen* and everyone else of the staff of *Lapin ATK-Huolto* for giving their support for this project. I would like to thank two Trisquel GNU/Linux forum users *Michał Masłowski* and *Fabián Rodríguez* for proof reading this final project. *Michał Masłowski* also helped me in solving a problem with one of my scripts.

Most notably, I would like to thank *Antonio Gallo* for his invaluable work he has done with the h-node hardware database project and for answering my many questions for this project through e-mail. Without him, there would be no h-node and the Free Software community would be missing one vital pillar to stand on. Lastly, I would like to thank everyone who has contributed to h-node in any way.

This final project is under Creative Commons Attribution-NoDerivs Unported 3.0 (CC BY-ND 3.0) license. Copying, distributing and transmitting this project is permitted worldwide in any medium, with or without royalty, provided no changes are made. The four scripts, which can be found in the appendixes as appendixes 1, 2, 3 and 4, are under GNU GPLv3 license. The content on the included CD or compressed file are under their respective licenses which can be found in the *LICENSES* folder.

14 May 2013
Matti Juhani Lammi

ABSTRACT

KEMI-TORNIO UNIVERSITY OF APPLIED SCIENCES, Technology

Degree programme:	Information Technology
Author:	Matti Juhani Lammi
Thesis title:	H-node.org – Hardware Database for Free Software Community
Pages (of which appendixes):	78 (25)
Date:	14 May 2013
Thesis instructor:	Tapani Ruokanen, M.Sc. (Tech.)

The topic of this final project was a hardware database called h-node for GNU/Linux operating system distributions which include only Free Software. This project aimed to introduce that database to new users and to contribute to its development.

The reader will be introduced to fundamental concepts which form an ideological base for the database. The website of the database will be introduced to the reader and analyzed section by section. Then several ways of contributing to this database project are discussed. Finally, a summary of the author's contributions are displayed and further development of the project is discussed.

This final project was based on a few literal sources but mainly on the author's own experience and deductions. Several interviews of the main developer of the database project are also used.

The objects of this final project were met. The database was extended significantly and this final project was written successfully as an introduction to the database project and the fundamental concepts on its background.

Keywords: computer hardware, databases, free software, linux, open information, websites.

TIIVISTELMÄ

KEMI-TORNION AMMATTIKORKEAKOULU, Tekniikka

Koulutusohjelma:	Tietotekniikan koulutusohjelma
Opinnäytetyön tekijä:	Matti Juhani Lammi
Opinnäytetyön nimi:	H-node.org – Hardware Database for Free Software Community
Sivuja (joista liitesivuja):	78 (25)
Päiväys:	14.5.2013
Opinnäytetyön ohjaaja:	Tapani Ruokanen, DI.

Tämän työn aiheena oli h-node niminen laitetietokanta vain vapaista ohjelmistoista koostuville GNU/Linux käyttöjärjestelmän jakeluversioille. Aihe valittiin, koska tietokanta haluttiin esitellä uusille käyttäjille sekä sen kehitystä haluttiin tukea.

Työssä esiteltiin tietokannan taustalla olevat peruskäsitteet, jotka muodostavat sen ideologisen pohjan. Tietokannan nettisivujen rakenne esiteltiin ja analysoitiin osa osalta. Useita erilaisia tapoja tukea h-node projektia esiteltiin. Työssä esitettiin myös tiivistelmä työn tekijän omista tukitoimista projektille sekä tulevia kehityskohteita.

Tämä työ perustui muutamaan kirjalliseen lähteeseen, mutta enimmäkseen työn tekijän omiin kokemuksiin ja päätelmiin. Työssä käytettiin myös hyväksi tietokantaprojektin ylläpitäjän haastatteluja.

Tämän työn tavoitteet saavutettiin. Tietokantaa laajennettiin merkittävästi ja tämä työ kirjoitettiin onnistuneesti johdannoksi h-node-tietokantaprojektiin ja sen taustalla oleviin peruskäsitteisiin.

Asiasanat: atk-laitteistot, avoin tieto, linux, tietokannat, vapaat ohjelmistot, WWW-sivut.

TABLE OF CONTENTS

PREFACE 2
ABSTRACT 3
TIIVISTELMÄ 4
TABLE OF CONTENTS 5
EXPLANATION OF CHARACTERS AND ABBREVIATIONS 7
1 INTRODUCTION 8
2 BACKGROUND 10
2.1 Free Software 10
2.2 GNU, Linux and GNU/Linux 12
2.3 Hardware compatibility in GNU/Linux 14
2.4 Problem of proprietary drivers and firmware 15
2.5 Fully free GNU/Linux distributions 16
2.6 Existing GNU/Linux hardware databases 17
2.7 H-node 18
2.7.1 Short history 20
3 ANALYSIS 21
3.1 Page layout 21
3.2 Home 22
3.3 Hardware 23
3.4 Issues 27
3.5 Search 28
3.6 News 29
3.7 Downloads 29
3.8 Help 30
3.9 Wiki 32
3.10 FAQ 33
4 DEVELOPMENT 35
4.1 Expanding the database 35
4.1.1 Using Trisquel GNU/Linux live-CD 36

4.1.2 Comparing one’s hardware with the database 39
4.1.3 Adding missing hardware and updating older entries 42
4.1.4 Example: Testing a laptop for h-node 45
4.1.5 Example: Unusable or otherwise problematic laptop 49
4.2 Expanding the wiki 50
4.3 Discussing with other users 51
4.4 Solving and reporting issues 53
4.5 Translating h-node to other languages 54
4.6 Acting as moderator and administrator 56
4.7 Developing source code 58
5 RESULTS OF DEVELOPMENT 60
5.1 Expanding the database 60
5.2 Figuring out distribution of laptop’s compatibility levels 64
5.3 Actions as administrator and moderator 66
5.4 Writing the FAQ 66
5.5 Issues 67
6 FURTHER DEVELOPMENT 68
6.1 Automation 68
6.2 H-client 70
6.3 Documentation 71
6.4 Statistics 72
7 CONCLUSIONS 75
8 REFERENCES 77

EXPLANATION OF CHARACTERS AND ABBREVIATIONS

BIOS	Basic Input Output System, the first program run by the computer when it is powered on
FSF	Free Software Foundation, a non-profit organization formed to increase the development and recognizability of Free Software
GNU GPL	GNU General Public License, a software license used by many Free Software developers
RAM	Random Access Memory, a type of memory technology used in computers to access temporarily stored data quickly
UEFI	Unified Extensible Firmware Interface, a program on newer computers meant to replace the BIOS
XML	Extensible Markup Language, a language used by programmers that is both human-readable and machine readable

1 INTRODUCTION

The author of this final project started to use GNU/Linux operating systems in summer 2008. He was amazed by the power and flexibility of these systems and the vast and friendly community formed around them. Soon, he started to study the history of GNU/Linux. The author found himself reading about ethical computing, hacker culture and the Free Software movement.

Now in 2013, he has become a full time GNU/Linux user and a devoted Free Software supporter. After learning a lot about computing and ethics from the community, he felt it was time to give something back.

A database about Free Software compatible hardware called h-node was picked as a topic for this project because the author considers hardware compatibility to be important for the community. Not much has been written about this subject, so the community could use a small contribution on this field.

Since h-node is a hardware database accessible through the Internet, h-node.org website which acts as an interface for the database will be introduced the reader. Different possibilities to participate in developing the website and expanding the database will then be inspected and presented to the reader.

The shortage of written material regarding hardware compatibility with Free Software limits this project in a sense that the author has to deduce some of his points from written material which does not discuss the matter directly and from interviews of the h-node’s main developer. Some of the points are given as the author’s personal opinions.

The main objective for this final project is to introduce h-node to people who have never heard of it before and thus bring new users and contributors to it. For this to succeed, this final project is aimed to be easy to read and understand and become as widely available as possible. The other objective is to help developing h-node.org website and expand the hardware database.

There is a separate version of h-node.org designed to be used with mobile devices. It will be ruled out of the scope of this project. How to make a GNU/Linux live-CD will not be presented either. There are plenty of instructions on how to make one on the Internet. A live-CD is the most convenient method for both inspecting details about one’s hardware and reporting them to h-node.

The reader will be guided on how to add devices to h-node in one part of this project. This requires light usage of a text-based command line interpreter. It is not necessary to know how to use one to understand the process or to replicate it without too much effort. Although, having some basic skills makes the task easier. It is assumed that those readers who do not know how to use the command line interpreter are willing to learn to do it by themselves. Again, there are many resources about the subject on the Internet. Detailed explanations about how to get all the detailed information about one’s hardware will not be included because long and detailed articles about this matter can be found in the h-node’s wiki page.

2 BACKGROUND

In order to fully understand the significance of h-node and what makes it different compared to other hardware databases, a few basic concepts must be introduced. They form the ideological and practical base which h-node is based on.

2.1 Free Software

Free Software is software that respects its users' rights as computer users instead of restricting them. A program is Free Software if the user of the program is granted the four essential freedoms defined by the FSF, usually by having its source code available to the user and being licensed under a Free Software license.

The most widely used Free Software license is the GNU General Public License or GNU GPL for short. It requires that the user must give the same four freedoms to other users with the program if he publishes an exact copy or a modified version of it.

The four essential freedoms are as follows: (Stallman 2010, 20, 43, 199-204)

- Freedom 0: The freedom to run the program, for any purpose.
- Freedom 1: The freedom to study how the program works, and adapt it to one's needs. Access to the source code is a precondition of this.
- Freedom 2: The freedom to redistribute exact copies, so one can help his or her neighbor.
- Freedom 3: The freedom to redistribute modified version of the program, and release them to the public, so the whole community benefits. Access to the source code is a precondition of this.

Freedom 0 requires that there are no artificial restrictions in the program's source code or its license about what can be done with the program. As it is with physical tools like a hammer, it is up to the user what he or she wants to do with it.

Freedom 1 requires that the user can obtain a copy of the program's source code with the program, so that the user can see what the program is actually doing and change it as he or she sees fit. This makes sure the user is in control of the program and not *vice versa* if he or she has the required skills to put this freedom into action.

Freedom 2 requires that the user must be allowed to make exact copies of the program either for him- or herself as backups or for other people in order to help them. This forms a base for a digital society where people can help each other by sharing important programs like word processors, device drivers and even whole operating systems.

Freedom 3 requires that the user is allowed to share the improvements which he or she made to the program with other people. This is a base for development in a digital society where people share the improvements they make to the programs they use and everyone benefits. This freedom also makes it possible to provide software customization services for profit.

With these freedoms, the user of a program is in full control of it. The user can do whatever he or she sees fit with it. The user can be sure that the original developer of the program cannot have an unjust control over him or her through the program. For example, the developer cannot restrict the program to work only with certain file formats or protocols or make it spy the user and collect private information. Because the source code is available, someone will eventually notice if the developer tries to control the user.

There are programs that do not respect their users. They are proprietary software. They do not give their users all of the four freedoms, if any. Generally, they will not allow the user to study how the program works, see what it is really doing, share it or change it.

They can even have hidden malicious features. Free Software is a way to avoid these problems. (Stallman 2010, 43, 117-119, 135)

If the user of a Free Software program which is licensed under the GNU GPL does not have the necessary skills to change the program, he or she can acquire the skills by studying, ask other users for help or hire a professional programmer to do the changes on his or her behalf. If the user hires a professional, the programmer makes the changes to the program's source code and compiles a new version of the program based on the modified source code. He or she then sells a copy of the program licensed with the same Free Software license to his or her client along with the new source code, so the user still has the same rights as he or she did with the older version of the program. The GNU GPL license gives the user legal rights for all of this. (Stallman 2010, 22-23, 43-44, 199)

Due to the ambiguity of the English word "free", it is easy to misunderstand the meaning of Free Software. The word "free" does not refer to price in any way. It refers to the four essential freedoms. A Free Software program can be sold for a profit or it can be shared without one as gratis. To get rid of the chance of misconception, an alternative term can be used: Libre Software.

2.2 GNU, Linux and GNU/Linux

Richard M. Stallman started the GNU Project in 1983. He's aim was to form a community of computer users who would use, share and develop Free Software programs and an operating system made entirely of Free Software: GNU. The logo of the GNU Project can be seen in Figure 1. (Stallman 1983, retrieved 2.3.2013)

Richard Stallman founded the essential principals of the Free Software philosophy, wrote the GNU GPL license together with legal aid and worked with the GNU Project to develop the GNU system. He also co-founded the Free Software Foundation with his colleagues in 1985 to increase Free Software development. By 1991, the GNU

operating system was almost complete. Only one essential piece was missing, a kernel. (Williams & Stallman 2010, 125-127, 137-141; Stallman 2010, 174)

Figure 1. GNU head, symbol of the GNU Project (Suvasa & Gerwinsky)

A kernel is a program which does a wide variety of tasks like allocating different recourses to other programs in the system and managing processes. The GNU Project had started developing their own kernel, but developing it proved to be difficult due to technical design choices. (Nemeth & Snyder & Hein 2008, 868; Stallman 2010, 27)

In 1991, Linus Torvalds started developing his own kernel and called it Linux. Linus Torvalds developed his kernel separate from the GNU Project and licensed it with a non-free license. He changed his kernel's license to GNU GPL a little later, however, and made it Free Software. Tux, the mascot of the Linux kernel can be seen in Figure 2. (Stallman 2010, 174; Torvald 1991, retrieved 2 March 2013)

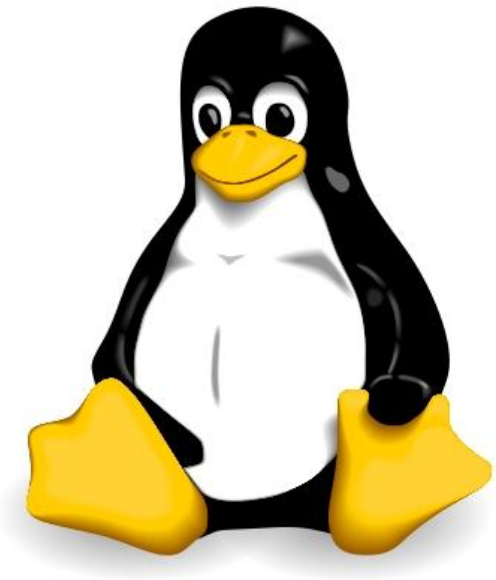


Figure 2. Tux, mascot of Linux kernel (Ewing & Budig & Gerwinski 1996)

In 1992, the nearly complete GNU system and the Linux kernel were successfully combined to form a complete and functional operating system. This operating system is today widely known as “Linux”. This naming implies wrongly that Linus Torvalds started the whole operating system development in the 90’s. It also creates confusion. It is not immediately clear whether “Linux” refers to the operating system or its kernel. To give credit for developers of both the GNU Project and the Linux kernel and to get rid of the confusion, Richard Stallman and the FSF recommend using the name “GNU/Linux” when speaking about the operating system. (Stallman 2010, 28, 53-55, 174-176)

2.3 Hardware compatibility in GNU/Linux

Hardware compatibility is generally speaking good in GNU/Linux. GNU/Linux can work on a great variety of hardware ranging from embedded systems to super computers. It can be used on latest computer models and old machine too. (Gallo 16 April 2013, e-mail)

The two biggest hardware categories with incompatibility issues are video and Wi-Fi cards. When video cards have issues, they are usually partly supported and only lack the support for 3D acceleration. Incompatible Wi-Fi cards, on the other hand, do not work at all. (Gallo 16 April 2013, e-mail)

Hardware compatibility depends on whether the device manufacturer or operating system developer has released the needed driver and firmware for a particular device. If these are not available for one's operating system, the device will not work. It is up to them to choose which operating systems they want any particular device to work on. (Gallo 16 April 2013, e-mail)

2.4 Problem of proprietary drivers and firmware

When a user updates his or her operating system to a newer version, a new version of the device drivers and firmware compatible with that system are needed for all the devices to work. If there is no driver or firmware available for a particular device, it cannot be used anymore. When operating system developers stop supporting a device by not developing and publishing its driver and firmware anymore, the device will become unusable when the user updates the operating system a newer version. This happens if the device's driver or firmware is proprietary software.

Many proprietary software drivers and firmware are released only in binary form without the source code. Binaries are a form of programs which computers can understand easily, but which are exceedingly difficult for humans to understand. Even the most skillful of programmers cannot make sense of them. Since the source code does not come with the driver or firmware, it is very hard for anyone to make a new version to replace the proprietary one. This creates a problem of hardware which can become unusable against its owner's will.

This problem is solved if the driver and firmware is released as Free Software. Then their source code is available to the public and it is likely that new versions are released

even if the official support for the device has been dropped. It is likely that there are skilled people who like to keep using that device. If not, the user still has the four essential freedoms in his or her disposal.

2.5 Fully free GNU/Linux distributions

Many GNU/Linux distributions like Ubuntu and Red Hat include some proprietary software like applications, drivers and firmware. This puts the users of these systems into harm's way of proprietary software.

There are, however, GNU/Linux distributions which do not include any proprietary software but only Free Software. The maintainers of these distributions take great care to remove any pieces of proprietary software from the system, should they find any, in order to guarantee their users their four essential freedoms.

The completely free GNU/Linux distributions are as follows: (GNU Operating System 2013, retrieved 18 March 2013)

- BLAG
- Dragora
- Dyne:bolic
- gNewSense
- Musix
- Parabola
- Trisquel
- Ututo.

The mainline Linux kernel nowadays includes some programs which are non-free software like proprietary device drivers and firmware and some programs which are released only in binary form without the source code. Therefore, the completely free GNU/Linux systems cannot use the mainline Linux as their kernel without removing the

non-free parts first. Fortunately there is such a version of Linux already available. This cleaned kernel is called Linux-libre and its mascot, Freedo, can be seen in Figure 3.

Figure 3. Freedo, mascot of Linux-libre kernel (Pérez 2009)

Linux-libre is an official part of the GNU Project. FSF's sister organization in Latin-America, the Free Software Foundation Latin-America, is taking care of the cleaning process. They copy the mainline Linux kernel when a new version is released and run programs which identify and remove all non-free parts of the kernel. After that, they release the completely free Linux-libre kernel.

2.6 Existing GNU/Linux hardware databases

Many GNU/Linux distributions have their own specific hardware compatibility database which their users can use to check whether a particular device works with their distribution or not. These databases usually list laptop computers and other devices.

Because many GNU/Linux distributions contain non-free software such as proprietary drivers, their hardware databases have hardware which might become unusable due to

dependence on proprietary software. Some of the hardware listed on these databases as “compatible” is therefore not reliable in the long run. This is problematic for users who are looking for reliability.

Another issue, although not as big as the previous one, is that one distribution’s database might not be usable for other distributions’ users. All distributions are different and thus might have different levels on compatibility with same hardware. Multiple databases also waste effort. Same hardware needs to be tested and reported many times.

2.7 H-node

H-node is a hardware compatibility database for GNU/Linux. It has many features which set it apart from the others. H-node has a bigger mission, a wide and growing selection of different devices and potential to become the most important hardware database for the whole GNU/Linux community. H-node’s logo can be seen in Figure 4.

Figure 4. Logo of h-node (Gallo 2010)

H-node’s objective is to give its users information about how well a particular device in its database works with Free Software and Free Software only. This means that h-node is the central hardware compatibility database of the whole Free Software community, not just for users of any single distribution. Users of h-node can use any of the completely free GNU/Linux distributions they want like Trisquel or Parabola.

H-node has numerous categories of hardware. Along with laptop computers there are separate categories for various internal and external devices of the computer like Wi-Fi, video, Ethernet and sound cards and webcams, scanners, printers and modems.

H-node is accessible through its website in *www.h-node.org*. Users can add and edit hardware entries anonymously or create a user account and make their changes identifiable with their usernames. Every user action is registered and can be overwritten with previous ones like in Wikipedia.

H-node has its own documentation to help its users. There is a wiki, an FAQ and Help pages which can all be edited by the users. H-node is Free Software and all of its hardware data and source code are downloadable.

H-node is not beneficial only to the Free Software community. Those users who are using a distribution which includes non-free software can use h-node to check which devices are compatible with GNU/Linux when acquiring new or used hardware. Since h-node's data comes from Free Software distributions only, hardware listed as "compatible" should work with non-free distributions too because they should have the same Free Software drivers and firmware as the free distributions do. Some devices listed as "incompatible" might work for them too but could prove to be unreliable due to dependency on proprietary software in the long run.

H-node is still in beta phase and many features are not implemented yet. It is constantly growing. New hardware is inserted to the database, new hardware categories are made and new features are implemented and planned.

H-node has potential to become the most important hardware database for the whole GNU/Linux community. It can help users to choose hardware that will work with their operating system, whether it be a free one or not, and guide their purchasing decisions which, in turn, will have influence toward hardware manufacturers.

The ideal situation would be if hardware manufacturers wouldn't make any hardware that is dependent on proprietary software but, instead, release all of their software which the hardware is dependent on as Free Software. A world like that is still a distant dream, but h-node can get us a few steps closer at least.

2.7.1 Short history

H-node was created by its current main developer, an Italian Antonio Gallo in summer 2010. He wrote a message on Trisquel GNU/Linux's mailing list about his website inviting other users to insert their hardware to the database. At the time, only laptops, Wi-Fi cards and video cards could be inserted. (Gallo 6 February 2013, e-mail)

H-node's original domain was *h-node.com* owned by Antonio Gallo. In February 2011, Richard Stallman contacted him and asked if h-node could become one of FSF's Free Software promoting activities and if h-node could be moved to a new domain in *www.h-node.org* owned by the FSF. (Gallo 6 February 2013, e-mail)

H-node became a FSF activity, which is why FSF's logo can be seen on h-node's header and FSF started promoting h-node on its own website. Later the same year, h-node was moved to FSF's own servers where it is currently hosted. FSF also provides Antonio Gallo technical support when needed. (Gallo 6 February 2013, e-mail)

Free Software Foundation had its own hardware database. They decide to integrate it to h-node. After Donald Robertson, the person responsible for FSF's hardware database and Antonio Gallo discussed to find the best way for the integration, it was decided that it would be best to ask for the Free Software community to manually insert the devices to h-node. The integration has been completed. (Gallo 6 February 2013, e-mail)

3 ANALYSIS

H-node.org website acts as a user interface for the hardware database. Through it, users can access available hardware information, edit existing entries and add new ones. The website also has sections which are not directly related to using the database itself like documentation to help users to test and report their devices and users' profile pages, for example.

In order to get a clear picture of h-node.org and its functions, it is necessary to go through it section by section. The site has a simple page layout which is present in every section of h-node. This layout keeps the function specific sections uniform in style with each other and makes navigating and using the site simple.

