Collaboration in Open-Source Hardware: Third-Party Variations on the Arduino Duemilanove

David A. Mellis & Leah Buechley
MIT Media Lab
75 Amherst St., Cambridge, MA 02142
{mellis, leah}@media.mit.edu

ABSTRACT
This paper looks at collaboration in open-source hardware: physical goods whose digital design files are shared for others to make or modify. Through research into nine variations on the Arduino Duemilanove (an electronic circuit board) and interviews with their developers, we explore the process and outcome of open-source hardware development. We find a structure that differs substantially from that of most open-source software projects, involving many small-scale collaborations rather than a centralized process. We discuss three possible reasons for this structure: differing component selections, the investment required for prototyping, and the lack of software collaboration tools.

Author Keywords
open-source hardware, collaboration, electronic circuit boards, Arduino, version control

ACM Classification Keywords
K.5.1 [Legal Aspects of Computing]: Hardware/Software Protection.

General Terms
Documentation, Economics, Legal Aspects

INTRODUCTION
Recently, the philosophy and principles of open-source software have begun to be applied to the design and production of physical products. This practice of open-source hardware involves sharing the original design files for an object in a way that allows it to be modified or reproduced by others, including for commercial use. The recently drafted open-source hardware statement of principles [6] explains it this way: "Open source hardware is hardware whose design is made publicly available so that anyone can study, modify, distribute, make, and sell the design or hardware based on that design." Open-source hardware has been perhaps most-widely adopted in the domain of electronic circuits, both by producers of do-it-yourself kits (e.g. Adafruit Industries) and of mass-manufactured devices (e.g. Chumby). These companies publish schematics and layout files for circuit boards and computer-aided design (CAD) files for enclosures or other mechanical parts. Others can study and modify these designs to create their own versions of a product or to contribute improvements back to the original manufacturer.

We see these activities as a form of distributed collaboration, enabled by the internet and by the increasing accessibility of fabrication technology. This paper explores the nature of such open-source hardware collaboration through a prominent example, the Arduino microcontroller development platform. Here, we examine the nine variations on the Arduino Duemilanove (an electronic circuit board) that were included a comprehensive list published in 2009 by Make Magazine [8]. By reviewing online information about these variations and through present-day interviews with their developers, we form a model of open-source hardware collaboration. We find that its structure differs substantially from that of open-source software, emphasizing multiple alternatives from small-scale partnerships rather than centralized public processes. In this work, we are drawing partly on our own experience as developers of Arduino and of one of the variations discussed here (the LilyPad Arduino).

In the next section, we briefly review related work. Then, we introduce the Arduino platform and analyze the changes found in the third-party variations of its circuit board. This is followed by information about developer motivations and processes gleaned from our interviews. We then sketch a model of the collaboration found in this community and speculate on the ways it is shaped by available tools and processes.

RELATED WORK
There's a long, established body of research that discusses open-source software (e.g. [3]). Only recently, however, has this kind of attention been focused on the application of open-source to hardware. For example, [4] surveys a group developing an open-source electric car conversion kit, detailing their background and motivation. In another work [5], we developed a new product – an open-source FM radio – in order to explore possibilities for user customization and construction of consumer electronic

CSCW'12, February 11–15, 2012, Seattle, Washington, USA.

