III

Production Bits

11

Personal Manufacturing in the Digital Age

David A. Mellis

> "From my point of view, the greatest developments to be expected of technics in the future . . . will not be, as we are usually led to think, in the direction of universalizing even more strenuously the wasteful American system of mass production: no, on the contrary, it will consist in using machines on a human scale, directly under human control, to fulfill with more exquisite adaptation, with a higher refinement of skill, the human needs that are to be served. . . . Much that is now in the realm of automatism and mass production will come back under directly personal control, not by abandoning the machine, but by using it to better purpose, not by quantifying but by qualifying its further use."
>
> —Lewis Mumford, *Art and Technics* (1952)

Digital technology is enabling new alternatives to industrial production. Computer-aided design (CAD) tools encode objects as information, allowing their designs to be freely shared online—the practice of open source hardware. Digital fabrication machines turn this information into objects, allowing for precise, one-off production of physical goods. A variety of sophisticated off-the-shelf electronic components enable complex sensing, actuation, communication, and interfaces. Together, these technologies enable individuals to produce complex devices from digital designs, a process we can think of as *personal manufacturing.*

Because open source hardware involves treating physical objects as digital information, it suggests that we may be able to apply principles and practices from other kinds of online collaboration to the design of hardware. Open source software, Wikipedia, and other digital artifacts incorporate the creativity of many different individuals working without the direction of markets or firms, a process known as peer production. It works because the means of production of digital goods—computers and software—are widely distributed, the Internet makes communication and coordination efficient, and the work can be

divided into pieces that individuals can choose to work on based on their own interests, needs, and abilities. The extent to which peer production can apply to hardware will shape the extent to which this approach can provide a viable alternative to mass production for the technology in our lives.

To make electronic devices amenable to these peer production approaches, we need to design with them in mind. This process yields devices that look very different than ones that are industrially produced. Such devices are optimized for translation from the digital design to the physical object. They make use of a variety of processes, from the much-hyped 3D printing to the more prosaic (but potentially more useful) techniques of laser cutting, CNC milling, and circuit board fabrication. They allow for a variety of materials and aesthetics. They can be adapted by individuals for their own needs and interests. They allow for different business models, in which objects can be made on demand or in small quantities to serve specific markets or particular individuals.

Of course, none of this eliminates the need for individual skill, whether in the design process or in the use of the fabrication machines. Good CAD tools can make the process easier, but translating an idea into concrete form requires many decisions and compromises that rely on human skill, experience, and intuition. Similarly, making effective use of a fabrication machine relies on knowledge of its configuration, operation, limitations, and quirks. Technology offers possibilities, but people turn those possibilities into reality. Similar considerations exist in open source software, where peer production doesn't eliminate the need for expertise on the part of contributors but rather provides new ways of organizing and combining those individuals' skills and efforts.

The two case studies discussed in this chapter—dealing with Arduino boards and my own consumer electronic devices—illustrate different possibilities and limitations of working with these techniques. Together, they illustrate this new personal manufacturing ecosystem, highlighting its implications for product design, for collaboration, and for business. They show some of the ways that digital technology can transform the production of objects, but also indicate some of the constraints derived from industrial systems that persist in personal manufacturing. They provide some hints of what a peer production ecosystem for electronic devices might look like, yet also point out some of the difficulties to be overcome in creating one.

The next section gives an overview of personal fabrication and the considerations involved in going from an open source hardware design file to an actual physical object. This discussion is followed by the two case studies. The lessons from the case studies are used to derive some general principles for open source hardware and personal manufacturing. Finally, I conclude with some questions and thoughts for the future.

Personal Fabrication, Processes, Parts, and Materials

Digital fabrication machines translate open source hardware designs into actual physical objects. In theory, this process depends only on the digital file and the choice of fabrication machine, allowing for iteration and refinement through successive changes to the file. In practice, though, the constraints and intricacies of various fabrication processes mean

that a certain amount of skill is required to use the machine and that the results can vary each time. As a result, open source hardware depends on the selection of appropriate processes and effective use of them. This section discusses some of the considerations involved in various popular fabrication processes.