3.1 Page layout

At the top of each section, there is a header which contains the main menu giving access to all sections of h-node. The header also contains logos of h-node and the Free Software Foundation. The main menu gives access to the following main sections: *Home*, *Hardware*, *Issues*, *Search*, *News*, *Download*, *Help*, *Wiki* and the *FAQ*.

Below the header on the right, there is a narrow panel which contains additional functions and information. Among others, it provides access to different translations of h-node, a log in field, a link to new user account registration form, a short history of resent changes made to the database and a table showing how many different devices are listed in each device category and how many users are currently logged in. The mobile version of h-node can also be accessed through the side panel. The rest of the page, which covers most of the space on each section, has section specific content.

3.2 Home

H-node's home page, shown in Figure 5, provides the user brief information about h-node and its objectives. Users are told how they can contribute back to h-node and why it is necessary to use only certain GNU/Linux operating system distributions which consist of only Free Software when testing various devices. Users are also noted that all text they add to h-node will be put in the public domain.

At the top, important notifications are presented when necessary like in case of server maintenance. At the bottom of the page, a few links to other free hardware related websites are available.

Figure 5. H-node's home page (H-node.org 2013a)

3.3 Hardware

H-node's hardware page, which is shown in Figure 6, has a list of all available hardware categories in the database. Through this list, every single hardware category can be accessed and its individual devices can be listed as a whole and also separately accessed and viewed in detail.

Many new categories of hardware have been added to h-node since its creation. Many details, like unique device and vendor identification numbers, have also been added to already existing categories.

In time of writing, there are 16 available hardware categories in total. New categories are planned and will be implemented when they are ready.

Figure 6. H-node's hardware page (H-node.org 2013a)

The hardware categories are as follows: (H-node.org 2013)

- Notebooks, netbooks and tablet PCs
- Wi-Fi cards
- Video cards
- Printers and multifunction
- Scanners
- 3G cards
- Sound cards
- Webcams
- Bluetooth devices
- TV/Video/FM acquisition cards
- Fingerprint readers
- Ethernet cards
- SD card readers
- Modems and ADSL cards
- RAID adapters
- Host controllers.

In Figure 7, a view of one device category can be seen. In this view, users can see a summarized view of the corresponding category's devices. For example in *Notebooks, netbooks and tablet PC* category view, users can see the model name of a device, its vendor, which year the device was commercialized and its compatibility level. Each device category has similar details available of their devices. The nature of these details varies depending on the type of the device.

Figure 7. H-node's hardware page, PC category view (H-node.org 2013a)

The category view can be changed by using filters. These filters, visible as yellow boxes in the Figure 7, can filter out devices from the category view. The user can, for example, choose to view only devices released in a certain year by a certain vendor with a certain compatibility level. The user can also use a model name based filter which he or she can freely write on a field below the yellow filters.

More detailed device view can be accessed though a link which can be found among the devices' summarized details in the category view as can be seen in Figure 7. This detailed device view, as shown in Figure 8, contains all the data concerning a particular device present on h-node. Again, these details will vary depending on the type of the device. In case of a device of the *Notebooks, netbooks and tablet PC* category, the following details can be seen: (H-node.org 2013)

- Model name

- subtype (notebook, netbook, tablet)
- vendor's name
- device's architecture (x86, x86-64/amd64, …)
- if the device has free boot firmware (BIOS, UEFI, …)
- if a free operating system can be installed
- if the device prevents the user from installing Wi-Fi cards which are not approved by the device's vendor
- year of the commercialization of the device
- compatibility level with Free Software
- on which operating system(s) the device was tested
- which version of the Linux-libre kernel was used during testing
- device's video card model
- device's Wi-Fi card model
- device's webcam card model

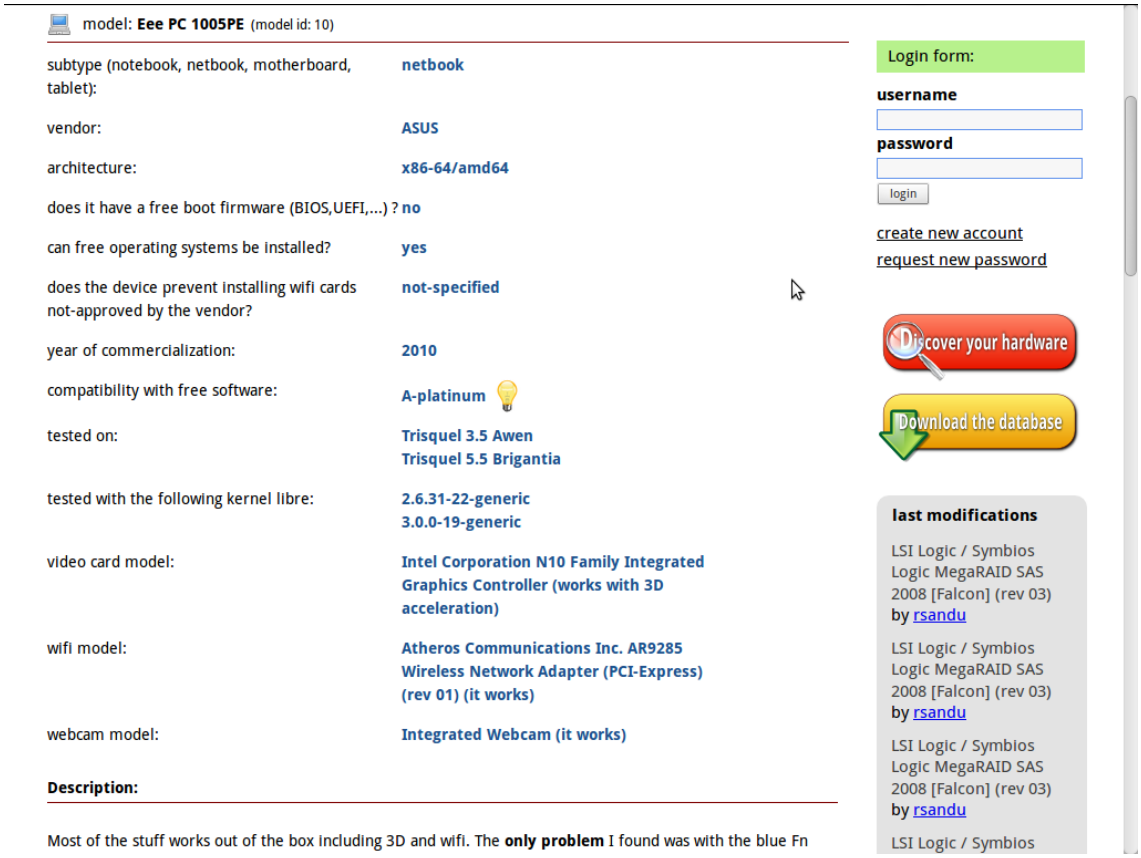
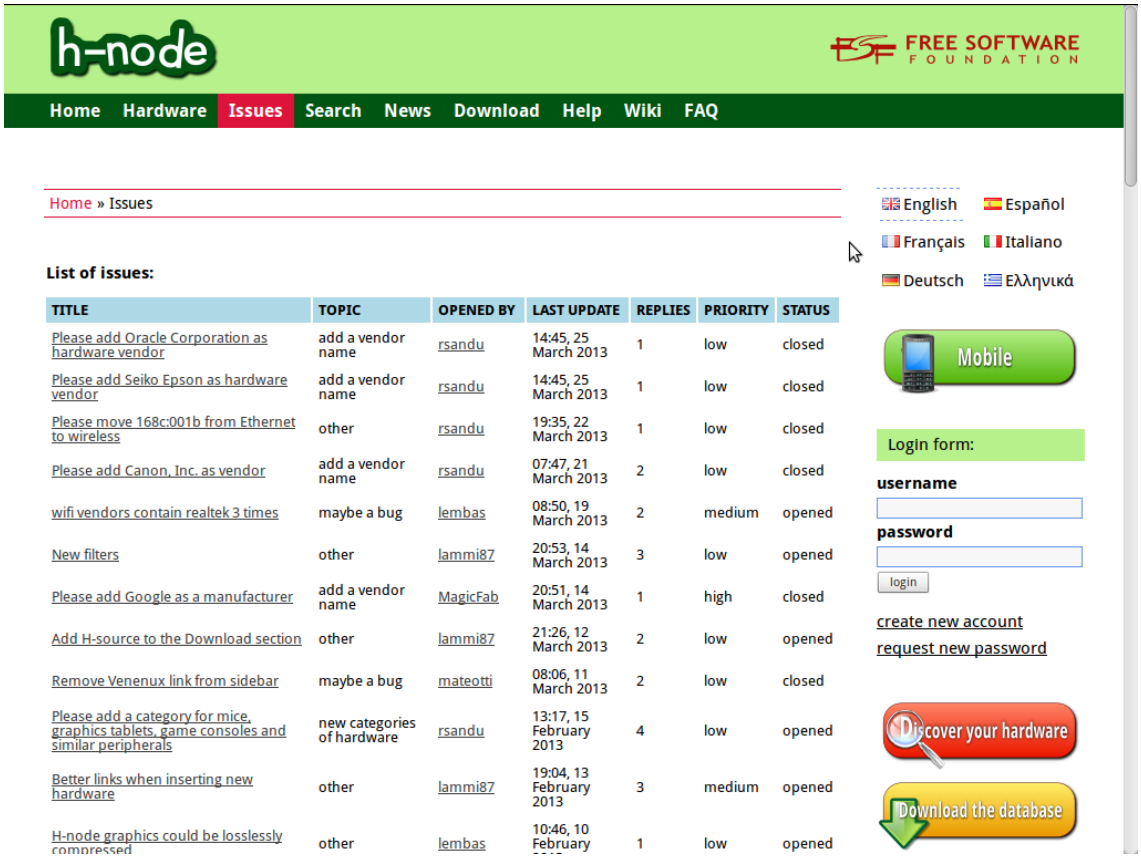


Figure 8. H-node's hardware page, PC category, device view (H-node.org 2013a)

After the details listed above, each device entry has a *Descriptions* field where users can freely add any additional details about a corresponding device. Usually users write about how well the device worked for them, if they had any problems with the device and how they can be fixed.

3.4 Issues

H-node's issues page, shown in Figure 9, has a list of all reported issues on h-node. Here registered users can report new issues or discuss the existing ones. Issues are typically bug reports or requests to implement a certain kind of feature the users would like to have. There are four possible topics for an issue; *Add a new vendor*, *Maybe a bug*, *Add new categories of hardware* and *Other*.



TITLE	TOPIC	OPENED BY	LAST UPDATE	REPLIES	PRIORITY	STATUS
Please add Oracle Corporation as hardware vendor	add a vendor name	rsandu	14:45, 25 March 2013	1	low	closed
Please add Seiko Epson as hardware vendor	add a vendor name	rsandu	14:45, 25 March 2013	1	low	closed
Please move 168c:001b from Ethernet to wireless	other	rsandu	19:35, 22 March 2013	1	low	closed
Please add Canon, Inc. as vendor	add a vendor name	rsandu	07:47, 21 March 2013	2	low	closed
wifi vendors contain realtek 3 times	maybe a bug	lembas	08:50, 19 March 2013	2	medium	opened
New filters	other	lammi87	20:53, 14 March 2013	3	low	opened
Please add Google as a manufacturer	add a vendor name	MagicFab	20:51, 14 March 2013	1	high	closed
Add H-source to the Download section	other	lammi87	21:26, 12 March 2013	2	low	opened
Remove Venenux link from sidebar	maybe a bug	mateotti	08:06, 11 March 2013	2	low	closed
Please add a category for mice, graphics tablets, game consoles and similar peripherals	new categories of hardware	rsandu	13:17, 15 February 2013	4	low	opened
Better links when inserting new hardware	other	lammi87	19:04, 13 February 2013	3	medium	opened
H-node graphics could be losslessly compressed	other	lembas	10:46, 10 February	1	low	opened

Figure 9. H-node's issues page (H-node.org 2013a)

The list view presents brief information about each issue; its title, topic, who opened it, when it was last updated, how many replies it has, its priority (high, medium or low) and its status (opened or closed). A detailed view of an issue can be accessed by clicking its title.

3.5 Search

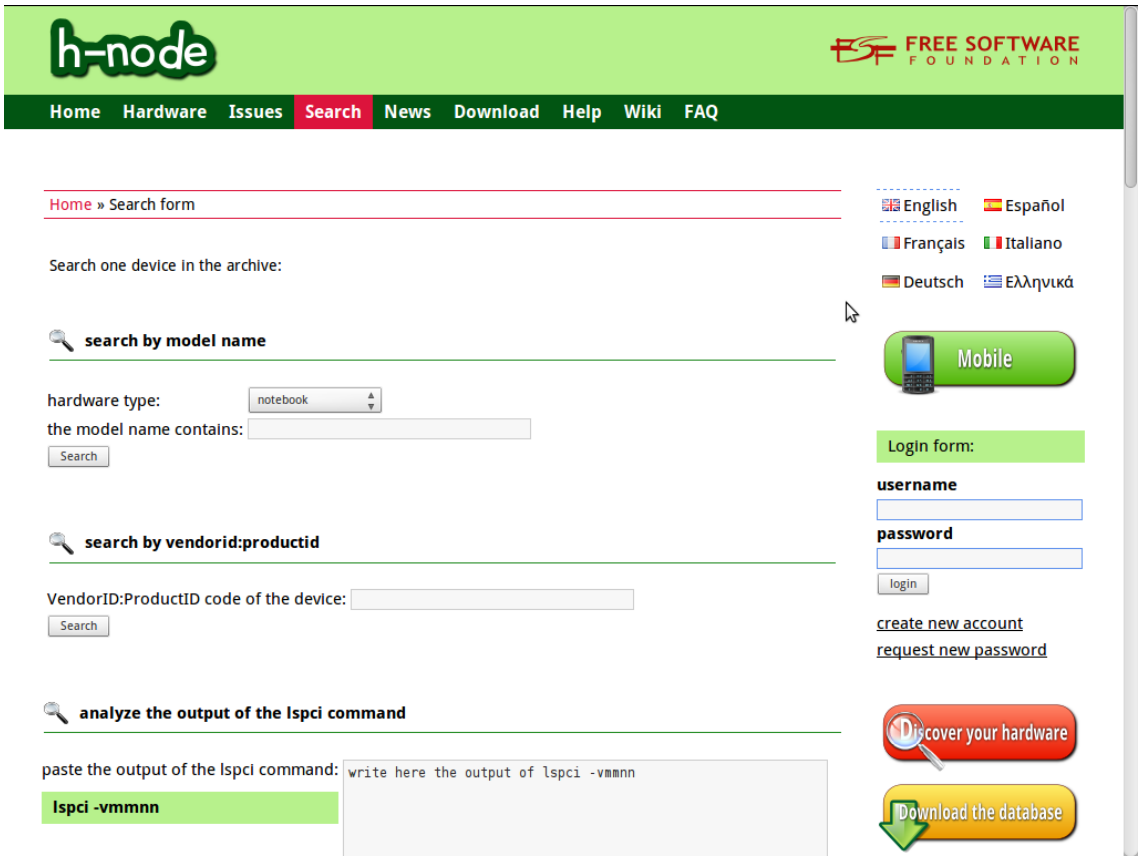


Figure 10. H-node's search page (H-node.org 2013a)

H-node's search page, shown in Figure 10, lets users to search devices from the database and compare their hardware with the database. The former functionality resembles the filters in the hardware page but is still relevant since users expect to find such functionality in the search page and it offers one feature not found in the hardware page: a search by ID numbers. The latter functionality can easily identify which of

user’s devices are already present in the database and which are not, making adding new devices to the database easy.

Users can search devices from the database either by a pair of model name and device type or by using device’s and its vendor’s unique ID numbers. Searching by specific ID numbers is handy, since devices can have similar names.

In order to compare user’s devices with the database, users have to run a specific GNU/Linux command and provide its output to the corresponding field on the search page and click a *Search* button. This functionality is very useful for adding new hardware to the database and will be explained later in detail.

3.6 News

The news section was removed just prior to the completion of this final project. The news section contained news regarding the development of h-node. There were only ten news entries on the page and it was not up to date. According to it, the latest new hardware category added to h-node was webcams. It was added in 2011. Other hardware categories have been added after that.

3.7 Downloads

Anyone can use h-node’s download section, which is shown in Figure 11, to download h-node’s XML based database files. One can choose whether he or she wants to download the whole hardware database as one file or any single device category as its own file.

The files are written in standard XML and can therefore be used by any script or program made for that purpose. All text added to h-node, including the hardware

information, is in the public domain. Thus, the data can be used freely for any purpose. (H-node.org 2013)

H-node's source code is not available from this section. Instead, it can be found in the wiki. There is also a link in the side panel which guides the users to a page where it can be downloaded.

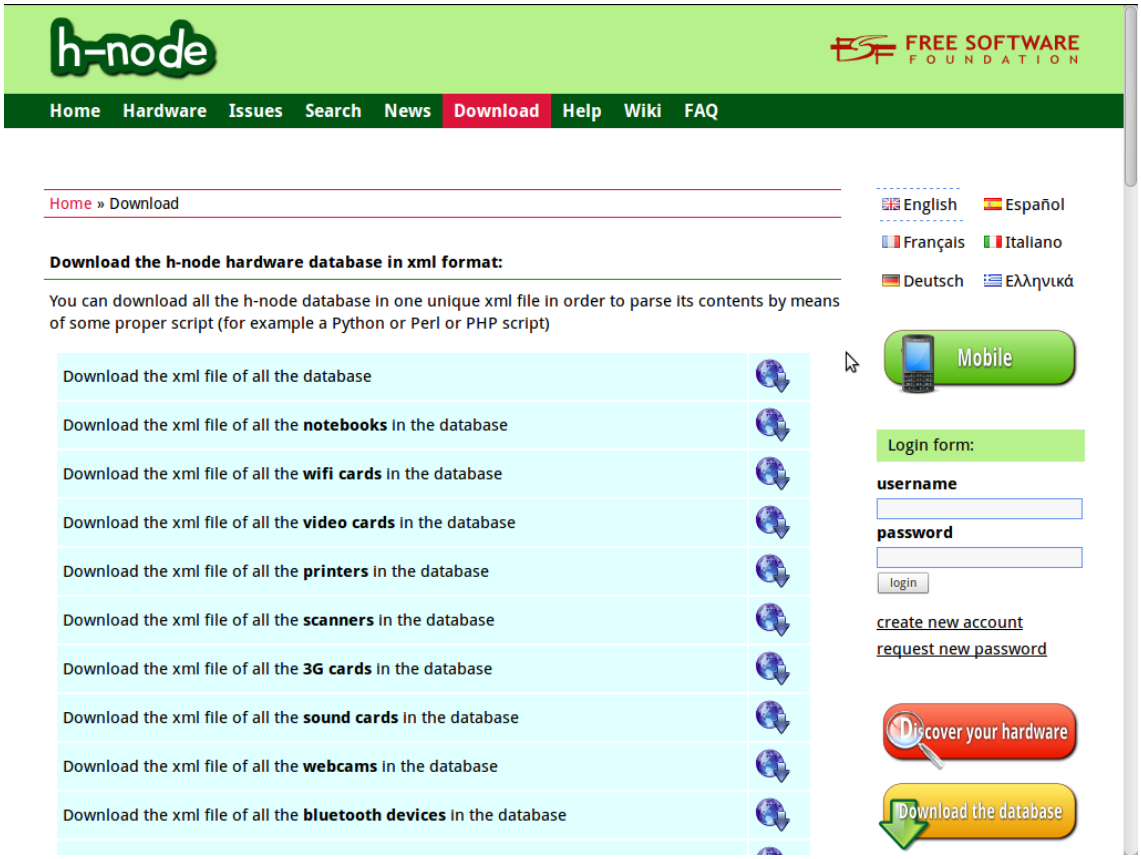


Figure 11. H-node's download page for database files (H-node.org 2013a)

3.8 Help

Help page provides the user links to four different wiki articles and a short explanation about each of them. The user is introduced to the wiki tutorial which provides information about how to modify wiki pages by using wiki tags, an article about how to

acquire detailed information about one's hardware, what are the different compatibility levels used to grade hardware's compatibility with Free Software and what are the fully free GNU/Linux distributions.