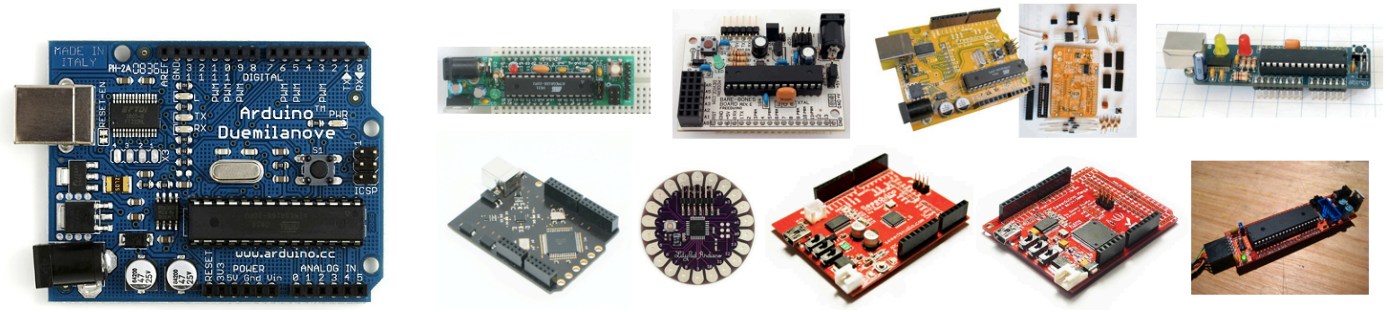

Figure 1. The Arduino Duemilanove (left) and variations: (top row) Boarduino, Bare Bones Board, Freeduino, iDuino; (bottom row) Illuminato Genesis, LilyPad Arduino, Seeeduino, Seeeduino Mega, Sanguino.

devices. Our paper [1] about the LilyPad Arduino discusses further variations on that product's hardware.

Open-source hardware can also be seen as an example or enabler of technological appropriation [2] and user-driven innovation [9]. By providing design files for others to modify and build from, it allows them to adapt technology to their own circumstances or needs. This blurs the division between producer and consumer in ways similar to those discussed in the theories of Eglash and von Hippel.

THE ARDUINO PLATFORM AND VARIATIONS

The Arduino platform consists of both microcontroller-based circuit boards and the software for programming them. It is widely used by designers, artists, computer programmers, and other electronics hobbyists. Both the source code for the Arduino software and the original design files for the Arduino hardware are published under open-source licenses. Arduino has become one of the better known examples of open-source hardware, featured, for example, in a Wired Magazine article [7] on the subject.

The standard, official version of the Arduino hardware was, at the time of the Make Magazine list, the Arduino Duemilanove (shown in Figure 1). This board is sold pre-assembled, includes USB connectivity for communication with the Arduino software, and has female headers (sockets) for connecting electronic components or wires. The nine variations on the Duemilanove included in the list are also shown in Figure 1. Two (the Seeeduino and the Seeeduino Mega) were produced by the same company, but each of the others originates from a different group of developers. Eight of the nine boards (all but the iDuino) continue to be available for purchase from their original producers. All of the boards can be programmed using the Arduino software, allowing them to benefit from the Arduino documentation and user community.

The differences between the variations and the original Arduino hardware can be grouped into five major categories:

- aesthetics: e.g. PCB color and shape, component layout, and additional LEDs.
- form factor: e.g. four of the variations can be inserted into a solderless breadboard.
- assembly: five of the variations come as kits to be soldered together by the end user, using larger through hole components instead of surface mount parts.
- processor: three variations include a different processor than that used on the standard Arduino board.
- other functional variations: e.g. omission of on-board USB connectivity, alternative power connections.

It is interesting to note that many of these variations offer alternative functional or aesthetic design choices as opposed to simple improvements or added functionality. These kinds of modification don't lend themselves to contribution back to the original design because they reflect a different (and incompatible) approach than that taken by the Arduino hardware. Still, together they provide a useful range of products to the Arduino community, an example of the open-sourcing of hardware designs leading to distributed development of a platform's offerings.

DEVELOPER MOTIVATIONS AND PROCESS

In order to gather additional information for this paper, we requested interviews with the developers of the Arduino variations. Five responded, four by email and one by phone. We asked questions about topics including motivation, collaborators, communication methods, software design tools, the prototyping process, and financial arrangements. Below we present quotes and information from the interviews, supplemented with data gleaned from public websites. We also draw on our own experience in developing the LilyPad Arduino.

The motivations for many of the Arduino derivatives stemmed from functional requirements for the hardware itself. The Sanguino website explains: "We needed a more powerful microprocessor, but we were reluctant to leave the warm bosom of the Arduino community." The creator of the Bare-Bones Board wanted "low-cost boards that would fit into breadboards." The Seeeduino has "13 changes like [surface-mount microcontroller], 2.54 [mm] pitch pin, mini USB and etc." We created and commercialized the LilyPad Arduino to make it easier for people to attach electronics to fabric. The developers of the Freeduino had more varied motivations; one explains: "the Arduino team had just released the Diecimila [the predecessor of the Duemilanove], but not the accompanying reference

design.... D____... was angry at the lapse in open-sourceness and wanted cheaper versions for his students.... I... was mainly interested in doing the PCB design as a sort of education exercise."