3D Printing

The purest of these digital fabrication processes are the various forms of 3D printing. These turn digital design into physical objects by gradually adding material in the desired locations, allowing for a wide range of possible geometries. The term *3D printing* encompasses a broad range of machines, from personal plastic printers costing a few hundred dollars to industrial machines that sinter metal and cost hundreds of thousands of dollars. Different machines work with different materials and offer different resolutions and tolerances. The materials may have different strengths, optical properties, appearances, finishing possibilities, and so on. Depending on the object being fabricated, some or all of these characteristics may be crucial to creating a useable result. In designing and sharing objects for 3D printing, therefore, it's important to specify not just their geometries, but also the required tolerances, materials, and other characteristics—most of which are less easily captured in digital form. In addition, many 3D-printing processes need some form of manual post-processing, such as removal of support material, finishing, or curing. These require an operator with appropriate knowledge and skill—and can create variations from one print to the next, even with the same file and machine. Finally, 3D printing technology is evolving and diversifying rapidly. For all these reasons, it's important not to think of 3D printing as a way to automatically create things from information, but rather as a material process with specific qualities and affordances.

Milling and Cutting

Other fabrication processes work by cutting or removing pieces of a larger stock material. Laser cutters cut 2D shapes out of plywood, cardboard, acrylic, and other flat materials. Vinyl cutters do the same, but with a knife that cuts through thin materials like paper or adhesive-backed vinyl. The water-jet cutter handles stronger and thicker materials like wood, metal, and glass, cutting with a stream of hard particles in a powerful jet of water. CNC (computer-numeric control) machines, like mills or routers, work in three (or more) dimensions, removing material from solid blocks of stock with a variety of cutting bits. They are often capable of very precise operations, albeit only within specific axes of movement. Compared with 3D printers, these cutting and milling tools have the advantage of being able to work with a variety of existing materials, including natural ones with complex structures that are difficult or impossible to replicate with the homogenous stock of most 3D printers. They are more limited in the geometries they can produce, however, and often require more steps in fabricating or assembling the parts.

In addition to specifying the geometry of the design itself, it's important to be explicit about the nature of the stock material and the characteristics of the cutting process. Whether two parts press-fit tightly together, slip past each other, or don't fit at all depends as much on the precise thickness of the stock (which can vary even across nominally

equivalent materials) and the thickness of the cut as on the shape in the file. Some constructions may be infeasible to achieve given the tolerances of a particular machine. (Laser cutters may yield slightly different cut thicknesses on different sides of their working area; water-jet cutters can give rough, nonvertical edges, for example.) Traditional engineering drawings often capture the required tolerances for various surfaces and the material to be used. A quickly created CAD file used for a prototype and then thrown up on a webpage may not. Parts might be sanded, glued, pounded together, or otherwise tweaked in ways not reflected in the design files. Generating tool paths for a CNC machine is a complex process with a significant impact on the form and finish of the resulting object; this complexity may not be possible to capture in a way that can be easily shared with others, particularly if they are using a different machine. Finishing and assembling parts created with CNC devices requires careful craft, which might be difficult to communicate or learn. All of these factors need to be kept in mind when designing or sharing a digital file for someone else to replicate.

Other Fabrication Machines

A variety of other digital fabrication processes exist, each with its own affordances and constraints. For example, a host of machines are available for working with soft materials: CNC embroidery machines apply custom designs to fabric, knitting machines generate colors and constructions based on digital files, and Jacquard looms are possibly the oldest digital fabrication machines in existence. Industrial production uses a variety of automated machines, including robot arms and other adaptable parts of an assembly line. Furthermore, as digital fabrication becomes more established, more people are creating their own machines for custom purposes of various kinds.

Printed Circuit Boards and Electronics

The production of printed circuit boards (PCBs) can also be considered a digital fabrication process—and a relatively mature one. Digital designs are etched from copper or other materials using a photographic process, then covered with an isolating layer and text and other annotations. While the processes for creating circuit boards in this way are generally toxic and the automated systems for doing so are expensive, many services will produce PCBs on demand for individual customers with small or nonexistent minimums and standard specifications and tolerances. (As a board's specifications get more demanding, however, costs can increase, sometimes dramatically.) Circuit boards can also be manually etched or milled on a CNC machine, processes that are more directly accessible to individuals but also less robust and precise. While some circuits are sensitive to the precise characteristics of circuit board's substrate or the exact tolerances of the fabrication process, a great many can be shared with relative confidence that they will work when made on a different machine from a different provider.