H-node's help page was created to help users to use h-node and to provide them instructions and other documentation. When the help page was created, it was a static web page and thus not editable by the users. Back then, the wiki section did not exist yet, so the help page was the only source of documentation for the users. (Gallo 27 March 2013, e-mail)

Later, help page and its content was integrated into the wiki section, as shown in Figure 12. After the integration, it became editable by the users and keeping different translated versions of the help page up to date became easier since all the users could participate in it. (Gallo 27 March 2013, e-mail)

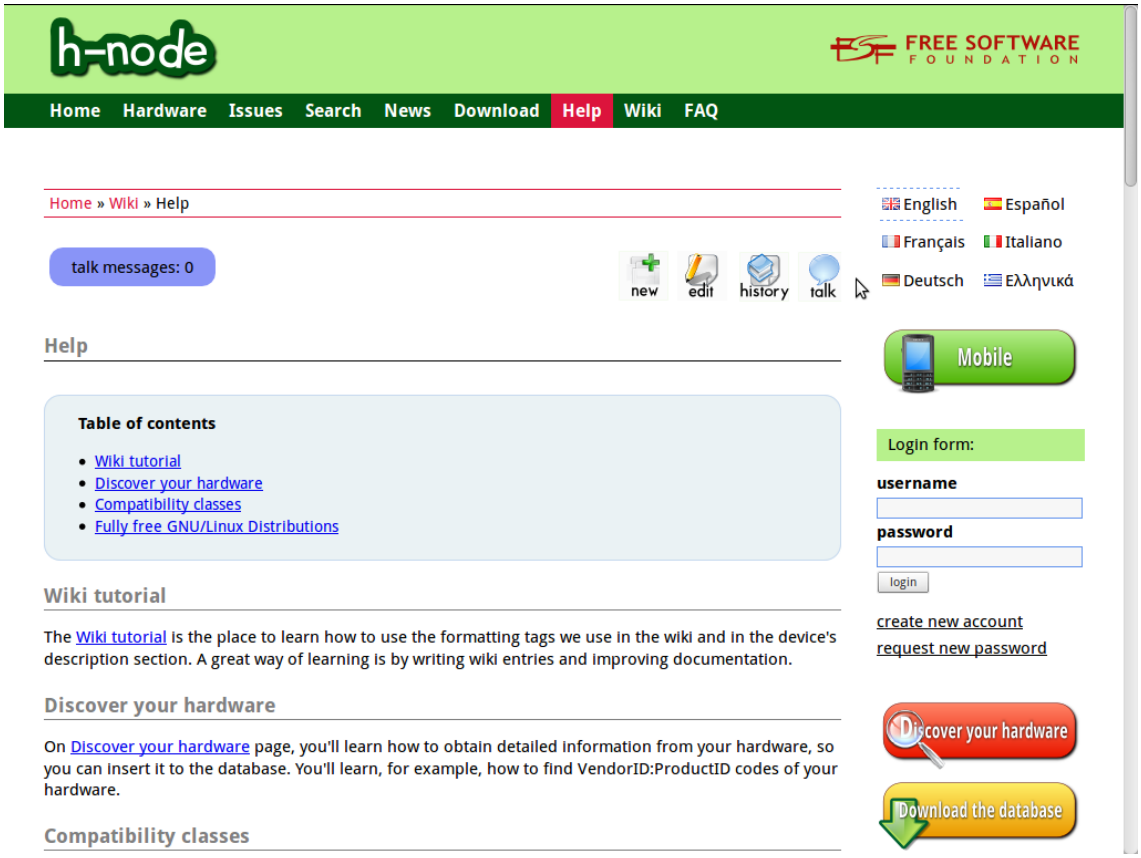


Figure 12. H-node's help page (H-node.org 2013a)

3.9 Wiki

A separate wiki section, shown in Figure 13, was created for user generated content when the help section was still static. The aim was to enable users to participate in writing the h-node's documentation. (Gallo 27 March 2013, e-mail)

Nowadays, the wiki contains various articles about h-node and how to contribute to the project. Anyone can create new wiki articles and edit the existing ones as long he or she is logged in. The changes any user makes to an article can be overwritten if necessary. (Gallo 27 March 2013, e-mail)

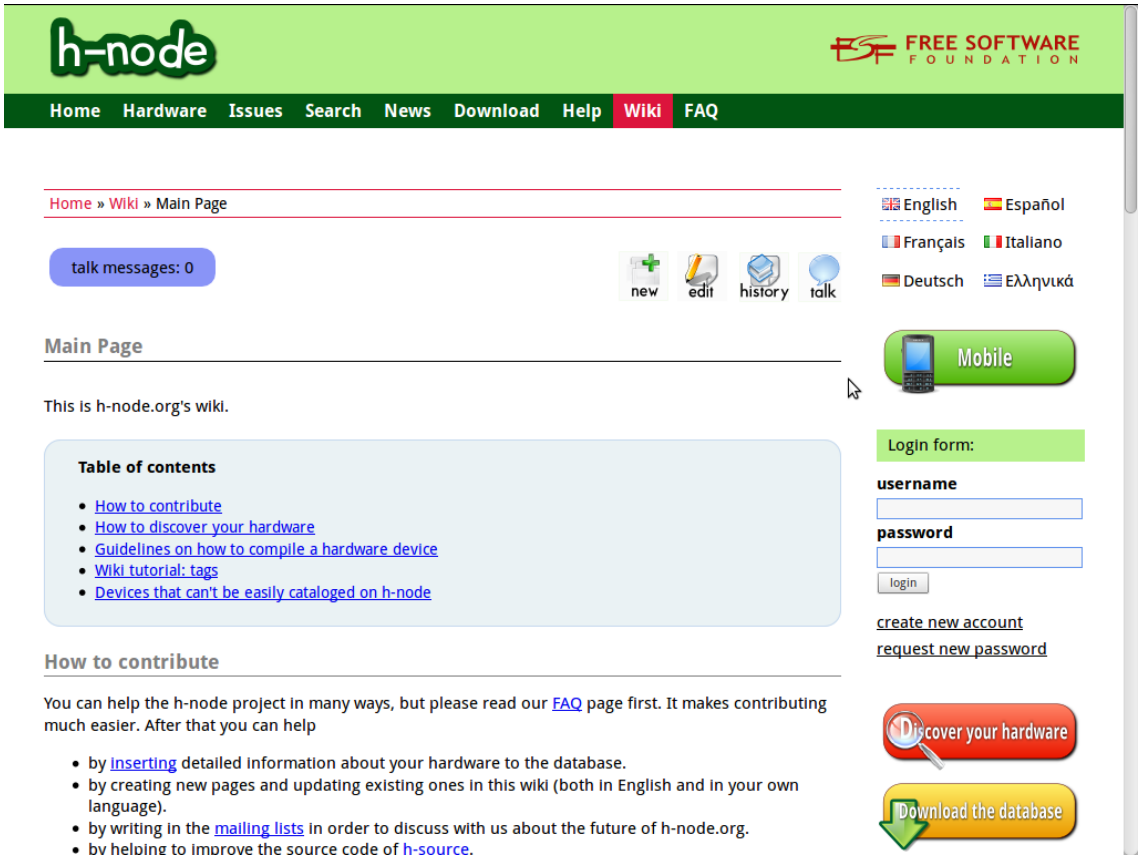


Figure 13. H-node's wiki page (H-node.org 2013a)

3.10 FAQ

H-node's FAQ page, as shown in Figure 14, is a regular wiki article. What makes it differ from the rest is its purpose and a special link in the main header made especially for this article. FAQ's purpose is to act as one of the first resources of information for new users when they are using h-node for the first time.

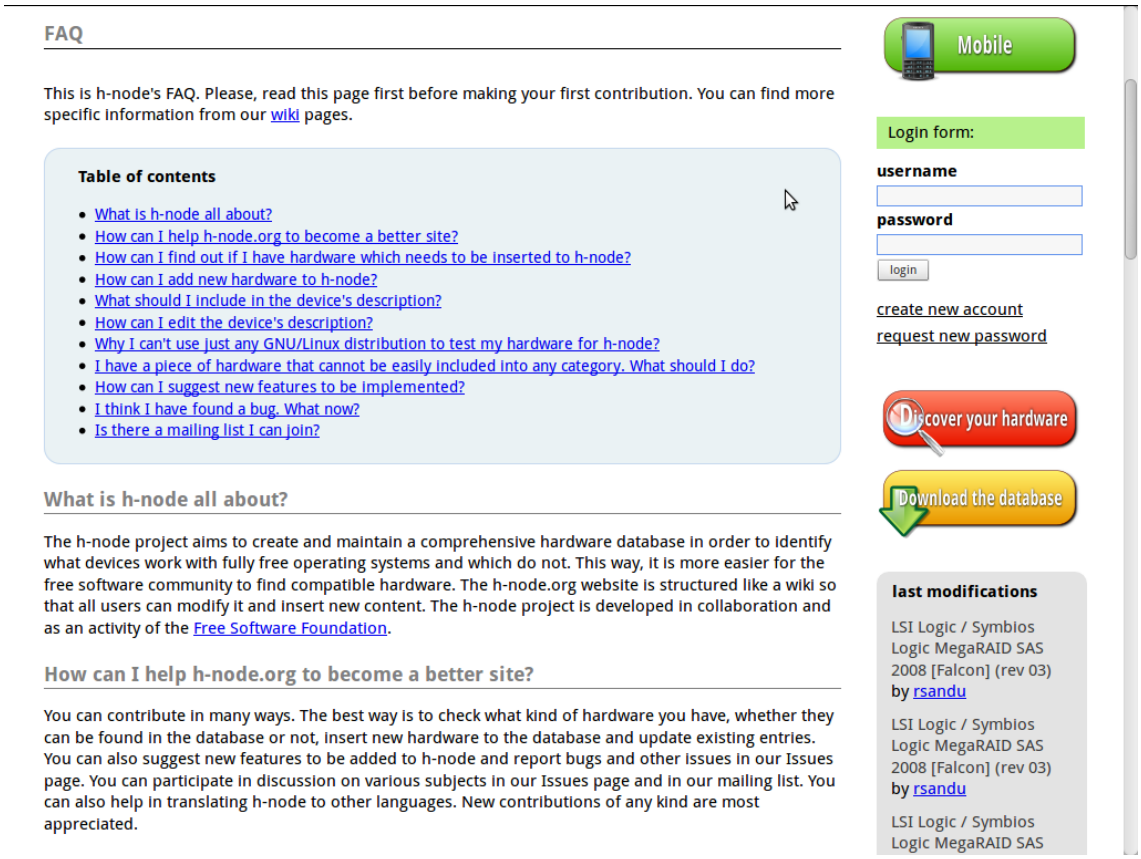


Figure 14. H-node's FAQ page (H-node.org 2013a)

In time of writing, the FAQ has 11 pairs of questions and answers to help new users to understand h-node's objectives and relationship with Free Software, how they can use h-node and how to contribute back.

The FAQ does not give the users long and detailed technical information about how to get information about one's hardware or how to add devices to the database but, instead, gives a short explanation for each question and guides the users to read the dedicated

wiki article related to the subject of each question when such an article exists. The FAQ is written in a way that it effectively is a short beginner’s guide to h-node.

4 DEVELOPMENT

H-node is a young and important project for the Free Software community and, more broadly, to the whole GNU/Linux community. Antonio Gallo, the creator and main developer of h-node, develops and maintains the h-node.org website and the database itself. Developing such an important and huge task which h-node aims to become is very difficult to do alone. Therefore Antonio Gallo decided to ask other members of the Free Software community to participate in the development.

There are many ways for a single user to contribute to h-node. He or she can add missing hardware to the database and update already existing hardware entries. Anyone can also participate in writing new documentation and improving older entries and help in translating h-node to other languages. Those familiar with programming, can contribute to h-node.org's source code, adding new features and fixing bugs. Of course, talking about h-node and its mission with other people is a valuable contribution too since expanding the user base is very beneficial for the project.

4.1 Expanding the database

No matter how well the website is made or how convenient and innovative features it has, a database project falls short without much actual content both old and new. Thus, checking whether one has some pieces of hardware which are missing from h-node or which are in need of updating is probably the most beneficial thing a single user can do. The process of doing this is rather simple but can be a bit daunting for a user who has never done it before and is not at all familiar with GNU/Linux.

The first thing one needs is a way to boot one of the completely free GNU/Linux distributions endorsed by the FSF with one's computer. Then, the user needs to run a few simple commands on a text-based command line interpreter to get information about his or her hardware. That information is then passed on to h-node.org website

which processes the information and tells the user which devices are missing from the database and should be added and which devices are already in the database but might need updating. Finally, the user manually inserts and/or updates his or her devices to the database.

One of the easiest to use free distributions is Trisquel GNU/Linux. It will be used as a platform to demonstrate the contribution process in this section. Bootable disk images of Trisquel and live-CD burning instructions can be found on Trisquel's home page on *www.trisquel.info*. (Trisquel GNU/Linux 2013)

4.1.1 Using Trisquel GNU/Linux live-CD

After the user has acquired a bootable CD/DVD of Trisquel or any other free distribution of his or her choice, it needs to be booted up with the computer the user wishes to test. If the computer does not boot the disk and boots its default operating system instead, the booting order of the machine needs to be changed. This is quite easy to implement.

Usually the user needs to access the BIOS and tell the computer to search for an operating system from the CD/DVD drive first before the hard drive. After that, the disk should boot. Sometimes the user doesn't have to go to the BIOS at all. If this functionality has been implemented in the BIOS, users can directly choose from which device to boot an operating system from by hitting a required key before the BIOS starts to boot the default operating system. This feature is not always implemented and the surest way to configure boot order is through BIOS.

The BIOS is a piece of software which is installed on the computer's motherboard. Its purpose is to do various hardware related inspections, find an operating system to boot and pass on information to the booted system among other things. Nowadays, newer computers might have a similar but newer piece of software on their motherboards called UEFI.

Every computer is different and the way to access the BIOS or UEFI varies. Usually, the user needs to hit a certain key before the booting process of the default operating system starts. This key is usually one of the function keys like F2. After gaining access, the required changes in the boot order should be easy to make since BIOS and UEFI have a simple user interface.

When the live-CD of Trisquel GNU/Linux boots, the user is greeted with the live-CD's main menu as shown in Figure 15. The user can select the language of the system and the keyboard layout he or she wants to use by hitting F2 and F3. After that, the distribution can be booted by selecting *Try Trisquel without installing* from the menu and hitting the enter key.

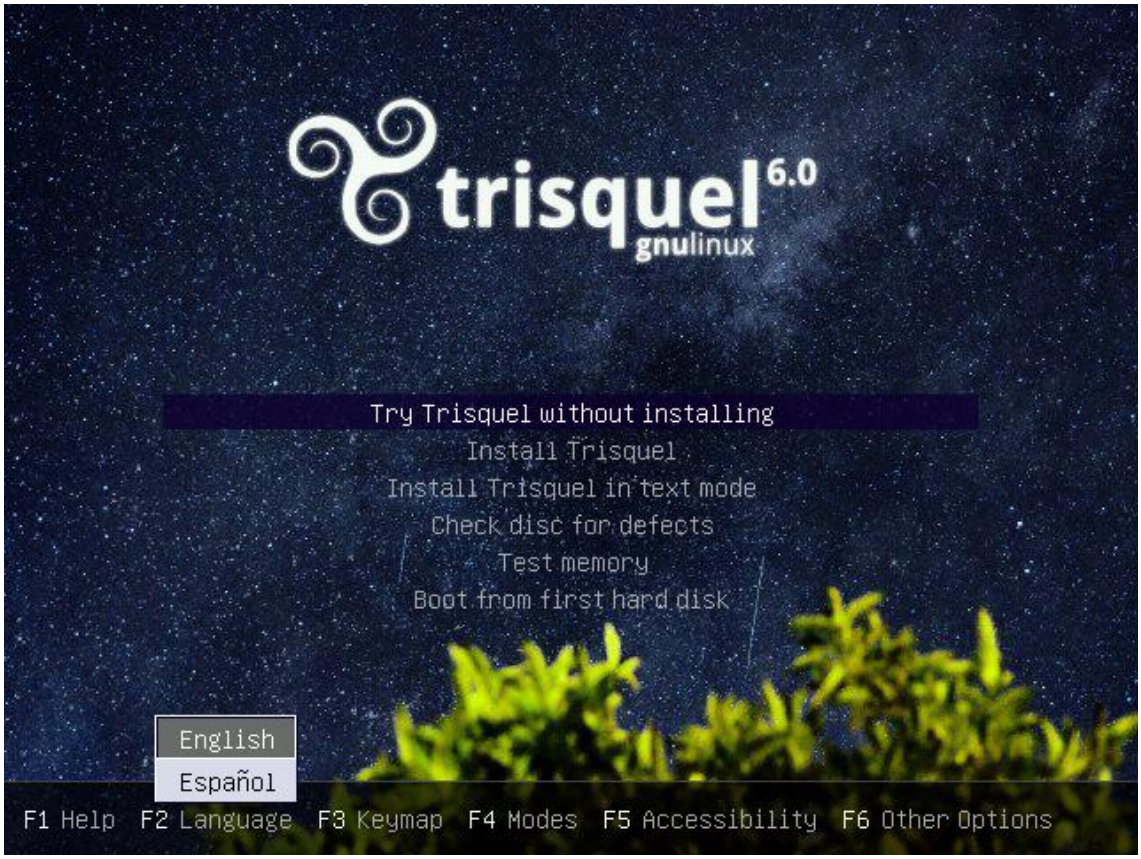


Figure 15. Main menu of Trisquel 6.0 live-CD (Trisquel GNU/Linux 6.0 live-CD 2013)

Any operating system booted from a live-CD rather than the hard drive, is run completely in RAM. This means that every change the user makes to the system, Trisquel in this case, will be lost after the computer is turned off.

The user can try out different features of Trisquel and even install and uninstall programs while keeping his or her default operating untouched. The user can also access his or her hard drive easily if needed. This is handy when the user wants to save files from a broken system which does not boot up anymore. The user can also install Trisquel permanently to the hard drive along with his or her default system or replace it entirely. (Trisquel GNU/Linux 2013)

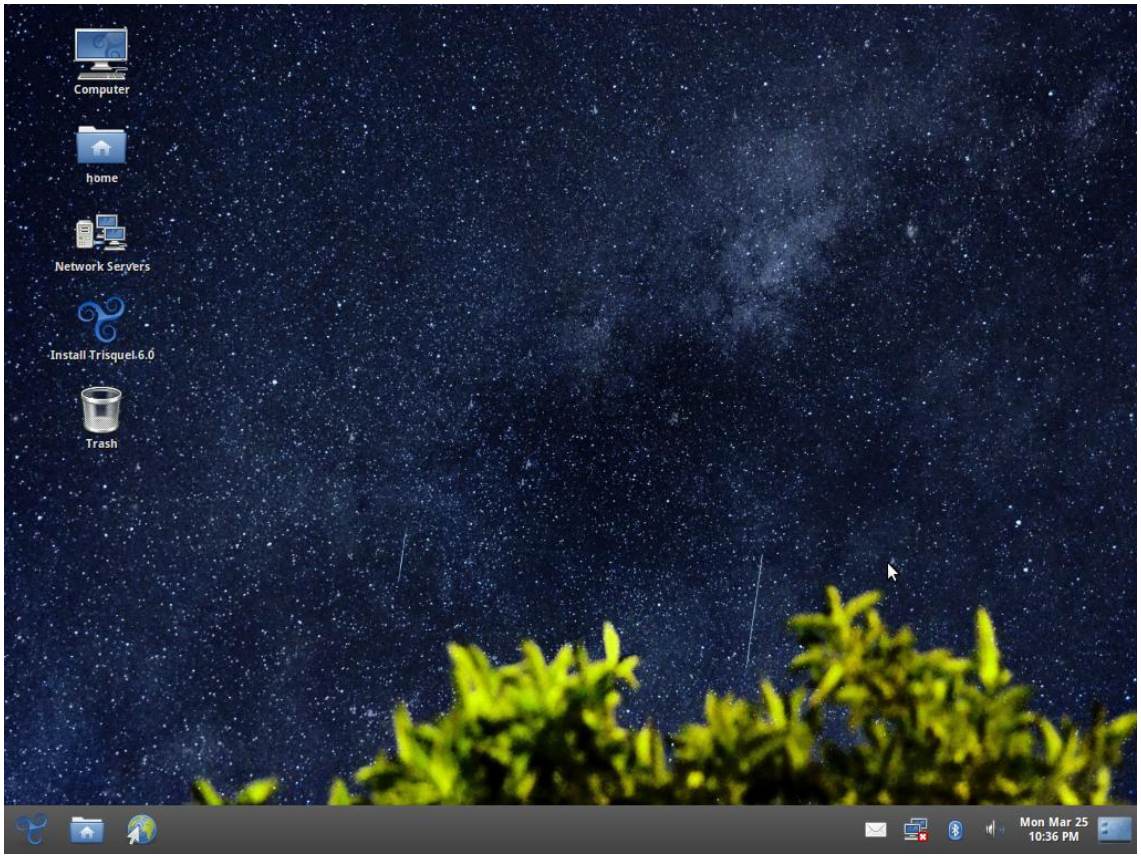


Figure 16. Desktop of Trisquel 6.0 (Trisquel GNU/Linux 6.0 live-CD 2013)

Fully booted and ready to be used system is shown in Figure 16. The main menu can be found by clicking the blue Trisquel logo on the left end of the panel. All applications and locations can be found there. The browser can also be launched by clicking an icon

on the panel which looks like Earth with a mouse cursor on it. The system should already have a connection to the Internet if a wired connection is in use. Wi-Fi controls can be found on the right side of the panel.

At this point, the user should register an account for him- or herself in h-node if he or she has not already done so. If the user does not log in to h-node, the new device he or she adds to the database will not be visible to regular users right away but only after an administrator has approved it. Only those users who have logged in can edit existing hardware entries.

New accounts can be created by clicking the *create new account* link on the right side panel on any page in h-node as can be seen in Figure 14. Those users who already have an account can use the same panel to log in.

4.1.2 Comparing one's hardware with the database

A user can compare his or her devices with the h-node's hardware database by providing information about them to h-node.org's search page. There is a section called *analyze the output of the lspci command* in there. In this section, there is a field which accepts hardware information produced by a command called *lspci*. The user must use a text-based command line interpreter to run the command in order to get the needed information.

A text-based command line interpreter called Bash can be run on a graphical terminal, as seen in Figure 17. The terminal can be launched by hitting Ctrl + Alt + T in Trisquel or by finding a launcher for it in the main menu. The launcher is in the *Accessories* sub-menu.

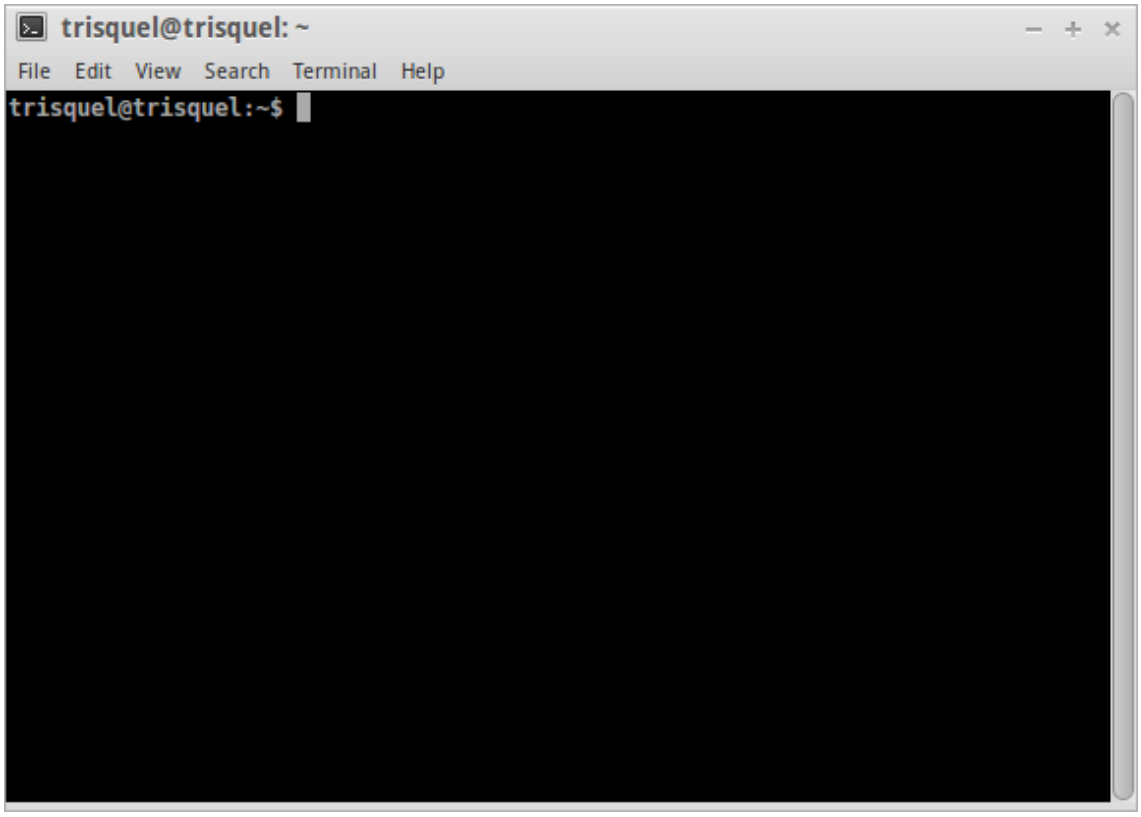


Figure 17. Bash in a graphical terminal (Trisquel GNU/Linux 6.0 live-CD 2013)

The user can run a command just by typing it in bash and hitting enter. The search page accepts the output of the *lspci* command only if *lspci* is used with options *v*, *mm* and *nn*, so they need to be added to the command. Since this final project aims to make accessing the output easy for those users who have not used a command line interpreter before, an additional section will be added to the command which writes the output of the command to a file rather than on the terminal itself. So, the whole command the user needs is:

```
lspci -vmmnn > file.txt
```

This will write the needed output to a text file called *file.txt* which can be found in the user's home folder. The home folder can be found in the main menu or as a link on the desktop. Due to the nature of bash, the command will not work correctly if the space between the command name *lspci* and the options *-vmmnn* is missing or if the name of

the output file has spaces. The two space characters before and after the > character are optional.