In contrast with open-source software, where development tends to be done publicly with anyone able to contribute, much of the development of the Arduino variants was done privately by select collaborators. For example, the Boarduino was "just L____." The developer of the Bare Bones Board worked with "students, co-teachers, and a couple of other electronic suppliers / hobbyists." A developer of the Freeduino described the roles this way: "roughly, D____ was 'marketing,' T____ was 'manufacturing,' and I was 'engineering.'" Interestingly, the three had "no previous relationships; we met on the Arduino forum." Three developers worked on the Illuminato Genesis as part of a commercial partnership. The Seeeduino boards were developed by a company (Seeedstudio) based on "complaints / suggestions to the Arduino Diecimila." We went to an existing company, SparkFun Electronics, to commercialize the LilyPad Arduino.

Communication processes vary across the collaborations. Some of the teams were distributed, like the Illuminato Genesis developers – who coordinated using phone calls – and the Freeduino developers. In this latter case, "communication was almost exclusively by email, with the freeduino web site eventually being set up as a drop-box for design files." The developers of the Bare-Bones Board also "mostly emailed gifs of files, or actual files back and forth." The Seeeduino developers "collaborate with wiki and dropbox [file sharing], project management basing [sic] on Excel." The other developers didn't mention use of software for project management.

Development of the variations involved different prototyping processes. The Boarduino, for example, "took about 2 [iterations]!" For the Freeduino, "N____ had all the prototypes made (100 of each [of four] style[s]) and distributed bare boards to each of the team members.... all four boards worked." The developer of the Bare-Bones Board also mentioned mailing boards to collaborators. The company producing the Seeeduino has a "quick board house able to generate PCB in 24 hours, then our RnD tech will help solder the prototype." We received prototype hardware from SparkFun during the development of the LilyPad Arduino.

Seven of the nine variations (all but the Bare-Bones Board and Illuminato Genesis) were designed using the same circuit layout software (Eagle) as the original Arduino board. Some of the developers started from the original Arduino design files in creating their variations; others simply recreated them from scratch using the originals as a reference (in the words of the Boarduino developer: "I only looked at the schematic to make sure it was compatible"). None of the projects make regular public use of a software version control system (like Subversion or Git), although the Sanguino used an online version control system to publish the finished design files.

The design files for the Arduino variants tended to be published, if at all, only when they were finished and the board had been produced. As a result, the boards have very few public versions (to date). Three of the variations (the Boarduino, Illuminato Genesis, and Seeeduino Mega) appear to have had only a single public release and all but the Freeduino have had five or fewer. Interestingly, though, over half of the variations were themselves the basis for further variations, typically by the same developer or manufacturer.

In all but two of the cases (the LilyPad Arduino and the Freeduino), the Arduino variations were designed and developed by the same people or company that eventually made and sold the boards. The money invested for prototyping can therefore be seen as an investment in the expectation that it will be made up with sales of the finished boards. In the other two cases, one person did the initial conceptual development and initial versions of the product (LilyPad Arduino) or the final hardware design (Freeduino) but another party handled the cost and logistics of manufacturing the prototypes and boards. In both cases, all or almost all of the eventual revenues from sales of the board went to the manufacturer, not the designer. As the designer of the Freeduino explains it, "As far as I know, N____ is the only team member who spent real money, and the only one to have seen any real income from the project." In the case of the Illuminato Genesis, the board was never intended as a source of income. Instead, it was sold at or below cost as a promotion for the engineering consulting services of the company making it.

DISCUSSION: COLLABORATION STRUCTURE AND INFLUENCES

These variations on the Arduino circuit board present a very different portrait of open-source collaboration than that seen in most open-source software projects. It is an ecosystem of many small groups or companies providing alternative offerings within a product space. Individual products undergo few iterations; instead, their designs are used as the basis for new alternative variations. In this model, open-source collaboration means increasing the number of choices offered to users rather than improving the functionality of a single, central option. This may be partially a result of the relatively simple design of the Arduino hardware but it also reflects the nature of physical products. With software, the functionality and appearance of a program can be customized by the user after it's been downloaded; with hardware, user control typically requires a choice of products. As a result, making an alternative version of an open-source hardware design can serve as a positive contribution to the community, in contrast to software where divergent code bases are often seen as resulting from arguments or differences among developers.