In reproducing circuits, then, the main difficulties are typically getting the necessary parts and assembling them. While vast quantities of components are available to individuals—and many distributors specifically target hobbyists—advanced parts with

specific functionality may not be accessible. These may be simply impossible to purchase, require an extended procurement process that makes replication infeasible, or be difficult or impossible to assemble with the processes available. As parts are optimized for size and automated assembly, they become harder for individuals to work with. Even easier-to-solder parts rely on manual skill and the knowledge to troubleshoot problems. Different electronic components may be available or preferred in different locations. Parts may go out of stock, become obsolete, or cease being made altogether. All of these factors mean that while making a PCB may be a robust and accessible process, much work must be done to ensure that individuals are able to replicate a complete electronic circuit for themselves. (It's also worth noting that while the problem may be worse for electronic components, other materials—such as plywood or 3D printer stock—are also industrial products and may not be available everywhere or all the time.)

Access to Fabrication

Access to digital fabrication processes comes in a variety of forms. Some machines, particularly 3D printers and vinyl cutters, are being targeted at individual consumers via low-cost, easy-to-use models. Local workshops, whether at schools, libraries, community centers, or commercial locations, provide access to larger, messier, and more expensive machines. They also offer opportunities for people to learn how to use the machines and can provide a community of like-minded individuals. Online services offer an alternative for those without local, hands-on access. They can provide a larger variety of processes and materials than those found in a single workshop and obviate the need to learn to operate the machines directly. On the downside, the time required for parts to be produced and shipped—and the lack of direct control over the process—can make it harder to iterate and refine designs when using an online service. Additionally, online services generally involve higher per-part prices than direct machine access, since they need to cover the cost of the machines, labor, and infrastructure required to support the service.

Case Studies

There's a lot more to open source hardware than just the fabrication and electronics technology. The following case studies draw on my personal experiences with open source hardware to discuss some of the real-world issues involved. The first case study looks at the Arduino electronics platform, a well-known open source hardware project. The second case study discusses my research at the MIT Media Lab, building open source and DIY consumer electronic products.

Case Study: Arduino Microcontroller Development Boards and Their Derivatives

Arduino is a platform for building interactive objects. It consists of microcontroller-based circuit boards and the software for programming them (both of which are open source), along with relevant documentation and community support.

Arduino builds on the work of many other projects, including the Wiring electronics prototyping platform, the Processing development environment, the GNU C Compiler (gcc), AVR libc, avrdude, and more. Since the Arduino electronics prototyping platform started in 2005, it has spawned and participated in a diverse ecosystem of software, hardware, communities, and companies. The relationships between the various actors in the Arduino ecosystem take many different forms: some specifically relate to the open source nature of the Arduino hardware, others reflect its open source software, and still others are based on more traditional business factors. As a co-founder of Arduino, I've witnessed many of these stories over the years. This case study attempts to make sense of the lessons of Arduino for open source hardware and personal manufacturing.

Because the original Arduino circuit boards are relatively simple, were created with a low-cost circuit design tool (Eagle), and use widely available parts, it's relatively straightforward for someone to make their own versions of them. This has led to a proliferation of Arduino derivatives, with a number of different modifications. These boards reveal an open source hardware ecosystem with a very different structure than that of most open source software projects. Successful open source software projects typically involve decentralized collaboration efforts, in which a number of individuals contribute to a single body of source code. The derivatives of the Arduino hardware, in contrast, tend to be produced by a small group of people, often the same ones who sell the resulting product. These derivatives often undergo few public revisions, even though some have remained available for purchase for a number of years. Moreover, relatively few changes have been contributed back to the design of the official Arduino boards. Overall, the derivatives constitute a diverse set of alternatives from different producers, in contrast to the centralized codebase that seems to prevail in most open source software projects (including, in many respects, the Arduino software itself). While some derivatives (like the LilyPad Arduino) have been incorporated into the official Arduino product line, very few (if any) modifications have been contributed to existing boards.