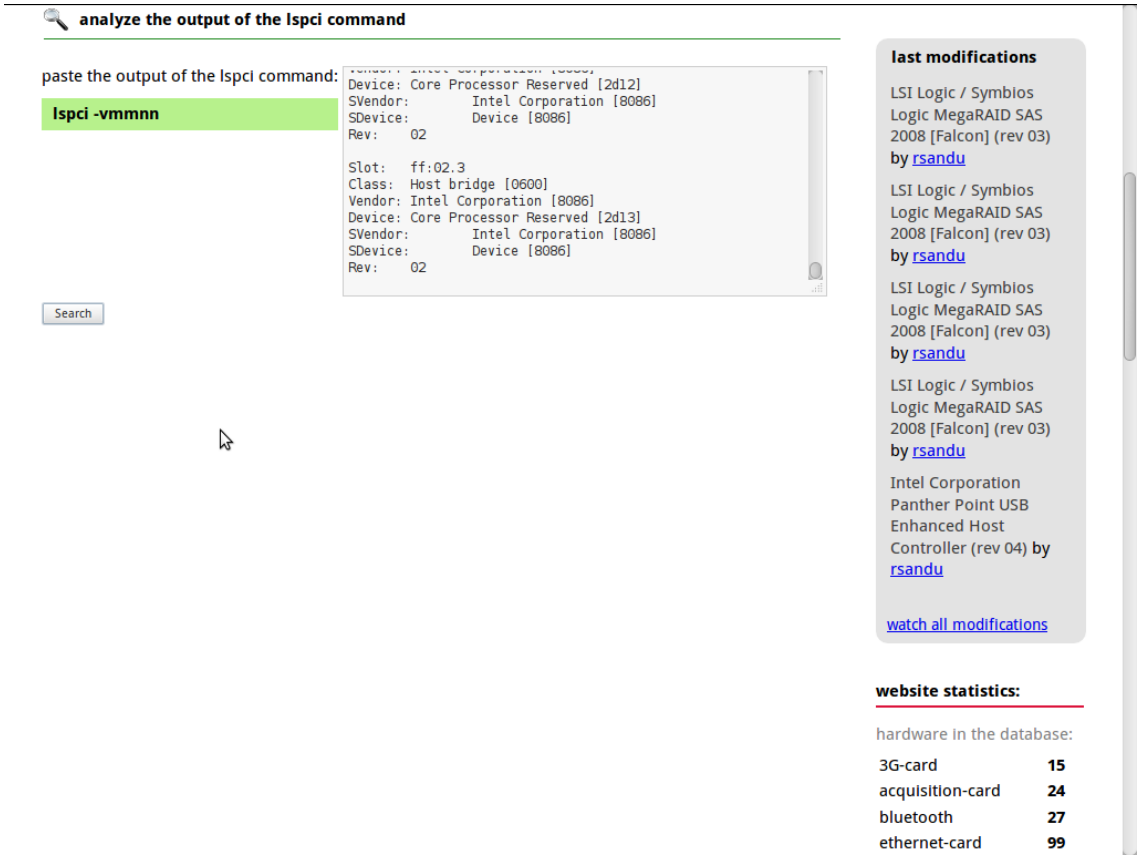


Figure 18. Using the output acquired from the *lspci* (H-node.org 2013a)

After getting the hardware information, the user copies it from the output file and pastes it on the appropriate field in the search page as shown in Figure 18. After clicking the *Search* button, the user will be presented a list of his or her hardware as shown in Figure 19.

All those hardware entries which are on a blue background are already in the database and the user should check if they need updating. Those entries which have a yellow background are missing from the database and the user should add them to it. The list includes only those pieces of hardware which have their own category in h-node since the search page does not recognize uncategorized hardware.

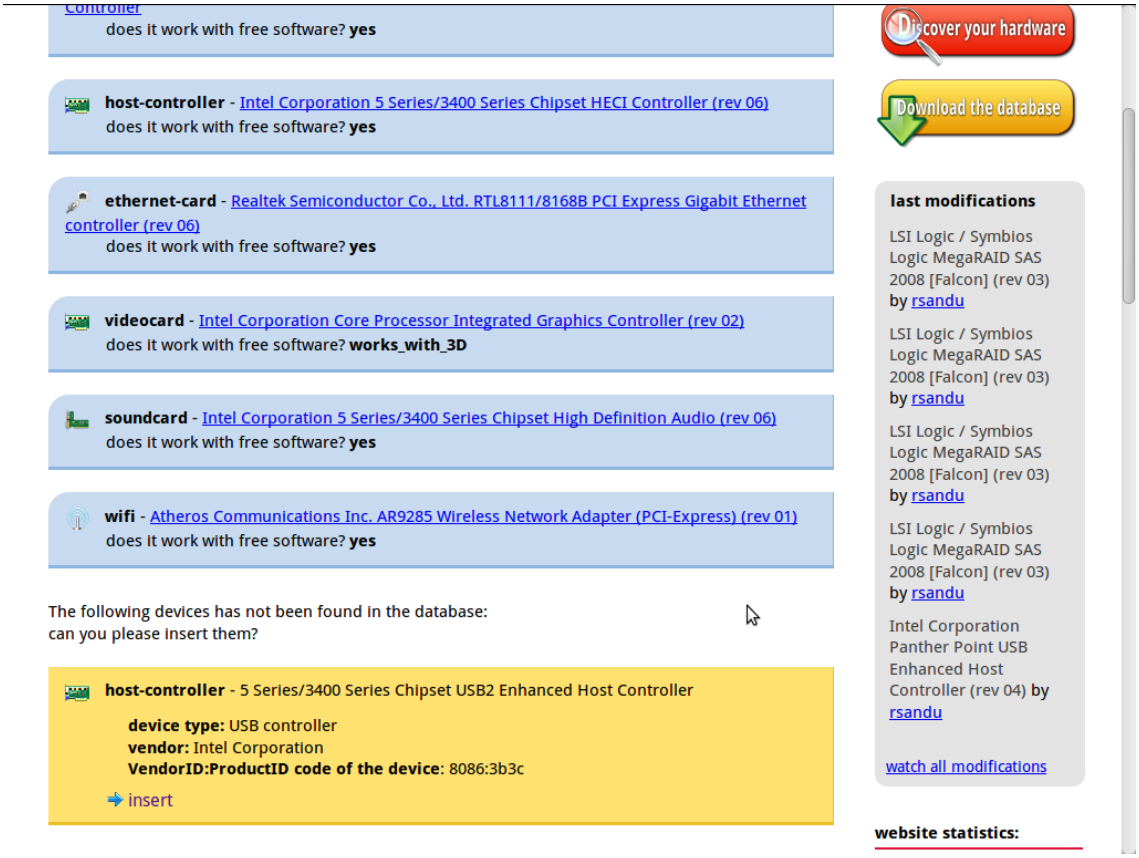


Figure 19. Results of comparing one's hardware with the database (H-node.org 2013a)

4.1.3 Adding missing hardware and updating older entries

The easiest way new hardware entries can be added to the database is to click the *Insert* link at the bottom of each listed yellow hardware entry as can be seen in Figure 19. This will take the user to a corresponding hardware category page where a new hardware entry can be created by clicking the *New* icon as seen in Figure 7. Clicking this icon will take the user to the actual hardware insertion page as shown in Figure 20.

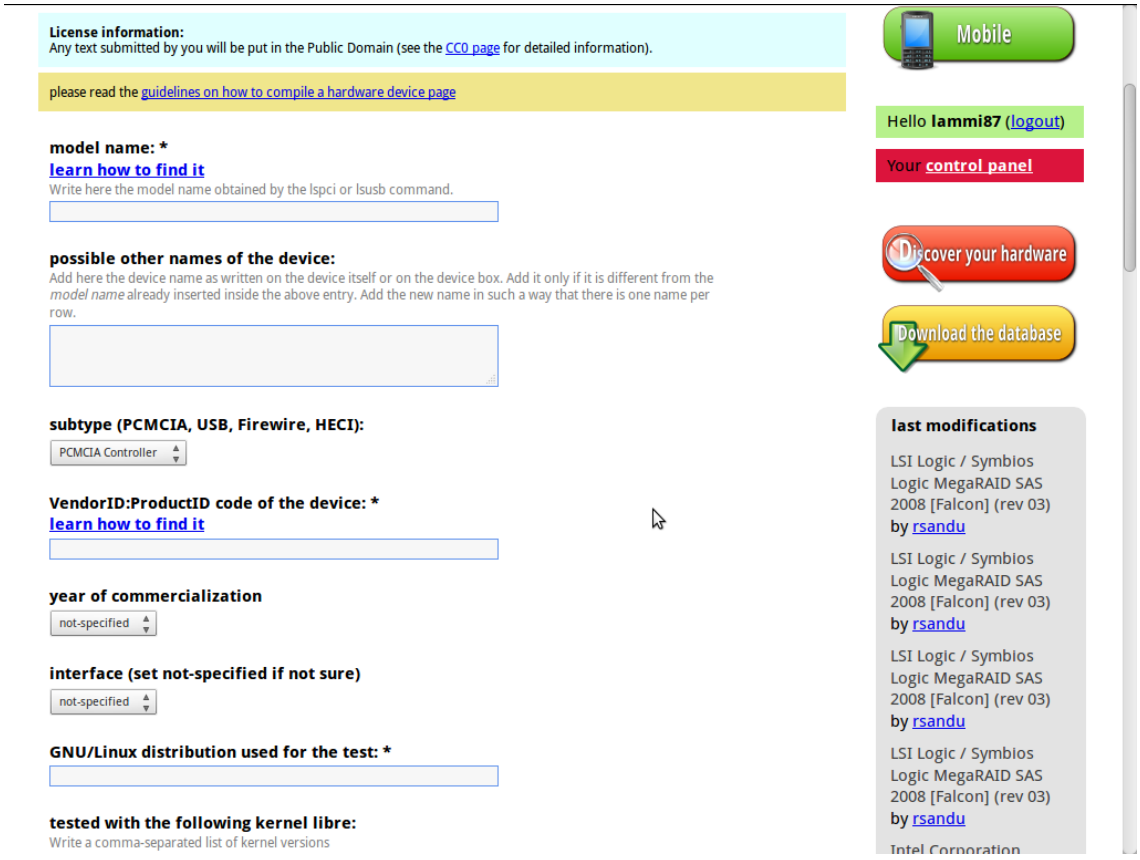


Figure 20. Hardware insertion page (H-node.org 2013a)

Here, the user has to manually insert all the details about his or her device to the corresponding fields. When ready, the user can create the new hardware entry by clicking the *Save* button at the bottom of the page. After that, the new piece of hardware is immediately added to the database and visible to all users as shown in Figure 21 if the user was logged in when adding the device.

The user can find documentation about how to find out the necessary details about his or her device by reading guides which can be found by clicking the two *learn how to find it* links or by searching the *Discover your hardware* article from the wiki.

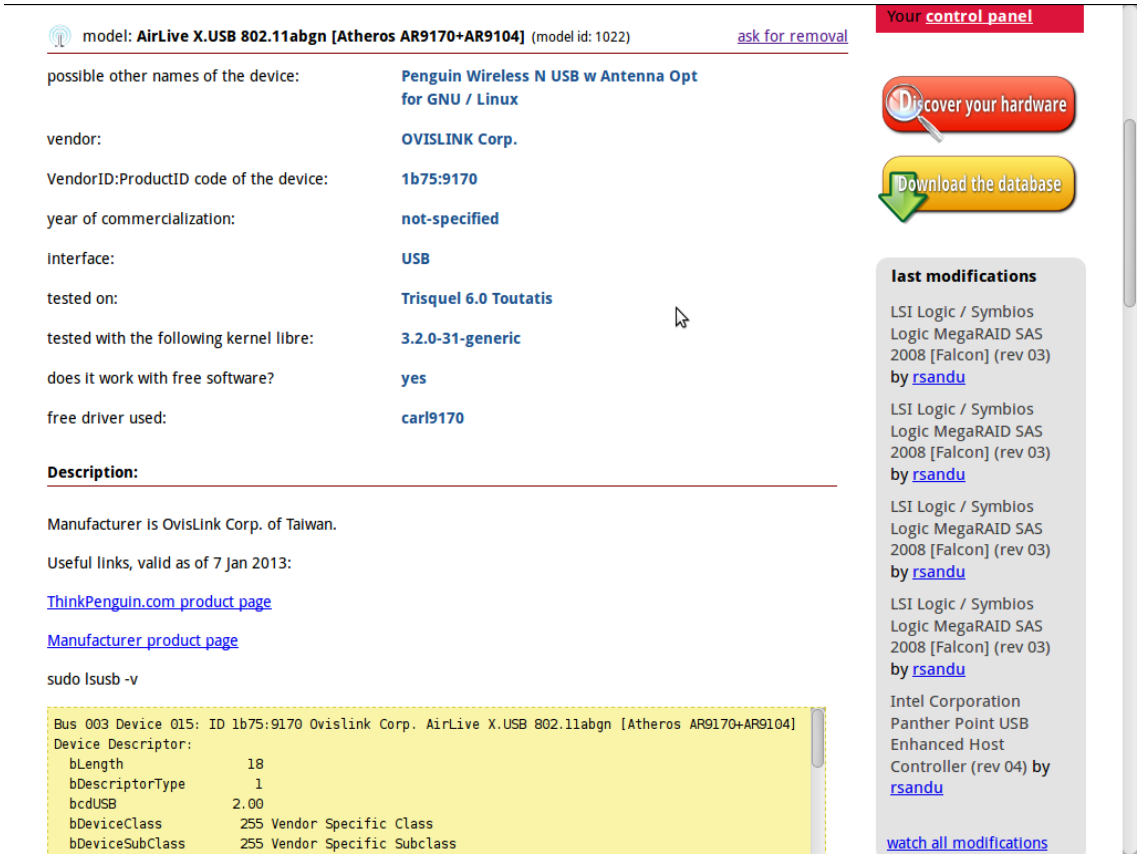


Figure 21. An example of a Wi-Fi card device entry (H-node.org 2013a)

The already existing hardware entries, listed on blue backgrounds as seen in Figure 19, can be accessed by clicking their name which acts as a link. The link will take the user to the corresponding device's page as shown in Figure 21. Here, the user can click the *Edit* icon at the top of the page to edit the device's information. The user will be taken to the hardware insertion page, as seen in Figure 20, which has some information about the device already inserted by other users.

After editing the device's details the user can save the changes he or she made by clicking the *Save* button. The new version of the device page will replace the older one. However, the older one can be restored from the page's history section if wanted.

4.1.4 Example: Testing a laptop for h-node

The whole process of testing a laptop for h-node will not be introduced here. Only the most important phases will be discussed. A Dell Vostro 3300 laptop will be used as an example machine. This laptop and its internal devices can already be found in h-node but, for the sake of this example, that fact is ignored.

After booting the machine with Trisquel GNU/Linux, its internal devices will be compared with the h-node's hardware database as described before. Then, the existing device entries will be checked if they need updating and will be updated if necessary. Then, all the missing devices will be inserted to the database if they can be successfully tested to see whether they work or not. If some device cannot be tested for whatever reason, It will not be added to the database.

In order to keep this example simple, It will not be explained how to get all the details about each device such as model names and ID numbers because the needed documentation for that can be found in the h-node's wiki section.

Let us assume that the following results are got after using the output of the *lspci* command in the h-node's search page:

- Wi-Fi card (present in h-node)
- Video card (present in h-node)
- Ethernet card (missing from h-node)
- Sound card (missing from h-node).

First, the Wi-Fi and video card's details are checked and updated if necessary. They will be tested whether they work or not and the findings are reported. Testing the Wi-Fi card is easy and quickly done. If the laptop can be connect to the internet by using a wireless connection, then the Wi-Fi card works. The network connections can be accessed on the right side of the panel on the Desktop or from the main menu: System Settings > Network Connections.

The video card is not as easy to test. It has three different compatibility levels where as the Wi-Fi card has just two; *works* and *does not work*. Video cards have an additional 3D acceleration feature which either is or is not supported. The cards can work without it and the computer can be used without 3D acceleration being supported but the computer will then have some problems with displaying high quality 3D graphics. The compatibility levels of a video card are thus; *works with 3D acceleration*, *works but without 3D acceleration* and *does not work at all*.

If Trisquel GNU/Linux can be successfully booted and the desktop view accessed, then the video card is at least partly supported. After that, all that needs to be done is to test whether the 3D acceleration is supported or not. Any application which uses 3D effects can be used to estimate this. If the application works fluently, the acceleration should be supported.

A simple 3D table tennis game called *Cannon Smash*, as shown in Figure 22, can be used to test the video card's 3D acceleration feature. This game is a very quick way to learn whether the 3D acceleration is supported or not. The game has a 3D animation as a background for its main menu and, just by starting the application, it can be estimated whether the video card is fully supported or not. This game can be installed by searching it from the package repositories by accessing the *Add/Remove Applications* entry in the main menu or by running the following command on the terminal:

```
sudo apt-get install csmash
```

Figure 22. Cannon Smash game (Trisquel GNU/Linux 6.0 live-CD 2013)

Ethernet card is maybe the easiest device to test. If a connection to the Internet can be acquired by using an Ethernet cable, the card works. The sound card is also relatively easy to test. A sound test can be used to test the card. The test feature can be found in the main menu: System Settings > Sound > Hardware > Test speaker. If the test sounds can be heard, the card works.

After testing the Ethernet and sound cards, they are inserted to the database, as described in section 4.1.3 *Adding missing hardware and updating older entries*. A wiki article called *Discover your hardware* can be used as a reference for getting all the necessary hardware details.

After updating the existing hardware entries and adding all the missing ones which could be tested, only the laptop itself needs to checked whether it is already in the

database or not. The database sees a laptop as a regular device entry so adding or updating a laptop is not really different compared to any other device. Only the device details are different.

The search page can be used, as shown in Figure 10, or the hardware page's *Notebooks, netbooks and tablet PC* category view, as shown in Figure 7, to search for the laptop from the database. If it can be found, it needs to be checked whether it needs updating just like before with the Wi-Fi and video cards in this example. If the laptop cannot be found, it needs to be inserted to the database.

Knowing one's laptop's full model name is not always easy. Usually the surest way is to check below one's laptop and search for a sticker on which the model name is printed. Sometimes this sticker is under the battery which must be removed in order to see it.

Usually the model name is quite simple like in case of this example: *Vostro 3300*. Sometimes, however it can be rather complicated like in some Acer's laptops; *Aspire 1825PT-734G32n (ZE8)*. It is important to report the full model name of a laptop. The manufacturers can release different versions of the same model assembled with different components and thus these versions can have totally different compatibility levels with Free Software.

Before adding a new laptop to the database, a few things must be tested first. The webcam must be tested, in case the laptop has one, and the Wi-Fi and video cards too. Since Wi-Fi and video cards were tested earlier already, it is not necessary to test them again. The webcam can be tested easily with applications like *Cheese* and *Camorama Webcam Viewer*. If a stream of video can be seen, it works. Both applications should be used, since sometimes some cameras do not work with *Cheese* but will work just fine with *Camorama*. Both can be installed easily from the *Add/Remove Applications* entry in the main menu or by running the following command on the terminal:

```
sudo apt-get install cheese camorama
```

4.1.5 Example: Unusable or otherwise problematic laptop

It is important that all kinds of laptops are reported to h-node, both compatible ones and less compatible ones. Even those which are totally unusable must be reported so that everyone can know which devices to avoid.

Sometimes when testing different laptops, a user may run into incompatibilities which will make it difficult to report the laptop to h-node. Based on the experience of the author of this final project, such situations are really rare but possible. So, it is necessary to elaborate what one can do should he or she face such a machine.

Reporting testing results to h-node becomes difficult when both the Ethernet and Wi-Fi cards are incompatible with Free Software meaning that the user cannot connect to the Internet. Another problematic situation is when the video card is barely functional enabling booting only to text based interface instead the graphical desktop.

In case of a pair of incompatible Ethernet and Wi-Fi cards, all the tests can be performed normally. The only difference is that all the test results have to be saved to a file, transferred to another computer via USB pen drive, for example, and reported with a machine which has a working connection to the Internet.

If the video card is totally incompatible, then hardly any tests can be performed to see which devices of the laptop work and which do not. A working computer must be used to check if that laptop can be found in h-node. If it cannot, it must be added to the database with the lowest compatibility class for laptops: E-Garbage. It means that the laptop is basically unusable. Comments should also be added to the description section explaining how the laptop was tested, what could and could not be tested and why it was given the lowest compatibility class.

4.2 Expanding the wiki

Every project needs a good documentation and h-node is no exception. All documentation of h-node can be found in the wiki section as wiki articles. These articles can be freely edited by any user who has logged in. Likewise, new articles can only be created by users who are logged in.