Besides the value of increasing the choice of products available to consumers, there are a number of aspects of

hardware development processes and tools that seem to promote the decentralized collaboration structure found in this example of open-source hardware. We discuss three of the most important in the following paragraphs, using our personal experience and knowledge to make some connections that may not be immediately obvious from the variants themselves. Still, we think they serve as a good illustration of the influence of these factors on open-source hardware collaboration. Here, we discuss these issues in the context of electronic circuit boards but we think they also apply to the development of other kinds of open-source hardware.

Agreeing on the components to use in a hardware design is critical to enabling collaboration between developers. There are many practical obstacles; for example, certain components may not be available in a given country or a company may already stock alternative versions of a part. A particularly pronounced split is between the smaller surface-mount components that are used for machine-assembled circuits (e.g. the Arduino Duemilanove, LilyPad Arduino, Seeeduino, Seeeduino Mega, and Illuminato Genesis) and the larger through-hole components favored for manual assembly (e.g. on the Bare-Bones Board, Boarduino, iDuino, and Sanguino). It's difficult for someone to make or test a change to the design of a board if it requires them to first acquire a new set of components or redesign large portions of the circuit to use the components they already have. In order to easily collaborate, therefore, individuals need to agree on and have access to a particular set of components and assembly processes.

The constraints of manufacturing make it more difficult and costly to test design revisions, limiting the frequency of iteration and the ability to integrate modifications from many contributors. Additionally, the money required to make a product emphasizes the financial and business side of the development process, aspects which can often be postponed or sidelined in open-source software development. This need to invest financial resources in the development process may encourage someone to make and sell their own version of the hardware, rather than contribute improvements back to a product in which they have no monetary stake. For example, the Seeeduino is similar in both functionality and production to the original Arduino Duemilanove. Its manufacturer didn't suggest their specific improvements to the Arduino project but instead uses them as a means of distinguishing their products in the marketplace.

Finally, unlike software, hardware doesn't have mature tools or techniques for tracking and integrating changes using free or low-cost tools. While existing version control systems can be used for hardware designs, they simply store subsequent versions of a file without providing mechanisms for understanding the differences between them. This makes it harder to examine or combine changes between multiple versions of a design, complicating collaboration. Another difficulty is the necessity of tracking the version of a design that was used to make a particular physical product, so that problems with the hardware can be diagnosed with reference to its design.

CONCLUSION

In examining these variations on the Arduino Duemilanove, we've sketched a model of collaboration in open-source hardware. It is very different from that typically seen in open-source software, involving multiple small-scale collaborations rather than many people working on a single project. We identified three possible reasons for this structure: the complication of sharing components, the investment required for prototyping, and the lack of mature software tools for collaboration. Still, despite these challenges, we think this case study shows some of the exciting and evolving possibilities of open-source hardware collaboration.

ACKNOWLEDGEMENTS

Thanks to the makers of the variations for responding to our questions.

REFERENCES

1. Buechley, L. and Hill, B.M. 2010. LilyPad in the wild: how hardware's long tail is supporting new engineering and design communities. DIS '10. ACM, New York, NY, USA, 199-207.
2. Eglash, R., Appropriating Technology: An Introduction. in Eglash, R., Croissant, J.L., Di Chiro, G., and Fouche, R. (eds.) *Appropriating Technology: Vernacular Science and Social Power*, University of Minnesota Press, 2004.
3. Feller, J., Fitzgerald, B., Hissam, S., and Lakhani, K. (eds.). 2007. *Perspectives on Free and Open Source Software*. MIT Press. Cambridge, MA.
4. Malinen, T., Mikkonen, T., Tienvieri, V., and Vadén, T. 2010. Open source hardware through volunteer community: a case study of eCars -- now!. MindTrek '10. ACM, New York, NY, USA, 65-68.
5. Mellis, D.A., Gordon, D., and Buechley, L. 2010. Fab FM: the design, making, and modification of an open-source electronic product. TEI '11. ACM, New York, NY, USA, 81-84.
6. Open-Source Hardware Statement of Principles and Definition. http://freedomdefined.org/OSHW
7. Thompson, C. 2008. Build It. Share It. Profit. Can Open Source Hardware Work?. Wired Magazine 16, 11 (October 2008).
8. Torrone, P. 2009. The definitive guide to open source hardware projects: Arduino. http://blog.makezine.com/archive/2009/12/arduino-open-source-hardware-2009.html
9. von Hippel, E. and Katz, R. 2002. Shifting Innovation to Users via Toolkits. Management Science 48, 7 (July 2002), 821-833.