There are multiple obstacles to collaboration on centralized hardware designs that go beyond the need for human skill and motivation common to other domains (like open source software). One is the difficulty and expense of fabricating and assembling boards. Soldering them by hand can be done in small quantities (facilitating changes and the creation of unique variations) but is time consuming, error prone, and limited in the parts it can work with. Automated assembly is more efficient but typically requires larger quantities, limiting the frequency of changes to the circuit's design. Even worse, it can be difficult to switch between these approaches because they may require the use of different components.

Another obstacle is the relative unavailability of tools for tracking and merging changes to the design of circuit boards. Open source software has robust version control tools that allow the tracking and merging of changes by many different people. The free and low-cost circuit design tools used by most of the people designing Arduino derivatives don't provide automated methods for viewing or merging changes. Without these capabilities, the process of proposing changes to the design of an Arduino board is one that in many ways fails to take advantage of its digital nature. That is, you describe the change you'd

like to see and rely on the original designer to re-create it for themselves, if desired, rather than providing a digital encoding of the change that can be automatically previewed and merged. This lack of easy methods for merging the efforts of multiple individuals reduces the viability of peer production for electronic circuits.

A final obstacle to centralized collaboration is the complications relating to the role of money and business in the production of hardware. When developers have to invest money and time in making and testing changes to the design of a circuit, they may be motivated to recoup those investments by selling their own version of a product rather than contributing their changes back to the original producer. This can create confusion around identity and branding as well, as it can become challenging to distinguish between boards of similar or identical design from different producers. As a result of these complications, Arduino has trademarked the Arduino name, using it to identify only products made by the company. This decision was initially contentious but since seems to have become an accepted practice.

The Arduino ecosystem also points out the importance of open sourcing the complements to the hardware itself. Because the Arduino software is open source (as are its underlying software tools), it gives the makers of derivatives a platform that people can use to program their boards. This factor has slowly pushed the Arduino software to become ever more general; it originally supported only a single AVR processor, then spread to most of the AVR product line, and now can support multiple processors with completely different architectures. This provides a uniform, centralized software platform for the whole ecosystem of derivatives. It also allows others to customize the software along with the hardware, adopting it to both specific uses and available resources. Online documentation (especially if liberally licensed) also makes it easier to support a new board, as that product doesn't need to be documented from scratch. This open source software and documentation, combined with accessible circuit board fabrication and electronic components, together yields a healthy ecosystem of Arduino derivatives and alternatives.

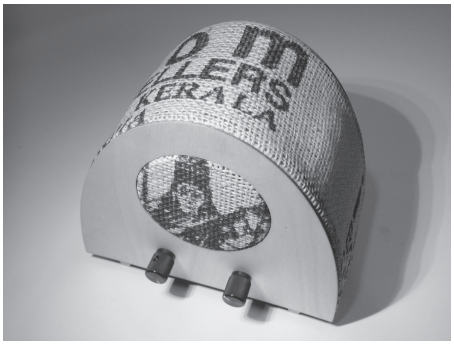
It's not clear what the relative importance of these various factors has been in promoting the vibrant Arduino ecosystem that exists today. Certainly, the Arduino software is more sophisticated (and, therefore, would be more difficult to re-create from scratch) than the basic Arduino circuits—and, in many cases, the derivative circuit designs have been re-created from scratch rather than derived from the files for the original Arduino. In theory, if the Arduino software were simply flexible and extensible, but not actually open source, it could still support a variety of derivatives of the Arduino hardware. In practice, it seems clear that many of the improvements that have been contributed to the software (including those for better support of third-party hardware) have relied in various ways on the fact that the code for the software is available. It's hard to guess which kinds of extensibility people will need, and we probably would have done a bad job if we had tried to predict those directions; instead, individuals have been able to modify the Arduino software in whatever ways they needed and the most useful of these changes have been merged back into the main codebase.