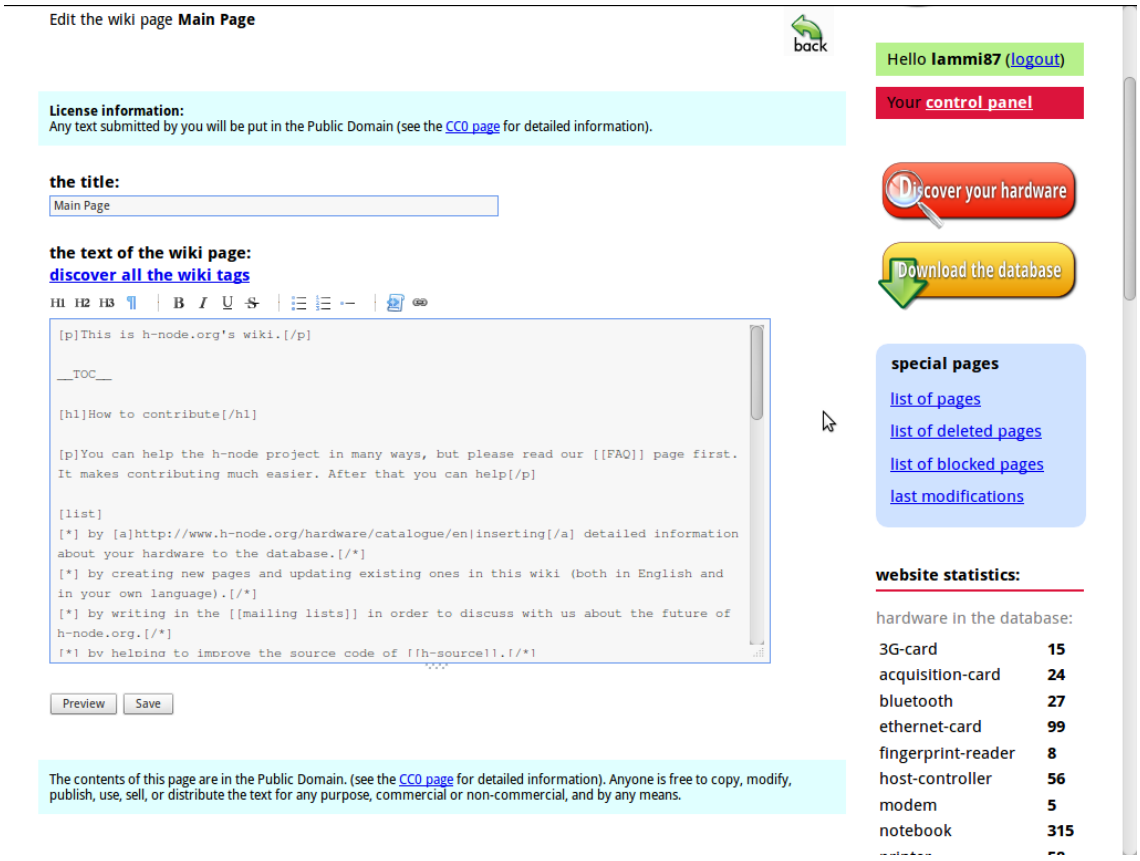


Figure 23. A view of the wiki article editing page (H-node.org 2013a)

To edit an existing wiki article, the user needs to click the *edit* button of the corresponding article which can be found at the upper right part of the page below the header as shown in Figure 13. The user will be taken to the wiki editing page, as shown in Figure 23, where he or she can find a simple user interface with all the needed tools to edit the article.

As can be seen in Figure 23, the user can edit the article by editing its source code. The h-node's wiki articles are mostly written in plain text but additional wiki tags are used to format the appearance of the article. A tutorial about how to use these tags can be found by clicking *discover all the wiki tags* link above the source code editing field. These tags can also be used when writing a description for a device.

After making some changes, the user can preview the new page before saving it by clicking the *Preview* button. A preview view of a wiki page is shown in Figure 24.

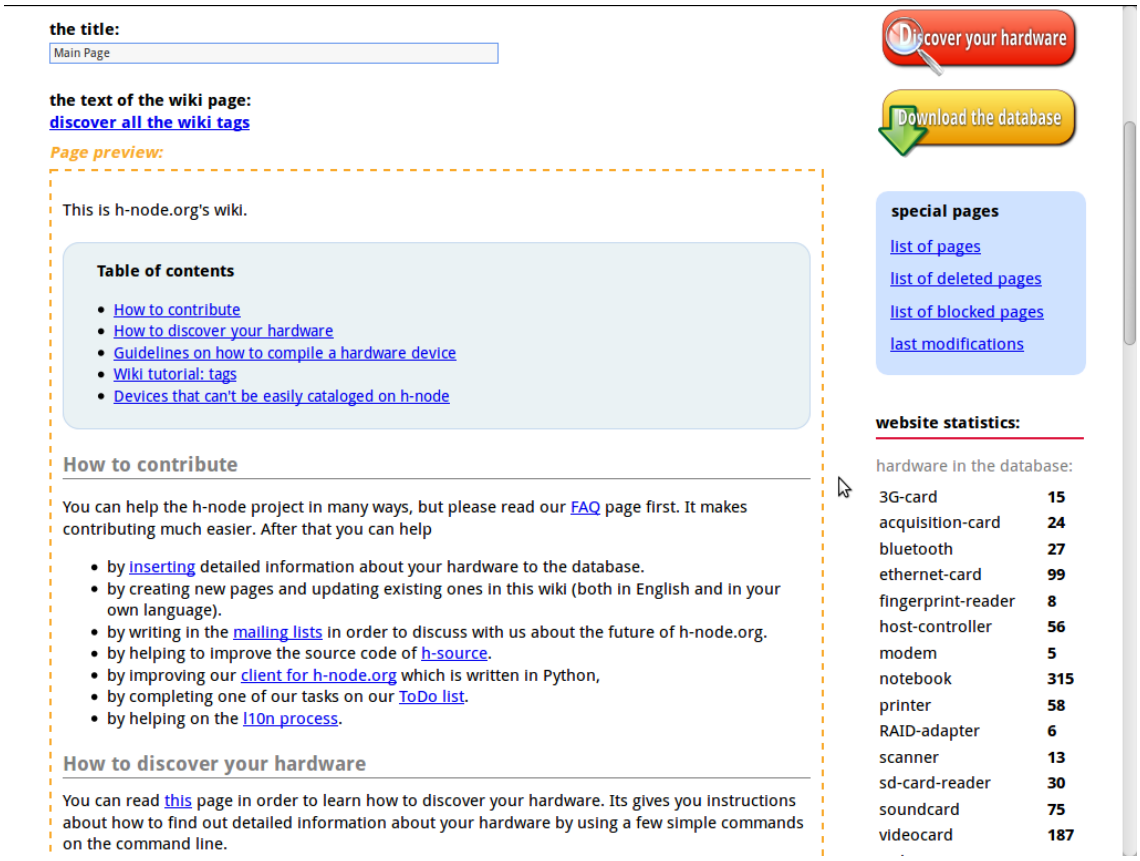


Figure 24. A preview of a wiki page (H-node.org 2013a)

4.3 Discussing with other users

Contributing to h-node does not have to be done alone. There are many places where one can discuss with other users in h-node. The best way to contact other users and the

main developer is the mailing list. It is the easiest way to reach many active users and discuss with them about h-node and its development. The mailing list's subscription page, as shown in Figure 25, can be accessed from the right side panel on any page.

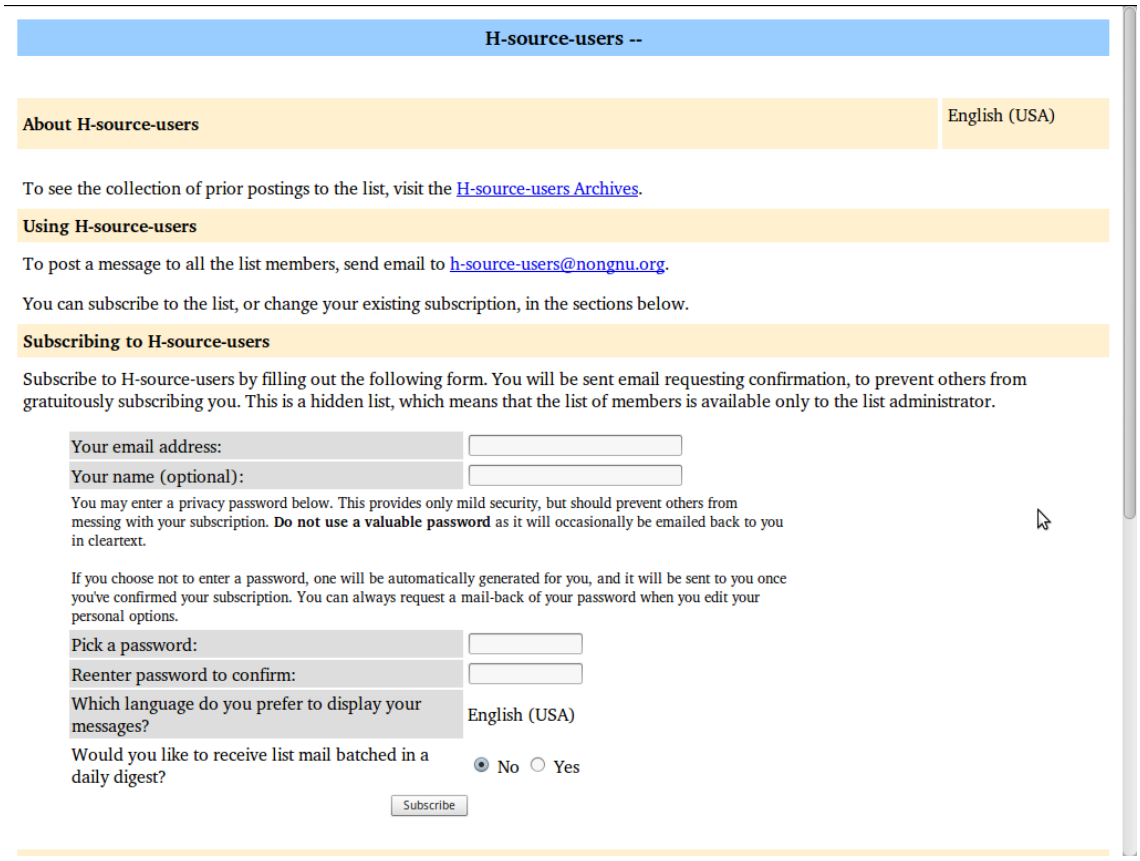


Figure 25. H-node's mailing list's subscription page (H-node.org 2013a)

Every device entry and wiki article has its own talk page in h-node. These pages can be used to leave messages to other users or to participate in a discussion about one specific matter like how some particular issue should be solved for example. These talk pages can be accessed through a *talk* button which can be found on the corresponding page like from a wiki article's page as shown in Figure 13.

The mailing list is still the best way to contact other users. The talk pages are useful mainly to notify users who edited a device entry about possible issues with their edits.

4.4 Solving and reporting issues

As stated in section 3.4 *Issues*, any registered user can create a new issue or participate in discussion with other users about existing ones. Usually users raise an issue when they have found some device whose vendor is not listed in the device vendors' list. Another very usual issue type is a feature request. Figure 26 shows one example of a feature request issue.

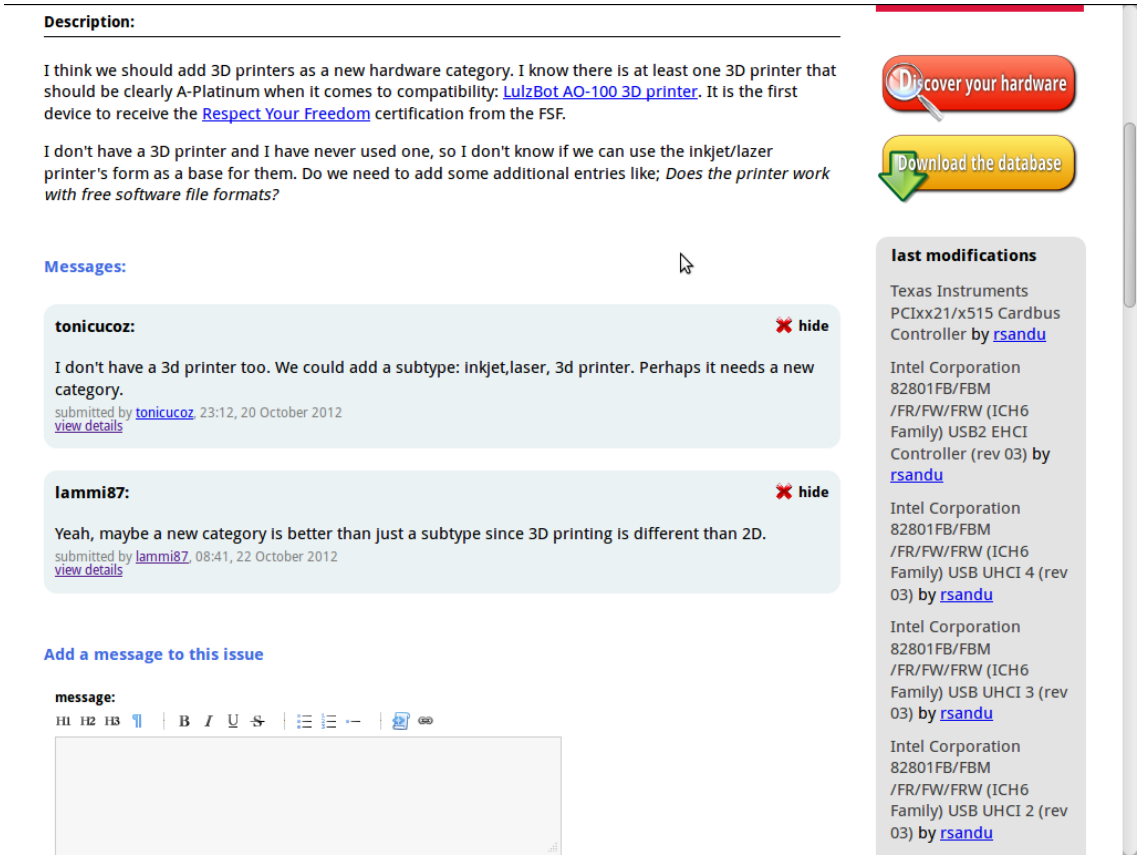


Figure 26. Example of an issue with users discussing it (H-node.org 2013a)

By participating in these discussions, the users can help the main developer in developing h-node. Technically skilled users can also participate directly in the development by solving these issues once a suitable solution has been agreed upon.

4.5 Translating h-node to other languages

English version of h-node has been translated into five different languages. In total, h-node is available in English, French, German, Spanish, Greek and Italian. All these versions of h-node are maintained by users who participate in the h-node translation project. This project has its own wiki article as can be seen in Figure 27.

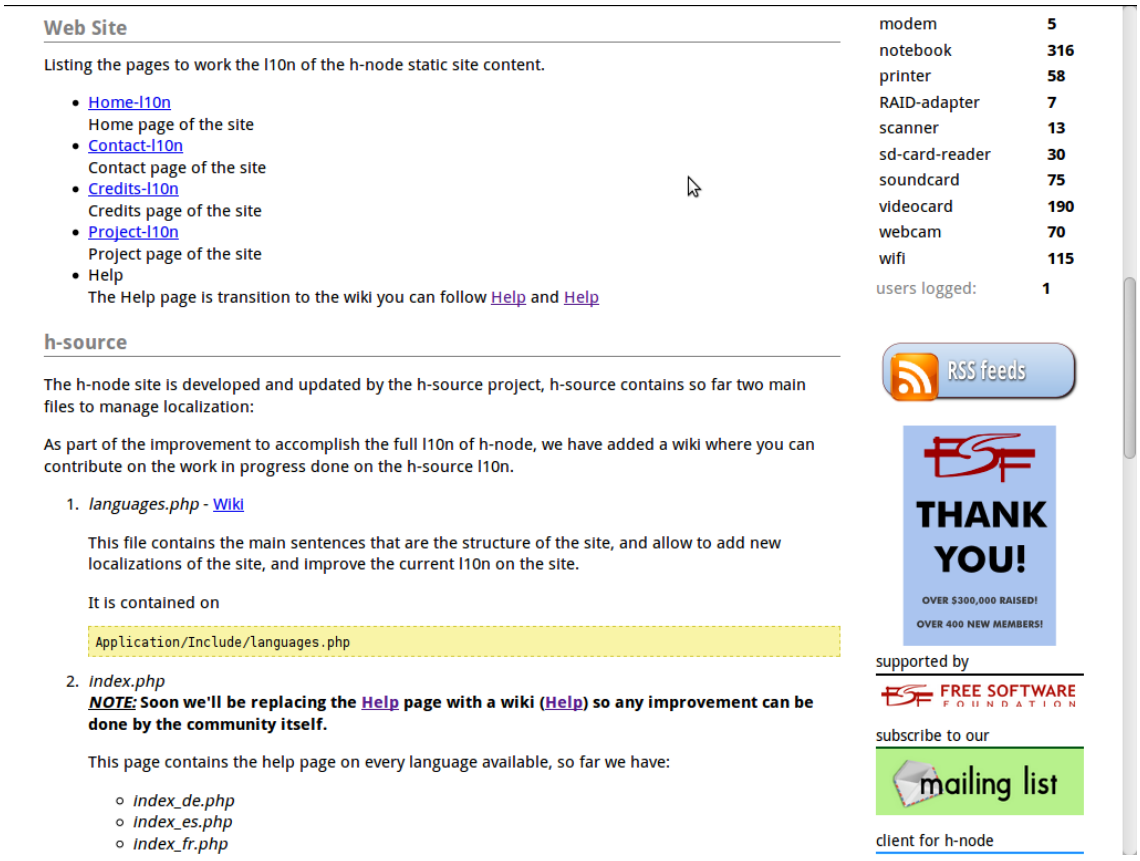


Figure 27. A view of the h-node's translation page (H-node.org 2013a)

There are many ways a single user can participate in the h-node translation project. One can use language related wiki tags while writing in the device's description sections to make separate tabs for each language. In Figure 28, a user has modified the description section to include two language tabs; English and Spanish which each contain the device description in the corresponding language. In this way, every user can view other language versions of the description easily.

Adding language tabs in the description sections is a natural way to translate device descriptions to other languages. The device descriptions are a part of the database itself and, hence, cannot be translated as a part of a web page.

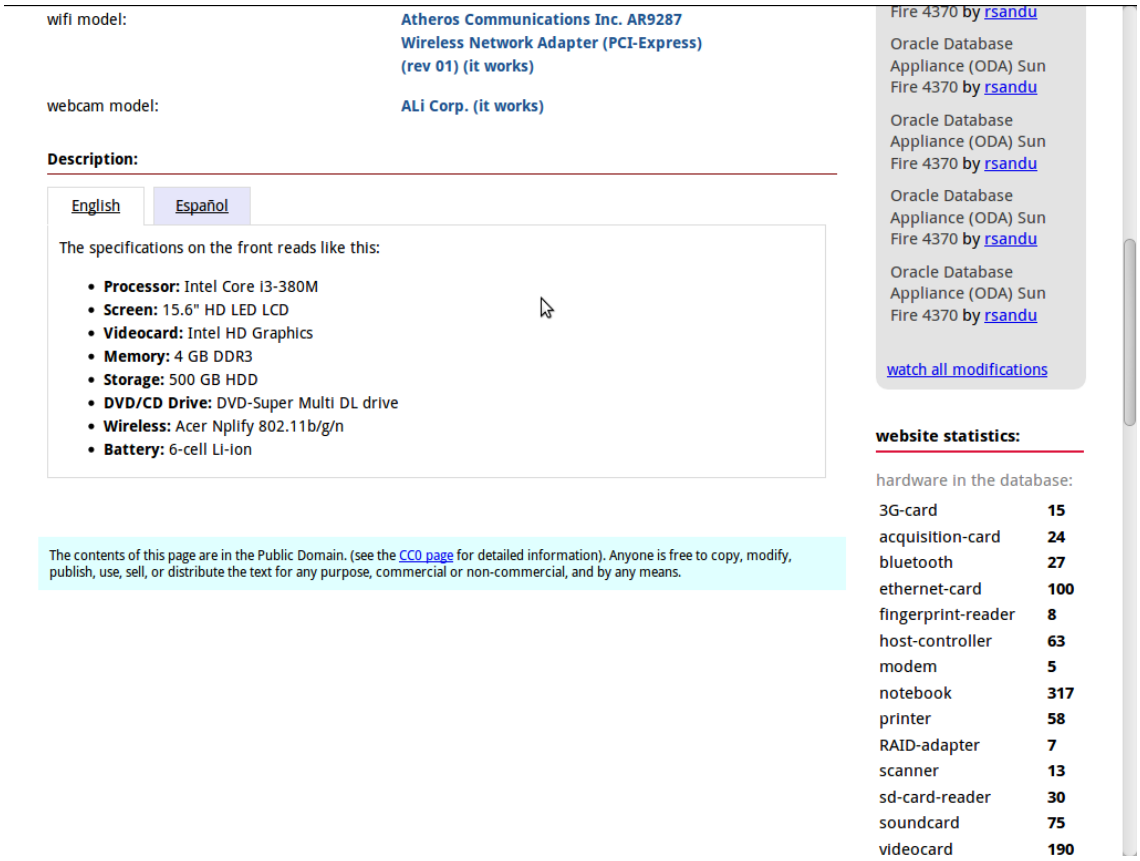


Figure 28. A device's description section with two language tabs (H-node.org 2013a)

Users can translate any wiki article freely to other languages as long as there is a version of h-node for that language. The articles have to be created and maintained separately for each version of h-node. At the bottom of each article, users can put links to other language versions of that article if they exist much like in Wikipedia.

Translating the actual web pages of h-node is handled differently than translating wiki articles or device's descriptions. Most of the pages in h-node consist of key words or phrases which have been listed in one file called *languages.php* in h-node's source code. Downloading the source code and editing this file is one way of translating different web pages of h-node. The other way is to find and edit those files which hold

translatable text itself like home page's notations about h-node's objectives and Free Software. More information can be found in the translation project's wiki article: *l10n process*.

4.6 Acting as moderator and administrator

Active users can gain administrator's and moderator's privileges. This means that they can perform various actions the regular users cannot. These privileges are granted by h-node's main developer when he sees fit.

The main developer can grant either one of the roles of an administrator or moderator to an active user or both of them. If one wishes to become an administrator and/or a moderator, he or she can contact the main developer through the mailing list and ask him.

As a moderator, the user can close or hide issues, wiki articles and users' talk entries which can be found on talk pages of device entries. He or she can do this by using tools which are not visible to regular users on the appropriate pages. In Figure 29, an issue which has been closed and hidden is shown.

This issue was first closed and then hidden from regular users by a moderator who considered it to be spam. On the dark orange panel at the top of the page just below the header, the moderator's controls can be seen. In case of this issue, any moderator can choose to re-open the issue and make it visible to regular users. (Gallo 28 September 2012, e-mail)

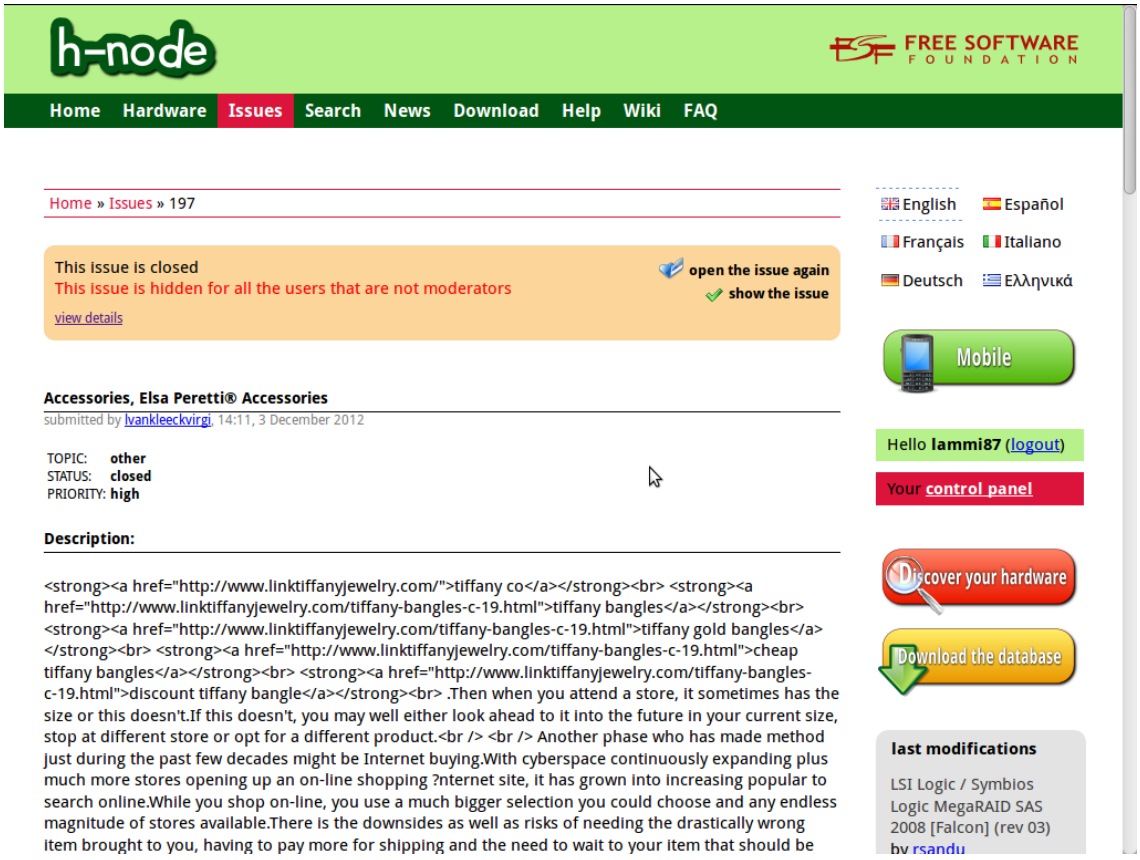


Figure 29. Moderator's controls on a hidden issue page (H-node.org 2013a)

A user who has posted only spam and does not have any contributions listed on his or her user page is not likely to start contributing to h-node and ought to be blocked. A moderator cannot do this but an administrator can.

As an administrator, a user can block other users from their user accounts, delete device entries and approve those device entries which have been made by anonymous users. After the approval, those device entries will be added to the database and made visible for other users. (Gallo 28 September 2012, e-mail)

Most actions carried out by any kind of users can be reversed. For example, a device entry can be returned to any previous state in its history if such an action is preferred. Even moderator's and administrator's actions can be reversed. A hidden issue can be made visible and a blocked user can be unblocked. The only action that cannot be

reversed is a deletion of a device entry carried out by an administrator. (Gallo 28 September 2012, e-mail)

4.7 Developing source code

Most of the development of h-node.org website is done by the h-node's main developer. This does not mean that other users who are skilled in programming couldn't participate in it. There is an article in the h-node's wiki about the development process as shown in Figure 30.

Figure 30. Article about h-node's source code development (H-node.org 2013b)

H-node.org is written in PHP programming language and *Subversion* is the version control system used by the developers. If one wants to participate in the development, he or she needs to download the latest version of the source code, install *Subversion* and

set up and configure a web server like *Apache* so that he or she can test his or her changes before contributing them.

More detailed instructions are available in the wiki article and in the source code's *README* file. However, there is no complete and coherent documentation about the matter yet which can make things difficult for those users who are not used to website development.

5 RESULTS OF DEVELOPMENT

One of the objectives for the author of this final project was to participate in the development of h-node. Since h-node is still a rather young and small project, one the most important ways of contributing to it is extending the hardware database itself. The best way for the author to get his hands on as many different devices as possible was working as a member of a maintenance team in an IT hardware store.

A local IT store called *Lapin ATK-Huolto* proved to be an ideal place. Not only was its location very convenient and its supply of devices to test endless, but its personnel also proved to be nice and supportive of this final project. The author worked there for 20 weeks from 9 September 2012 to 21 January 2013 as part of his practical training period which is a part of his studies. The job was to repair broken computers and assemble new ones. Every time there was a working device available for testing, it was tested and the results were reported to h-node.

Expanding the hardware database was not the only contribution the author made for h-node although it was the most important one. The followings are summarized result of the author's contributions to h-node since he started to work with it in May 2012.

5.1 Expanding the database

During the time at *Lapin ATK-Huolto*, the author had a chance to test a huge amount of different devices. These were mainly laptops and their internal parts but occasionally there was a chance to test some other types of hardware too.

H-node keeps track of every edit made by users but it is still difficult and cumbersome to get a clear picture of what kind of contributions any single user has made in a given time. Since the author wanted to know exactly this, a script was written to do the work

for him. This script, which can be found in the appendixes as *Appendix 1*, uses a few different sources of information to perform the calculations.

The script will calculate how many devices the author has added or edited in h-node and when these contributions were made starting from May 2012 and continuing to March 2013 when writing this final project was started. The script will also illuminate on which types of devices the author has been working with, that is, what kind of devices he has added or edited in h-node.

The script will produce a nice human readable output which is presented in *Appendix 5*. A line chart has been made based on a part of that output which can be seen below as Figure 31. The Y-axis indicates the amount of contributions made and the X-axis indicates the month when they were made.

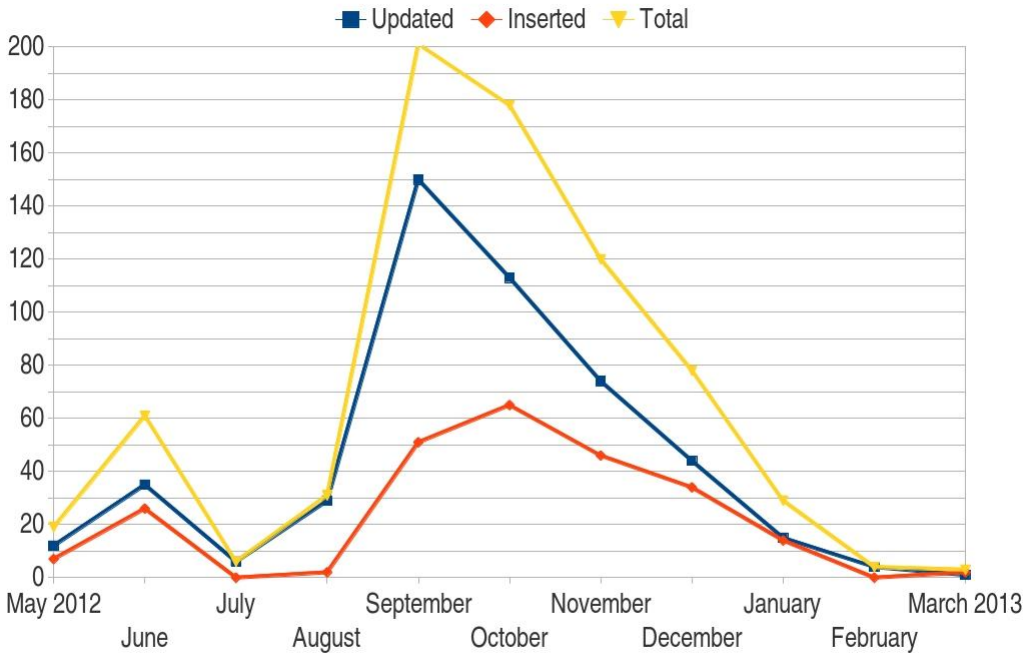


Figure 31. The author's hardware related contributions to h-node

In Figure 31, the red line indicates the amount of new devices which have been added to the database and the blue line indicates the amount of times an existing device was updated. A total value of contributions is indicated by the yellow line.

From the Figure 31, it can be clearly seen that most of the hardware related contributions were made while the author was doing his practical training period between September 2012 and January 2013. The amount of contributions also declined toward the end of the practical training period. This is no surprise since the author had to work with hardware which had been already tested and reported to h-node more often toward the end of the practical training period. There simply were not as much previously untested hardware available anymore.

The line chart in Figure 31 is a fine indicator of how the author's contributions distributed during time but it does not give very definitive figures about how many contributions were made in total. However, the script was made to calculate the specific total figures of both the author's contributions and other users' contributions as a whole, so they can be presented here. Here is a part of the output created by the script:

```
Hardware contributions:
=========================================

New hardware inserted by me:            247
Old hardware updated by me:             483
-----------------------------------------
All my contributions in total:          730
Other's contributions in total:         646
-----------------------------------------
All contributions in total:            1376
```

It can be seen from the script's output that the author managed to make more contributions in quantities than all the other users of h-node in total between May 2012 and March 2013. This was very surprising. The author managed to insert almost 250 new devices to the database and made updates to various existing devices roughly 480 times.

The author wanted to know how many laptops and other devices he had inserted to h-node. So, the script was made to specify the distribution of the contributions by

hardware class and, within each class, specify the type of the contribution. This way, it could be clearly seen what had been achieved, that is, how many devices had been added to each class and which types of device entries had been updated the most. The part of the script's output in question can be seen below:

```
My contributions per hardware class:
================================================

Class                Updated  Inserted     Total
------------------------------------------------
notebook                 204       117       321
videocard                 68        60       128
soundcard                 54        16        70
wifi                      51        10        61
ethernet-card             54        19        73
sd-card-reader            11         6        17
webcam                     6         4        10
bluetooth                  0         0         0
acquisition-card           3         3         6
3G-card                    6         2         8
host-controller            2         1         3
fingerprint-reader         1         0         1
RAID-adapter               0         0         0
modem                      3         0         3
printer                   15         8        23
scanner                    3         1         4
------------------------------------------------
Total:                   481       247       728
```

As it can be seen, most of the contributions are related to laptops as well as internal devices of laptops and desktop computers. The most contributed hardware class is the *Notebooks, netbooks and tablet PCs* category which is listed as *notebook* class in the script's output. More than 100 new laptops is a very nice result. At the time of writing, there are 314 laptops in h-node which include those added by the author. Being able to update existing laptop entries about 200 times is also a rather satisfying result.

It must be noted that some of the updates made can have been made to devices which have been previously inserted to the database. That is why the total amount of laptop related contributions, for example, is bigger than the amount of laptops in the database itself.

It must also be noted that the scripts made by the author may not be 100% accurate. Mistakes may have been made when the scripts were being written but the author believes that the results they give are reliable and should be considered being roughly correct. However, there are some inaccuracies in the results. One can see, for example, that the total amount of contributions as presented in the two parts of the script's output above are not completely the same. In the first part of the output, it is stated to be 730 but in the latter it is 728.

5.2 Figuring out distribution of laptop's compatibility levels

All devices in h-node have a specific compatibility level with Free Software depending on the type of the hardware. Some devices like Wi-Fi cards are simple to categorize. They either work or do not work. Laptops, on the other hand, have multiple very specific compatibility levels.

The compatibility of laptops depends on the compatibility of their internal devices. Thus, multiple compatibility levels are necessary to categorize laptops in h-node. The laptops' compatibility levels are as follows: (H-node.org 2013. Retrieved 28 March 2013)

- A-Platinum (Everything works perfectly)
- B-Gold (Everything work, but not with full performance)
- C-Silver (One device is not supported)
- D-Bronze (More than one device is not supported)
- E-Garbage. (Whole device is not compatible with Free Software)

With these compatibility levels, it is easy to categorize different laptops. The B-Gold level refers to laptops which work fine but cannot use all the features the hardware would allow. This is usually with laptops which have a video card which is only partly supported. The card itself works but cannot utilize its 3D acceleration features.

At the moment of writing, there was no easy way to tell how well the laptops listed in h-node's database are supported with Free Software. So, another script was written to figure this out. This script is listed in the appendixes as *Appendix 4* and its output in *Appendix 8*. A pie chart has been made with *LibreOffice Calc* based on the script's output. It can be seen in Figure 32.

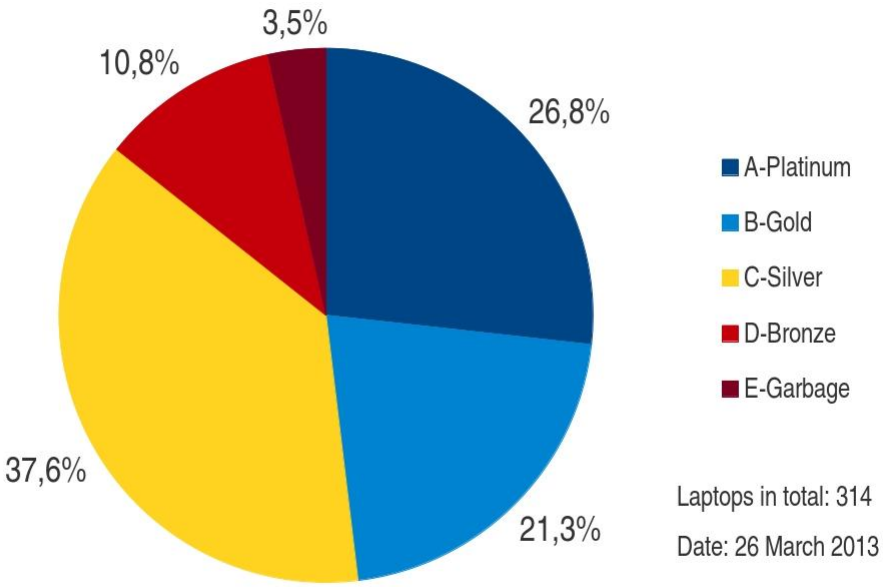


Figure 32. Distribution of laptop's compatibility levels in h-node

It can be seen that roughly half the laptops in h-node are really well supported with completely free GNU/Linux distributions and, hence, with Free Software. Most of the laptops which have some incompatibilities are also quite well supported since most of them have only one incompatible device like a non-functioning Wi-Fi card. Wi-Fi cards are usually the one device which is incompatible and they are quite easy to replace with a freedom friendly card in most cases.

Having this good compatibility is good news for the Free Software community and the entire GNU/Linux community as a whole, that is, even to the users of non-free

GNU/Linux distributions. But it must be noted that the sample of laptops used to make this chart is rather small. It is only 314. Thus, this result may not be reliable enough to give a good general picture of all the commercially available laptop's compatibility with Free Software but this is the best that can be done at the moment. The comparability of h-node's laptops compatibility and any laptop's compatibility in general will improve in time when more laptops are tested and reported to h-node.

5.3 Actions as administrator and moderator

The author was granted the administrator's and moderator's privileges while doing his practical training period. This was an unexpected but a pleasant surprise. A new way to contribute was gained. Since then, the author has hidden issues on the Issues page which were clearly spam, closed solved issues and blocked users from their accounts if they have written only spam.

These actions are not numerous since the focus was mainly on extending the hardware database. Nonetheless, two other scripts were written to calculate and present these contributions. The first script is about author's actions as an administrator. It can be found in the *Appendix 2* and its human readable output in *Appendix 6*. The second script is the equivalent of author's actions as a moderator. It can be found in the *Appendix 3* and its output in *Appendix 7*.

5.4 Writing the FAQ

The author felt that h-node was lacking a proper introduction to new users who have just found the site and are wondering what it is really about. So, he decided to write an FAQ where the most crucial questions the new users usually have are answered. The FAQ page can be seen in Figure 14.

The answers were written to give the new users a good general picture of each topic. If a more detailed wiki article about the matter existed, the user is guided to it with a link. Writing the FAQ may be the most important author's contribution after extending the hardware supply in the database.

5.5 Issues

The author has reported many issues in h-node's Issues page about various subjects. Some have been about small bugs while others were about some new feature which was suggested to the main developer to be implemented. Some of the issues which have been reported have been solved while others are either still being worked on or have been deemed unnecessary after discussion.

One example of an issue reported by the author is issue #167. Once when adding a new device to the database, the number of allowed characters in the model name field was not enough to fit in the whole name of the device. The number of allowed characters was increased after this issue was reported.

6 FURTHER DEVELOPMENT

H-node is still in development. The main developer has done much to make h-node what it is today and the user community has also done a great job in contributing to the development process. Still, many new features are waiting to be added and others to be improved.

Here are some points that the author would like to suggest for further development. Some are small features while others could serve as a base for a final project, even.

6.1 Automation

The process of inserting a new device to the database could use some automation. Firstly, h-node's hardware insertion form, as seen in Figure 20, could get some of the required hardware details from the *lspci* command's output which is used in the search page in the beginning of the process, as can be seen in Figure 18, and use them to fill in some of the fields in the hardware insertion form.

This would reduce the amount of work each user has to do and the possibility of a human error providing that the automation process is reliable. Among the details which could be get automatically would be the device's name and the device's and its vendor's unique ID numbers, for example. Of course, the user could always review and change any automatically filled in information.

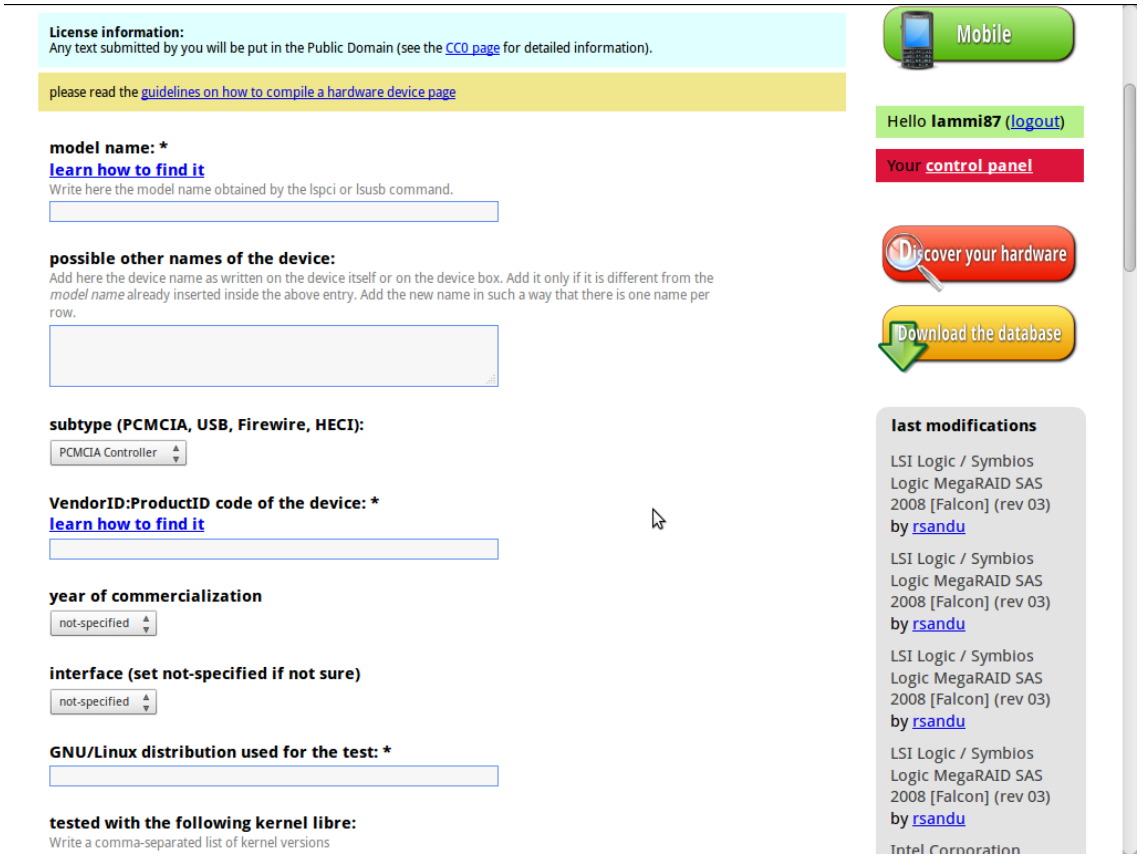


Figure 33. The hardware insertion form (H-node.org 2013b)

Secondly, the *Insert* link below each device entry in the list of hardware which should be added to the database presented after comparing one's hardware to the database, as shown in Figure 19, could take the user directly to the corresponding hardware insertion form, as shown in Figure 33. At the moment of writing, it takes the user to the corresponding hardware category page, as shown in Figure 7, where the user must click another link to get to the insertion form. There is no need for this one extra click.

These two features would reduce the amount of effort the users have to make just to be able to contribute. This is not so big of a problem if one contributes only a few devices occasionally but proves to be a bit annoying if one adds a lot of devices at once or contributes frequently.

6.2 H-client

H-client is a desktop application which, when run, collects as much hardware information of the user's computer as possible and reports it automatically to h-node. Its aim is to ease the contribution process to h-node.