In short, it's difficult to separate electronic devices from the software that works with them. On the one hand, open sourcing just the hardware limits the modifications that can

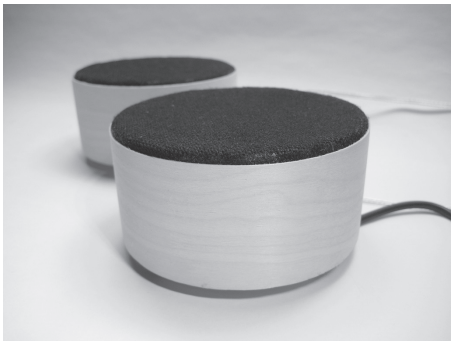
be made (without requiring reimplementation of the software). On the other hand, if the software is open source but the hardware isn't, it might not be clear that derivative designs are allowed. Thus open sourcing the hardware can help make it clear that it's acceptable to create derivatives of or add-ons to an electronic device. In addition, the design files can serve as a de facto specification and reference, facilitating the creation of compatible products. In general, the more aspects of a device's design are shared, the more likely it seems that others will reproduce or modify it.

Case Study: Open Source Consumer Electronic Products

While Arduino has demonstrated that open source hardware can create a thriving ecosystem, it's sobering to note that the vast majority of devices that people use remain proprietary. In my research at the MIT Media Lab, I've been researching the possibilities for people to build devices for use in their daily lives. I started with well-known consumer electronic products: a radio, speakers, a mouse, and, most recently, a cellphone (Figure 11.1).

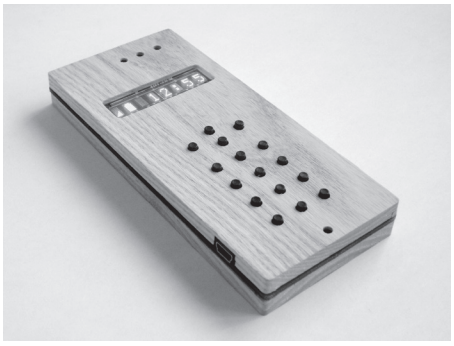


(a)



(b)

(c)



(d)

Figure 11.1 (a) A radio. (b) Speakers. (c) A mouse. (d) A cellphone. (Source: Images CC-BY 2.0 David A. Mellis)

I design the products, prototype them, and use them in my life. In workshops, I help others to make and modify the products for themselves. In general, I try to start from technologies that are accessible to individuals and find ways to put them together into robust and attractive devices. This requires integrating the enclosure, electronic circuit, and embedded software into a complete product—and doing so in a way that lends itself to replication and modification by other individuals. It also means designing specifically for personal fabrication, which has very different opportunities and constraints than the mass production that creates most of our electronic devices.

Electronics

For the electronic aspects of a device, determining the core functionality, interface, and components is an important first step in the design process. Knowing the parts that will compose a device gives a general sense of the required form and shapes the specifics of the electronic circuit. For example, in the speakers, the decision to use three AAA batteries as the power source placed constraints on the size and shape of the speakers and on the design of amplification circuit. Component selection is also crucial for mass-produced devices, of course, but many additional constraints apply when designing devices for personal fabrication. The components have to be available to individuals and possible to assemble without expensive machines or processes. Often, there are limited possibilities available, especially for key components. The radio, mouse, and cellphone all have, at their core, an electronic part that performs much of the basic functioning of the device (receiving radio signals, interpreting the mouse movements, or communicating with the cellular network). For all three, I've had problems finding or maintaining a supply of these core components: the radio receiver and mouse chip that I used have since become unavailable and the cellphone module may become obsolete as cellular networks are upgraded.

This experience points out the essential role that industry can play in DIY: the components it makes available shape the devices that individuals can make for themselves. (There are efforts to produce open source or DIY implementations of core technologies like microcontrollers and cellular baseband modules but, in general, these don't yet seem to offer a feasible alternative to commercial components.) For these reasons, the personal fabrication or DIY process is perhaps better viewed as an individual's ability to assemble the available technologies into a desired product rather than the ability to make everything oneself, from scratch.