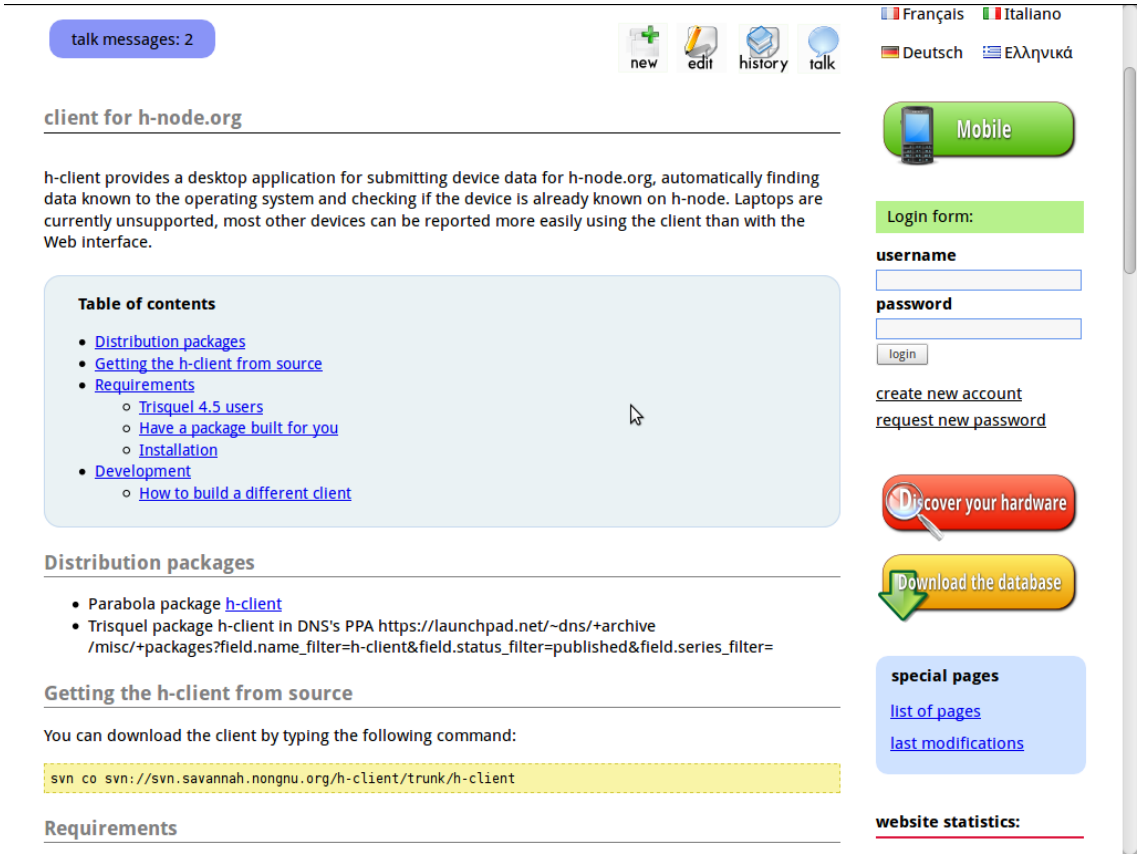


Figure 34. H-client project page in the wiki (H-node.org 2013b)

H-client is developed much the same way as h-node.org. It is developed by volunteers, its source code is available for download, as can be seen in Figure 34, and anyone can participate in developing it. The version control system used by the developers is *Subversion*.

H-client is still in development. Most of the devices can be reported to h-node by using the application according to its development page. Laptops are not currently supported.

Further improving this application could prove a good subject for a final project or a regular software development project.

6.3 Documentation

The level of documentation varies greatly in h-node. Some is well written, consistent and detailed while some is inadequate and difficult to use for a non-technical or inexperienced user.

Figure 35. Documentation about developing h-node's source code (H-node.org 2013b)

A good example of an inadequate documentation is the h-node's source code development project page, as shown in Figure 35. A user who is not familiar with website development cannot participate easily in the source code development project. The documentation explains how the user can get the source code but it does not clarify

how he or she is supposed to test his or her changes before committing them or how the committing process works.

The committing process is illuminated a bit in a separate source. There is a README file in the root of the h-node's source code. According to the file, the user should use a web server such as Apache in order to use his or her own version of h-node.

This means that the user must use a web server to view his or her changed version of h-node so that the user can verify that their changes work correctly before committing them. The file does not, however, clearly specify how the user could do this. An inexperienced user who just wishes to correct a simple typo on a static portion of h-node is likely to give up at this point. Short and simplified instructions of the whole process would help a lot.

H-node needs coherent documentation so that every user regardless of his or her level of skillfulness can participate in the development. Every new contributor is welcome. Improving the documentation is one of the most important things the h-node users can do to improve the state of the project.

6.4 Statistics

H-node can provide the Free Software community clear figures about GNU/Linux's hardware compatibility with Free Software. All the necessary information is in the database. All that needs to be done is to process this information and present it to the community in a usable form. A new Statistics section would be ideal for this purpose.

The Statistics section should have figures and charts about hardware compatibility. At least the most important hardware categories like laptops, Wi-Fi cards, video cards and printers should be covered. The charts should be simple and easy to understand like the pie chart presented in Figure 32.

The process of generating the information would be similar to what has been done with the scripts used in this final project. Only this time the information would be always up to date since it is fetched right from the database itself and processed automatically whenever a user moves to the Statistics section. The charts would be generated from scratch each time. One example is shown in Figure 36.

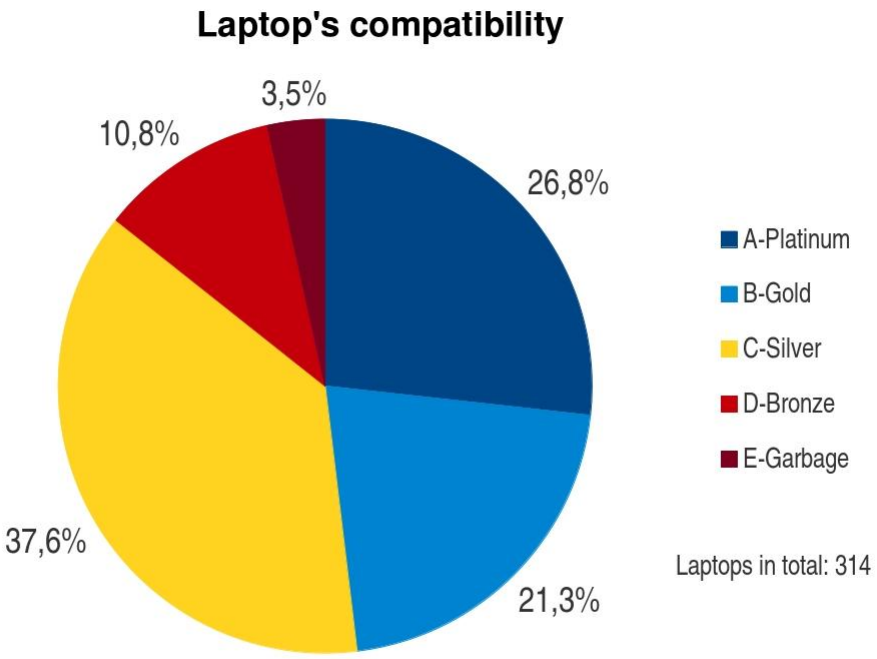


Figure 36. Example chart for the Statistics section

The Statistics section would not be useful only to the Free Software community. The whole GNU/Linux community could also use these figures and charts because most GNU/Linux distributions have the same Free Software drivers and firmware in the Linux kernel as the free distributions do as explained in section 2.7 *H-node*.

The Free Software community needs the hardware compatibility information which h-node can provide. Now, the state of hardware compatibility is based on user's experience. After implementing the Statistics section, it would be based on definite figures.

At the moment of writing, the supply of devices was too small to give a reliable general picture of the GNU/Linux's hardware compatibility. The information will not be reliable until more devices are reported to the database. It will improve with time. However, it would be useful right away for the users of h-node in monitoring the state of the database.

7 CONCLUSIONS

This final project had two objectives. The first was to introduce new users to h-node by writing and distributing this project. The second was to participate in the development of the h-node project by extending the hardware database and contributing to the h-node.org website.

The first objective was partly reached by completing this project. Distributing this final project to new users will start after this project is published. It will be introduced to various Free Software organizations and groups around the world, such as the Free Software Foundation and its sister organizations and the user communities of the completely free GNU/Linux distributions. With their help, this project can reach many more new users than the author could possibly reach by himself.

The second objective was reached and the results are encouraging. H-node's hardware database was extended greatly during this project. H-node.org website was also improved by submitting bug reports and feature requests, removing spam and blocking spam posting users and writing the FAQ section.

H-node's flaws were also discussed in this final project. It is important that the h-node community recognizes these flaws and can improve h-node to surpass them. This project helps the community in this task by identifying some of the flaws and suggesting new useful features to be implemented which can benefit the h-node project greatly.

This final project managed to support the h-node project also by introducing the Free Software philosophy, GNU/Linux operating systems and the history behind them to those users who are not familiar with them. This helps them to understand why Free Software is important and invites them to join the community. H-node project benefits from a lively community and the community benefits from a growing h-node project.

The author of this final project is very happy with the achieved results and feels confident that the h-node project will continue to develop into an important part of the Free Software community.

8 REFERENCES

Ewing, Larry & Budig, Simon & Gerwinski, Anja. 1996. Tux. Retrieved 18 March 2013 < http://commons.wikimedia.org/wiki/File:Tux.svg>
Gallo, Antonio. 2010. H-node logo. Retrieved 18 March 2013 < http://www.h-node.org>
Gallo, Antonio, Main developer, H-node. h-node moderator and administrator. E-mail tonicucoz@yahoo.com 28 September 2012.
Gallo, Antonio, Main developer, H-node. Questions for final thesis. E-mail tonicucoz@yahoo.com 6 February 2013.
Gallo, Antonio, Main developer, H-node. Questions for final thesis. E-mail tonicucoz@yahoo.com 27 March 2013.
Gallo, Antonio, Main developer, H-node. Questions for final thesis. E-mail tonicucoz@yahoo.com 15 April 2013.
GNU Operating System 2013. Free GNU/Linux distributions. Retrieved 18 March 2013 <http://www.gnu.org/distros/free-distros.html>
H-node.org 2013a. Retrieved 28 March 2013 <http://www.h-node.org>
H-node.org 2013b. Retrieved 1 May 2013 <http://www.h-node.org>
Nemeth, Evi & Snyder Garth & Hein, Trent 2008. Linux Administration Handbook. Massachusetts: Prentice Hall
Pérez, Rubén Rodríguez. 2009. Freedo. Retrieved 18 March 2013 < http://en.wikipedia.org/wiki/File:Freedo.svg>
Stallman, Richard 1983. Initial Announcement. Retrieved March 2 2013 < http://www.gnu.org/gnu/initial-announcement.html>
Stallman, Richard 2010. Free Software Free Society. USA: SoHo Books
Suvasa, Etienne & Gerwinsky, Peter. GNU Head. Retrieved 18 March 2013 <http://www.gnu.org/graphics/gerwinski-gnu-head.png>
Trisquel GNU/Linux 2013. Retrieved 17 April 2013 <http://www.trisquel.info>
Trisquel GNU/Linux 6.0 live-CD 2013. Retrieved 2 April 2013 <http://www.trisquel.info>
Torvalds, Linus 1991. RELEASE NOTES FOR LINUX v0.12. Retrieved March 2 2013 <https://www.kernel.org/pub/linux/kernel/Historic/old-versions/RELNOTES-0.12>
Williams, Sam Stallman, Richard 2010. Free as in Freedom (2.0): Richard Stallman and the Free Software Revolution. Boston: Free Software Foundation

LIST OF APPENDICES

Appendix 1.	HARDWARE CONTRIBUTIONS SCRIPT
Appendix 2.	ADMIN CONTRIBUTIONS SCRIPT
Appendix 3.	MODERATOR CONTRIBUTIONS SCRIPT
Appendix 4.	NETBOOK COMPATIBILITY SCRIPT
Appendix 5.	OUTPUT OF HARDWARE CONTRIBUTIONS SCRIPT
Appendix 6.	OUTPUT OF ADMIN CONTRIBUTIONS SCRIPT
Appendix 7.	OUTPUT OF MODERATOR CONTRIBUTIONS SCRIPT
Appendix 8.	OUTPUT OF NETBOOK COMPATIBILITY SCRIPT

HARDWARE CONTRIBUTIONS SCRIPT

```
#!/bin/bash
#
# hardware-contributions.sh
#
#   Copyright (C) 2013  Matti Lammi
#
#   This program is free software: you can redistribute it and/or
#   modify it under the terms of the GNU General Public License as
#   published by the Free Software Foundation, either version 3 of the
#   License, or (at your option) any later version.
#
#   This program is distributed in the hope that it will be useful,
#   but WITHOUT ANY WARRANTY; without even the implied warranty of
#   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#   GNU General Public License for more details.
#
#   You should have received a copy of the GNU General Public License
#   along with this program. If not, see
#   <http://www.gnu.org/licenses/>.
#
#
# This script will produce information about my (lammi87) hardware
# related contributions to h-node in human readable form. It will use
# tools commonly found in GNU/Linux operating system distributions
# and two source files to do this. Results will be saved to a file
# and also displayed immediately.
#
# The source files are text files consisting on several one line
# long entries of data. The data is copied from two of h-node's
# lists of hardware related contributions. The first one [1] is a
# list of actions carried out by all users. The second one [2] is a
# different kind of list containing my actions alone. Both of these
# lists can be accessed by anyone:
#
# [1] http://h-node.org/special/modifications/en
# [2] http://h-node.org/meet/hardware/en/lammi87
#
# The first list [1] is rather long, so I have included only the
# entries starting from the day when I first started reporting
# devices to h-node continuing to the present day in order to narrow
# it down.
#
# I had to use two separate sources because neither one alone did not
# include all the information I wanted. I also had to make sure that
# the source files I made based on those lists contained only the
# data I needed and nothing else. Anything extra like empty lines or
# comments might have an effect to the results since this script will
# search and count lines in various ways.
#
# I have used many line breaks in this script so that I can keep the
# layout clean when adding this script to my final thesis as an
# appendix.
```

```
#
# Please note that the output of this script may not be 100% accurate.
# I think, however, that the results are accurate enough.
#
# SOURCE FILES
#
# [1] File name:            source-hardware-1
#     Date for first entry:  3 May   2012
#     Date for last entry:  26 March 2013
#
# [2] File name:            source-hardware-2
#     Date for first entry:  3 May   2012
#     Date for last entry:  26 March 2013
#


### Print instructions.
###
# "echo" program's -e option allows special characters like
# \n to be implemented.
# \n = new line
# \t = tab
echo -e "H-node script: Hardware contributions\n"
echo -e "First argument:\t\t   Source file [1]"
echo -e "Second argument: \t   Source file [2]"
echo -e "Third argument:  \t   Output file\n"


### Quit if source file [1] is not given.
###
# Check if first argument is null.
#  ! = not
# -n = not null
if [ ! -n "$1" ]
then
echo "ERROR: No source [1] given. Read source code for more info."
exit 1
fi

# Check if source [1] does not exist.
#  ! = not
# -f = file exists
if [ ! -f "$1" ]
then
echo "ERROR: Source [1] not found. Read source code for more info."
exit 1
fi


### Quit if source file [2] is not given.
###
# Check if second argument is null.
#  ! = not
# -n = not null
if [ ! -n "$2" ]
then
echo "ERROR: No source [2] given. Read source code for more info."
```

```
exit 1
fi

# Check if source [2] does not exist.
#  ! = not
# -f = file exists
if [ ! -f "$2" ]
then
echo "ERROR: Source [2] not found. Read source code for more info."
exit 1
fi


### Quit if output file is not given.
###
# Check if third argument is null.
#  ! = not
# -n = not null
if [ ! -n "$3" ]
then
echo "ERROR: No output file given. Read source code for more info."
exit 1
fi


### Create output file.
###
# \n = new line
touch $3
echo -e "Output file created.\n\n"


### Define table format and lines
###
# I will print the data processed from the source files into easy
# to read tables. I use "printf" instead of "echo" because it has
# more options to format the output. I'll define a format for the
# tables so I can easily use it to print several tables and keep
# their formatting unanimous. The format is a 35 character long space
# followed by eight character long space and a new line character.
# I added the new line character because "printf" does not include
# it to its output by default. The first space will align characters
# printed into it to the left. The other space will align its content
# to right.
# %s = string
# NN = reserved character space in %NNs
#  - = alignment to left
# \n = new line
TABLE_FORMAT="%-40s%8s\n"

# I'll be using two kinds of "lines" of characters to make the
# output look more readable. I'll define them here.
THICK_LINE="================================================="
THIN_LINE="-------------------------------------------------"


### Find out my hardware contributions
```

```
###
# %s = string
# \n = new line
echo "Hardware contributions:" > $3
printf "%s\n\n" $THICK_LINE >> $3

# Find out how many new pieces of hardware I have added:
#
# Inspect all lines of the source file [1], get only those which
# mention me and the word "inserted", count them and print the
# result.
# NUM=$() = Run code defined in brackets and save result to NUM.
# Printf's syntax: printf <format> <argument>
NUM=$(cat $1 | grep lammi87 | grep inserted | wc -l)
printf $TABLE_FORMAT "New hardware inserted by me:" $NUM  >> $3

# Find out how many updates I have made to existing hardware
# entries:
#
# Inspect all lines of the source file [1], get only those which
# mention me and the word "updated", count them and print the
# result.
# NUM=$() = Run code defined in brackets and save result to NUM.
# Printf's syntax: printf <format> <argument>
NUM=$(cat $1 | grep lammi87 | grep updated | wc -l)
printf $TABLE_FORMAT "Old hardware updated by me:" $NUM >> $3

# Find out how many hardware related contributions I have made
# in total:
#
# Inspect all lines of the source file [1], get only those which
# mention me, count them and print the result.
# Printf's syntax: printf <format> <argument>
# NUM=$() = Run code defined in brackets and save result to NUM.
#      %s = string
#      \n = new line
printf "%s\n" $THIN_LINE >> $3
NUM=$(cat $1 | grep lammi87 | wc -l)
printf $TABLE_FORMAT "All my contributions in total:" $NUM >> $3

# Find out how many hardware related contributions others
# have made in total:
#
# Inspect all lines of the source file [1], get only those which
# do not mention me, count them and print the result.
# NUM=$() = Run code defined in brackets and save result to NUM.
# Printf's syntax: printf <format> <argument>
NUM=$(cat $1 | grep -v lammi87 | wc -l)
printf $TABLE_FORMAT "Other's contributions in total:"  $NUM >> $3

# Find out how many hardware related contributions there
# are in total:
#
# Inspect all lines of the source file [1], count them and print
# the result.
# Printf's syntax: printf <format> <argument>
# NUM=$() = Run code defined in brackets and save result to NUM.
```

```
#      %s = string
#      \n = new line
printf "%s\n" $THIN_LINE >> $3
NUM=$(cat $1 | wc -l)
printf $TABLE_FORMAT "All contributions in total:"  $NUM >> $3


### Find out how many contributions I did per month.
###
# For each year, write an entry to output file. For each
# month within that year, write an entry to output file and separately
# count how many devices I have updated, inserted and contributed in
# total (Each year is searched from the end of each line from the
# source file [1] due to "$" after the YEAR variable). Also, count
# total values for each three categories of contributions for each
# year and print it.
# Printf's syntax: printf <format> <argument>
#      %s = string
#       - = alignment to left
#      \n = new line
#       \ = line break
# NUM=$() = Run code defined in brackets and save result to NUM.
# The "let" command enables arithmetic operations on variables.

# I'll be using another kind of table format for these month tables.
NEW_TABLE_FORMAT="%-18s%10s%10s%10s\n"

echo -e "\n\nMy contributions per month:" >> $3
printf "%s\n" $THICK_LINE >> $3

for YEAR in 2012 2013
do

   printf "\n$NEW_TABLE_FORMAT" "Year: $YEAR" "Updated" \
          "Inserted" "Total" >> $3
   printf "%s\n" $THIN_LINE >> $3

      for MONTH in January February March April May June July August \
      September October November December
      do

         UPDATED=$(cat $1 | grep lammi87 | grep updated |\
                   grep $YEAR$ | grep $MONTH | wc -l)

         INSERTED=$(cat $1 | grep lammi87 | grep inserted |\
                    grep $YEAR$ | grep $MONTH | wc -l)

         TOTAL=$(cat $1 | grep lammi87 | grep $YEAR$ |\
                 grep $MONTH | wc -l)
         printf $NEW_TABLE_FORMAT $MONTH $UPDATED $INSERTED\
                $TOTAL >> $3
         let "UPDATED_IN_MONTH=$UPDATED_IN_MONTH+$UPDATED"
         let "INSERTED_IN_MONTH=$INSERTED_IN_MONTH+$INSERTED"
         let "TOTAL_IN_MONTH=$TOTAL_IN_MONTH+$TOTAL"

      done
```

```
   printf "%s\n" $THIN_LINE >> $3
   printf $NEW_TABLE_FORMAT "Total:" $UPDATED_IN_MONTH\
          $INSERTED_IN_MONTH $TOTAL_IN_MONTH >> $3

   UPDATED=0
   INSERTED=0
   TOTAL=0
   UPDATED_IN_MONTH=0
   INSERTED_IN_MONTH=0
   TOTAL_IN_MONTH=0

done


### Find out my contributions per hardware class per device.
###
# For each hardware class, get a device name from source file [2] and
# count how many times it can be found in the source file [1] with my
# user name in it. Then count how many update and insert type of
# contributions I have made. Also count the total number of
# contributions. Do this for each device in each hardware class.
# Count total values of all contribution types. Print the results.
#
# sed is a line editor. I use it to cut out unnecessary stuff from
# what I get from source file [2].
# Printf's syntax: printf <format> <argument>
#      \n = new line
#      %s = string
#       \ = line break
#      -n = not null (if statement)
# NUM=$() = Run code defined in brackets and save result to NUM.
echo -e "\n\nMy contributions per hardware class:" >> $3
printf "%s\n\n" $THICK_LINE >> $3

printf "$NEW_TABLE_FORMAT" "Class" "Updated" \
        "Inserted" "Total" >> $3
printf "%s\n" $THIN_LINE >> $3


for CLASS in notebook videocard soundcard wifi ethernet-card \
sd-card-reader webcam bluetooth acquisition-card 3G-card \
host-controller fingerprint-reader RAID-adapter modem printer \
scanner
do

   DEVICE_NAMES=$( sed -rn 's/^'"$CLASS"'\s+(.+)\s+/\1/p' "$2" )

   if [ -n "$DEVICE_NAMES" ]
   then

      UPDATED=$(grep -F "$DEVICE_NAMES" $1 | grep "lammi87" |\
          grep -c updated )
      INSERTED=$(grep -F "$DEVICE_NAMES" $1 | grep "lammi87" |\
          grep -c inserted )
      TOTAL=$(grep -F "$DEVICE_NAMES" $1 | grep -c "lammi87" )

   else
```

```
      UPDATED=0
      INSERTED=0
      TOTAL=0

   fi

   printf $NEW_TABLE_FORMAT $CLASS $UPDATED $INSERTED $TOTAL >> $3
   let "FINAL_UPDATED=$FINAL_UPDATED+$UPDATED"
   let "FINAL_INSERTED=$FINAL_INSERTED+$INSERTED"
   let "FINAL_TOTAL=$FINAL_TOTAL+$TOTAL"

done

printf "%s\n" $THIN_LINE >> $3
printf $NEW_TABLE_FORMAT "Total:" $FINAL_UPDATED\
       $FINAL_INSERTED $FINAL_TOTAL >> $3


### Show results.
###
cat $3


### Quit
###
# \n = new line
echo -e "\nAll done.\n"
exit 0
```

ADMIN CONTRIBUTIONS SCRIPT

```
#!/bin/bash
#
# admin-contributions.sh
#
#   Copyright (C) 2013  Matti Lammi
#
#   This program is free software: you can redistribute it and/or
#   modify it under the terms of the GNU General Public License as
#   published by the Free Software Foundation, either version 3 of the
#   License, or (at your option) any later version.
#
#   This program is distributed in the hope that it will be useful,
#   but WITHOUT ANY WARRANTY; without even the implied warranty of
#   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#   GNU General Public License for more details.
#
#   You should have received a copy of the GNU General Public License
#   along with this program. If not, see
#   <http://www.gnu.org/licenses/>.
#
#
# This script will produce information about my (lammi87)
# contributions to h-node as an administrator in human readable form.
# It will use tools commonly found in GNU/Linux operating system
# distributions and a source file to do this. Results will be saved
# to a file and also displayed immediately.
#
# The source file is a text file consisting on several one line long
# entries of data. The data is copied from h-node's list of actions
# carried out by administrators (available only to administrators and
# h-node's main developer) where a new entry is made every time a
# user who has administrational rights carries out an action which
# only administrators can do.
#
# This list is rather long, so I have narrowed it down by filtering
# out every entry about other administrator's actions using a search
# filter found in the list's page, thus keeping my source file short.
#
# I have used many line breaks in this script so that I can keep the
# layout clean when adding this script to my final thesis as an
# appendix.
#
# Please note that the output of this script may not be 100% accurate.
# I think, however, that the results are accurate enough.
#
# SOURCE FILE
# File name:                    source-admin
# Date of first entry:           6 October 2012
# Date of last entry:           22 March   2013
#
```

```
### Print instructions.
###
# Echo program's -e option allows special characters like
# \n to be implemented.
# \n = new line
# \t = tab
echo -e "H-node script: Actions as administrator\n"
echo -e "First argument:\t\t   Source file"
echo -e "Second argument: \t   Output file\n"


### Quit if source file is not found.
###
# Check if first argument is null.
# !  = not
# -n = not null
if [ ! -n "$1" ]
then
echo "ERROR: No source given. Read source code for more info."
exit 1
fi

# Check if source does not exist.
# !  = not
# -f = file exists
if [ ! -f "$1" ]
then
echo "ERROR: Source not found. Read source code for more info."
exit 1
fi


### Quit if output file is not given.
###
# Check if second argument is null.
# !  = not
# -n = not null
if [ ! -n "$2" ]
then
echo "ERROR: No output file given. Read source code for more info."
exit 1
fi


### Create output file.
###
# \n = new line
touch $2
echo -e "Output file created.\n\n"


### Define table format and lines
###
# I will print the data processed from the source files into easy
# to read tables. I use "printf" instead of "echo" because it has
# more options to format the output. I'll define a format for the
# tables so I can easily use it to print several tables and keep
```

```
# their formatting unanimous. The format is a 35 character long space
# followed by eight character long space and a new line character.
# I added the new line character because "printf" does not include
# it to its output by default. The first space will align characters
# printed into it to the left. The other space will align its content
# to right.
# %s = string
# NN = reserved character space in %NNs
#  - = alignment to left
# \n = new line
TABLE_FORMAT="%-35s%8s\n"

# I'll be using two kinds of "lines" of characters to make the
# output look more readable. I'll define them here.
THICK_LINE="============================================="
THIN_LINE="---------------------------------------------"


### Find out my actions as an admin:
###
# %s = string
# /n = new line
echo "Actions as administrator:" > $2
printf "%s\n\n" $THICK_LINE >> $2

# Find out how many users I have blocked:
#
# Inspect all lines of the source file, get only those which have
# my username followed by one or more spaces and the word "block",
# count them and print the result.
# NUM=$() = Run code defined in brackets and save result to NUM.
#      \s = any white space character (regex)
#       * = preceding character zero or more times (regex)
NUM=$(cat $1 | grep "lammi87\s*block" | wc -l)
printf $TABLE_FORMAT "Users blocked by me:" $NUM >> $2

# Find out how many devices I have deleted:
#
# Inspect all lines of the source file, get only those which
# have my username followed by one or more spaces and the words
# "perm deleted", count them and print the result.
# NUM=$() = Run code defined in brackets and save result to NUM.
#      \s = any white space character (regex)
#       * = preceding character zero or more times (regex)
NUM=$(cat $1 | grep "lammi87\s*perm deleted" | wc -l)
printf $TABLE_FORMAT "Devices deleted by me:" $NUM >> $2

# Find out how many devices I have hidden:
#
# Inspect all lines of the source file, get only those which
# have my username followed by one or more spaces and the word
# "hide", count them and print the result.
# NUM=$() = Run code defined in brackets and save result to NUM.
#      \s = any white space character (regex)
#       * = preceding character zero or more times (regex)
NUM=$(cat $1 | grep "lammi87\s*hide" | wc -l)
printf $TABLE_FORMAT "Devices hidden by me:" $NUM >> $2
```

```
# Find out how many actions I have made in total:
#
# Inspect all lines of the source file, count them and print
# the result.
#      %s = string
#      \n = new line
# NUM=$() = Run code defined in brackets and save result to NUM.
printf "%s\n" $THIN_LINE >> $2
NUM=$(cat $1 | wc -l)
printf $TABLE_FORMAT "All my actions in total:" $NUM >> $2


### Find out how many actions I did per month.
###
# For each year, write an entry to output file. For each
# month, write an entry to output file including how many
# actions I have made in that particular month. Also, count
# a total number of contributions after each year and print it.
# The "let" command enables arithmetic operations on variables.
#      \n = new line
#      %s = string
#       \ = line break
# NUM=$() = Run code defined in brackets and save result to NUM.
echo -e "\n\nMy actions per month:" >> $2
printf "%s\n" $THICK_LINE >> $2

for YEAR in 2012 2013
do

   printf "\n$TABLE_FORMAT" "Year:" $YEAR >> $2
   printf "%s\n" $THIN_LINE >> $2

   for MONTH in January February March April May June July August \
   September October November December
   do

      NUM=$(cat $1 | grep "$MONTH $YEAR" | wc -l)
      printf $TABLE_FORMAT $MONTH $NUM >> $2
      let "TOTAL=$TOTAL+$NUM"

   done

   printf "%s\n" $THIN_LINE >> $2
   printf $TABLE_FORMAT "Total:" $TOTAL >> $2
   TOTAL=0

done


### Show results.
###
cat $2


### Quit
###
```

```
# \n = new line
echo -e "\nAll done.\n"
exit 0
```

MODERATOR CONTRIBUTIONS SCRIPT

```
#!/bin/bash
#
# moderator-contributions.sh
#
#   Copyright (C) 2013  Matti Lammi
#
#   This program is free software: you can redistribute it and/or
#   modify it under the terms of the GNU General Public License as
#   published by the Free Software Foundation, either version 3 of the
#   License, or (at your option) any later version.
#
#   This program is distributed in the hope that it will be useful,
#   but WITHOUT ANY WARRANTY; without even the implied warranty of
#   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#   GNU General Public License for more details.
#
#   You should have received a copy of the GNU General Public License
#   along with this program. If not, see
#   <http://www.gnu.org/licenses/>.
#
#
# This script will produce information about my (lammi87)
# contributions to h-node as a moderator in human readable form. It
# will use tools commonly found in GNU/Linux operating system
# distributions and a source file to do this. Results will be saved
# to a file and also displayed immediately.
#
# The source file is a text file consisting on several one line long
# entries of data. The data is copied from h-node's list of actions
# carried out by moderators (available only to moderators and
# h-node's main developer) where a new entry is made every time a
# user who has moderator's rights carries out an action which only
# moderators can do.
#
# This list is rather long, so I have narrowed it down by filtering
# out every entry about other moderator's actions using a search
# filter found in the list's page, thus keeping my source file short.
#
# I have used many line breaks in this script so that I can keep the
# layout clean when adding this script to my final thesis as an
# appendix.
#
# Please note that the output of this script may not be 100% accurate.
# I think, however, that the results are accurate enough.
#
# SOURCE FILE
# File name:                    source-moderator
# Date of first entry:           2 October 2012
# Date of last entry:           22 March   2013
#
```

```
### Print instructions.
###
# Echo program's -e option allows special characters like
# \n to be implemented.
# \n = new line
# \t = tab
echo -e "H-node script: Actions as moderator\n"
echo -e "First argument:\t\t\tSource file"
echo -e "Second argument: \t\tOutput file\n"


### Quit if source file is not found.
###
# Check if first argument is null.
# !  = not
# -n = not null
if [ ! -n "$1" ]
then
echo "ERROR: No source given. Read source code for more info."
exit 1
fi

# Check if source does not exist.
# !  = not
# -f = file exists
if [ ! -f "$1" ]
then
echo "ERROR: Source not found. Read source code for more info."
exit 1
fi


### Quit if output file is not given.
###
# Check if second argument is null.
# !  = not
# -n = not null
if [ ! -n "$2" ]
then
echo "ERROR: No output file given. Read source code for more info."
exit 1
fi


### Create output file.
###
# \n = new line
touch $2
echo -e "Output file created.\n\n"


### Define table format and lines
###
# I will print the data processed from the source files into easy
# to read tables. I use "printf" instead of "echo" because it has
# more options to format the output. I'll define a format for the
# tables so I can easily use it to print several tables and keep
```

```
# their formatting unanimous. The format is a 35 character long space
# followed by eight character long space and a new line character.
# I added the new line character because "printf" does not include
# it to its output by default. The first space will align characters
# printed into it to the left. The other space will align its content
# to right.
# %s = string
# NN = reserved character space in %NNs
#  - = alignment to left
# \n = new line
TABLE_FORMAT="%-35s%8s\n"

# I'll be using two kinds of "lines" of characters to make the
# output look more readable. I'll define them here.
THICK_LINE="============================================="
THIN_LINE="---------------------------------------------"

### Find out my actions as moderator:
###
# %s = string
# \n = new line
echo "Actions as moderator:" > $2
printf "%s\n\n" $THICK_LINE >> $2

# Find out how many issues I have hidden:
#
# Inspect all lines of the source file, get only those which
# have my username followed by one or more spaces and the
# word "hide", count them and print the result.
# NUM=$() = Run code defined in brackets and save result to NUM.
#      \s = any white space character (regex)
#       * = preceding character zero or more times (regex)
NUM=$(cat $1 | grep "lammi87\s*hide" | wc -l)
printf $TABLE_FORMAT "Issues hidden by me:" $NUM >> $2

# Find out how many issues I have closed:
#
# Inspect all lines of the source file, get only those which
# have my username followed by one or more spaces and the word
# "close", count them and print the result.
# NUM=$() = Run code defined in brackets and save result to NUM.
#      \s = any white space character (regex)
#       * = preceding character zero or more times (regex)
NUM=$(cat $1 | grep "lammi87\s*close" | wc -l)
printf $TABLE_FORMAT "Issues closed by me:" $NUM >> $2

# Find out how many actions I have made in total:
#
# Inspect all lines of the source file, count them and print
# the result.
#      %s = string
#      \n = new line
# NUM=$() = Run code defined in brackets and save result to NUM.
printf "%s\n" $THIN_LINE >> $2
NUM=$(cat $1 | wc -l)
printf $TABLE_FORMAT "Total:" $NUM >> $2
```

```
### Find out how many actions I did per month.
###
# For each year, write an entry to output file. For each
# month, write an entry to output file including how many
# actions I have made in that particular month. The "let"
# command enables arithmetic operations on variables.
#      \n = new line
#      %s = string
#       \ = line break
# NUM=$() = Run code defined in brackets and save result to NUM.
echo -e "\n\nMy actions per month:" >> $2
printf "%s\n" $THICK_LINE >> $2

for YEAR in 2012 2013
do

   printf "\n$TABLE_FORMAT" "Year:" $YEAR >> $2
   printf "%s\n" $THIN_LINE >> $2

   for MONTH in January February March April May June July August \
   September October November December
   do

      NUM=$(cat $1 | grep "$MONTH $YEAR" | wc -l)
      printf $TABLE_FORMAT $MONTH $NUM >> $2
      let "TOTAL=$TOTAL+$NUM"

   done

   printf "%s\n" $THIN_LINE >> $2
   printf $TABLE_FORMAT "Total:" $TOTAL >> $2
   TOTAL=0

done


### Show results.
###
cat $2


### Quit
###
# \n = new line
echo -e "\nAll done.\n"
exit 0
```

NETBOOK COMPATIBILITY SCRIPT

```
#!/bin/bash
#
# netbook-compatibility-levels.sh
#
#   Copyright (C) 2013  Matti Lammi
#
#   This program is free software: you can redistribute it and/or
#   modify it under the terms of the GNU General Public License as
#   published by the Free Software Foundation, either version 3 of the
#   License, or (at your option) any later version.
#
#   This program is distributed in the hope that it will be useful,
#   but WITHOUT ANY WARRANTY; without even the implied warranty of
#   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#   GNU General Public License for more details.
#
#   You should have received a copy of the GNU General Public License
#   along with this program. If not, see
#   <http://www.gnu.org/licenses/>.
#
#
# This script will produce information about the compatibility levels
# of netbooks found in the h-node's hardware database with 100% free
# GNU/Linux distributions endorsed by the Free Software Foundation.
# The script will use tools commonly found in GNU/Linux distributions
# and a source file to do this. Results will be saved to a file and
# also displayed immediately.
#
# The source file is an XML file which contains detailed information
# about all the notebooks found in the database. The source file [1]
# can be found in the h-node's download page [2]. Anyone is free to
# copy, modify, publish, use, sell, or distribute the file's content
# for any purpose, commercial or non-commercial, and by any means.
# For more details, please check the license [3].
#
# [1] http://h-node.org/download/notebooks/en
# [2] http://h-node.org/download/index/en
# [3] http://creativecommons.org/publicdomain/zero/1.0/
#
# Netbook are given different compatibility levels in h-node depending
# on their compatibility with free software. From highest to lowest,
# the levels are; A, B, C, D and E. In simple terms, they are as
# follows:
#
#          A-Platinum - All devices work with full performance
#          B-Gold     - All devices work but not with full performance
#          C-Silver   - One device is not supported
#          D-Bronze   - More than one device is not supported
#          E-Garbage  - Netbook is not usable with free software
#
# I have used many line breaks in this script so that I can keep the
# layout clean when adding this script to my final thesis as an
```

```
# appendix.
#
# Please note that the output of this script may not be 100% accurate.
# I think, however, that the results are accurate enough.
#
#
# SOURCE FILE
#
# File name:            source-netbook-compatibility-levels.xml
# Download date:        26 March 2013
#


### Print instructions.
###
# "echo" program's -e option allows special characters like
# \n to be implemented.
# \n = new line
# \t = tab
echo -e "H-node script: Netbook compatibility levels\n"
echo -e "First argument:\t\t   Source file"
echo -e "Second argument: \t   Output file\n"


### Quit if source file [1] is not given.
###
# Check if first argument is null.
#  ! = not
# -n = not null
if [ ! -n "$1" ]
then
echo "ERROR: No source given. Read source code for more info."
exit 1
fi

# Check if source [1] does not exist.
#  ! = not
# -f = file exists
if [ ! -f "$1" ]
then
echo "ERROR: Source not found. Read source code for more info."
exit 1
fi


### Quit if output file is not given.
###
# Check if second argument is null.
# !  = not
# -n = not null
if [ ! -n "$2" ]
then
echo "ERROR: No output file given. Read source code for more info."
exit 1
fi
```

```
### Create output file.
###
# \n = new line
touch $2
echo -e "Output file created.\n\n"


### Define table format and lines
###
# I will print the data processed from the source files into easy
# to read tables. I use "printf" instead of "echo" because it has
# more options to format the output. I'll define a format for the
# tables so I can easily use it to print several tables and keep
# their formatting unanimous. The format is a 16 character long space
# followed by 17 and 20 character long space and a new line character.
# I added the new line character because "printf" does not include
# it to its output by default. All character spaces will align
# characters printed into them to the left.
# %s = string
# NN = reserved character space in %NNs
#  - = alignment to left
# \n = new line
TABLE_FORMAT="%-16s%-17s%-20s\n"

# I'll be using two kinds of "lines" of characters to make the
# output look more readable. I'll define them here.
THICK_LINE="================================================"
THIN_LINE="------------------------------------------------"


### Find out how compatible the netbooks are:
###
printf "$TABLE_FORMAT" "Level" "Amount" "Percentage" > $2
printf "%s\n\n" $THICK_LINE >> $2

# Find out how many netbooks there are and save the result to a
# variable. Run a loop to check each compatibility level separately
# to find out how many netbooks have a particular compatibility
# level. Format the result to a nice table. The "bc" program is used
# for math when a floating point is required and "let" when simple
# operations with variables are performed. Variables used are as
# follows:
#
# TOTAL                 total amount of netbooks in the database
# CALC_TOTAL            total amount of netbooks calculated from each
#                       compatibility level
# NETBOOKS              amount of netbooks with a certain
#                       compatibility level
# PERCENTAGE            netbooks of one compatibility level compared
#                       to total (not accurate, but good enough)
# CALC_PERCENTAGE       netbooks of all compatibility levels
#                       compared to total
#
# %f = float
#    => %NN.Zf = reserve a space of NN characters and input a float
#                type number to that space defined in <argument>
#                aligning right and define the number of decimal
```

```
#                digits in Z.
#  - = alignment to left
# \n = new line


TOTAL=$(cat $1 | grep "<compatibility>" | wc -l)
CALC_TOTAL=0
PERCENTAGE=0

for LEVEL in A-platinum B-gold C-silver D-bronze E-garbage
do

   NETBOOKS=$(cat $1 | grep "<compatibility>" | grep $LEVEL | wc -l)
   PERCENTAGE=$(echo "scale=4; ($NETBOOKS/$TOTAL)*100" | bc)
   printf "%-15s%7s%21.1f\n" $LEVEL $NETBOOKS $PERCENTAGE >> $2
   let "CALC_TOTAL=$CALC_TOTAL+$NETBOOKS"

done

CALC_PERCENTAGE=$(echo "scale=3; ($CALC_TOTAL/$TOTAL)*100" | bc)
printf "%s\n" $THIN_LINE >> $2
printf "%-15s%7s%21.0f\n" "Total:" $CALC_TOTAL $CALC_PERCENTAGE  >> $2


### Show results.
###
cat $2


### Quit
###
# \n = new line
echo -e "\nAll done.\n"
exit 0
```

OUTPUT OF HARDWARE CONTRIBUTIONS SCRIPT

```
Hardware contributions:
================================================

New hardware inserted by me:                 247
Old hardware updated by me:                  483
------------------------------------------------
All my contributions in total:               730
Other's contributions in total:              646
------------------------------------------------
All contributions in total:                 1376


My contributions per month:
================================================

Year: 2012           Updated  Inserted     Total
------------------------------------------------
January                    0         0         0
February                   0         0         0
March                      0         0         0
April                      0         0         0
May                       12         7        19
June                      35        26        61
July                       6         0         6
August                    29         2        31
September                150        51       201
October                  113        65       178
November                  74        46       120
December                  44        34        78
------------------------------------------------
Total:                   463       231       694

Year: 2013           Updated  Inserted     Total
------------------------------------------------
January                   15        14        29
February                   4         0         4
March                      1         2         3
April                      0         0         0
May                        0         0         0
June                       0         0         0
July                       0         0         0
August                     0         0         0
September                  0         0         0
October                    0         0         0
November                   0         0         0
December                   0         0         0
------------------------------------------------
Total:                    20        16        36


My contributions per hardware class:
================================================
```

APPENDIX 5 2 (2)

Class	Updated	Inserted	Total
notebook	204	117	321
videocard	68	60	128
soundcard	54	16	70
wifi	51	10	61
ethernet-card	54	19	73
sd-card-reader	11	6	17
webcam	6	4	10
bluetooth	0	0	0
acquisition-card	3	3	6
3G-card	6	2	8
host-controller	2	1	3
fingerprint-reader	1	0	1
RAID-adapter	0	0	0
modem	3	0	3
printer	15	8	23
scanner	3	1	4
Total:	481	247	728

OUTPUT OF ADMIN CONTRIBUTIONS SCRIPT

```
Actions as administrator:
=========================================

Users blocked by me:                     4
Devices deleted by me:                   6
Devices hidden by me:                    1
-----------------------------------------
All my actions in total:                11


My actions per month:
=========================================

Year:                                 2012
-----------------------------------------
January                                  0
February                                 0
March                                    0
April                                    0
May                                      0
June                                     0
July                                     0
August                                   0
September                                0
October                                  4
November                                 2
December                                 3
-----------------------------------------
Total:                                   9

Year:                                 2013
-----------------------------------------
January                                  1
February                                 0
March                                    1
April                                    0
May                                      0
June                                     0
July                                     0
August                                   0
September                                0
October                                  0
November                                 0
December                                 0
-----------------------------------------
Total:                                   2
```

APPENDIX 7 1 (1)

OUTPUT OF MODERATOR CONTRIBUTIONS SCRIPT

```
Actions as moderator:
=============================================

Issues hidden by me:                        5
Issues closed by me:                        8
---------------------------------------------
Total:                                     13


My actions per month:
=============================================

Year:                                   2012
---------------------------------------------
January                                    0
February                                   0
March                                      0
April                                      0
May                                        0
June                                       0
July                                       0
August                                     0
September                                  0
October                                    5
November                                   3
December                                   2
---------------------------------------------
Total:                                    10

Year:                                   2013
---------------------------------------------
January                                    2
February                                   0
March                                      1
April                                      0
May                                        0
June                                       0
July                                       0
August                                     0
September                                  0
October                                    0
November                                   0
December                                   0
---------------------------------------------
Total:                                     3
```

OUTPUT OF NETBOOK COMPATIBILITY SCRIPT

Level	Amount	Percentage
A-platinum	84	26.8
B-gold	67	21.3
C-silver	118	37.6
D-bronze	34	10.8
E-garbage	11	3.5
Total:	314	100