The limitations on component selection need to be considered when designing the circuit. For example, with the cellphone, I had to carefully balance the functional requirements against the overall size of the circuit board to yield a usable device. This imposed severe limitations on the functionality: the screen (an LED matrix) shows only eight characters at a time and there's no headphone jack, loudspeaker, removable storage, or many other common features. Even so, the phone can send and receive calls and text messages, keep time, and function as an alarm clock, which is enough functionality for me to have used it as my main phone for the past year. Cramming in more functionality may have made the device too big or fragile to actually use.

Being selective about the functions I needed (and being able to choose them for myself) allowed me to find a compromise that worked for me. In addition, because I faced

similar constraints on component selection as many other hobbyists, I ended up using components for which I could download open source libraries. I was also often able to find existing circuit designs incorporating the same parts, which I could use as a reference in designing my own board.

Developing successive versions of my devices has also shown me that it's important to design your circuit boards with iteration in mind. Building extra flexibility into a design speeds up the process by making it possible to try out different forms and functions without having to fabricate a new board. For example, breaking out additional microcontroller pins, allowing for multiple types of power, and providing different mounting options can allow the board to be used in new and unexpected ways. Another approach that's sometimes useful is to provide a footprint on the board for some parts but not actually solder them on unless they're needed. These development techniques mean that you can try out new variations on a device's form and function without have to wait for a new circuit board to be fabricated, assembled, and tested.

Finally, while the design files for a device capture the components selected, they don't necessarily document the requirements or tradeoffs that led to those decisions, which may make it more difficult for others to create their own modifications of the device. In the cellphone, for example, I restricted the circuit to components shorter than 6 mm so that they'd fit within the laser-cut enclosure. This decision isn't shown in the circuit's design file but is an important constraint on the components that can be used.

Enclosures

Most of my devices have been housed in cases made on the laser cutter. This has allowed me to use natural materials, such as wood and fabric, that are rarely seen in commercial devices. It has, however, required finding clever ways to combine the flat pieces made by the laser cutter into three-dimensional objects. The radio and speakers use two parallel laser-cut plywood faces connected by struts. The faces are then wrapped with another material (either fabric or veneer). For the cellphone, I've sandwiched the circuit board with two pieces of plywood and then covered them with veneer. (In general, I'm not a fan of the finger-jointed boxes found in many laser-cut projects.) All of the designs have fairly simple contours, making it fast to laser-cut them. That constraint has allowed me to quickly iterate through designs by actually making them and seeing how well the parts fit together and how they relate to the electronics. Because the parts are designed in a simple, open source 2D drawing software (Inkscape), they are relatively easy for someone else to modify, whether by simply adding personal text to be engraved or by changing the overall form.

Another approach, which I used for my computer mouse, is to model the circuit board in 3D (using Rhino or similar software) and then use it as a reference when designing the enclosure. This strategy takes advantage of the relative flexibility of 3D printing and the resulting ability to visualize the desired object in software. In addition, 3D-printing parts can be relatively slow, particularly if you are using a high-end machine via an online service. By working in CAD before printing the enclosure, it's possible to experiment with various designs and iterate on their form and relationship to the electronics. Because this

process uses more sophisticated 3D modeling software, it tends to be more difficult for a novice to modify the design of the enclosure, even in a simple way. Conversely, experts can capture more of their work in the 3D model, potentially achieving greater leverage of their skills as they share those models with others.

Assembly

I've tried to take advantage of the manual assembly required for my devices by using this step as an opportunity to engage people in their design and production. The radio and speakers include a fabric element that can be chosen by the individual making the device, giving it a unique appearance and personal significance. Other users, particularly those with prior CAD experience, have created more distinctive variations on the design of the products—creating an owl-shaped pair of speakers, in one case, or producing cellphone enclosures from a variety of materials. Assembling a device offers an opportunity and an engaging context for learning or practicing various skills, such as soldering or hand work. Many of the participants in my workshops are motivated by the desire to create a finished device but, in the process, gain experience with and appreciation for the skills involved in the process. In addition, the mere fact of putting an object together for oneself can invest it with a meaning not present for purchased products.

General Principles

The case studies suggest that there is more to making an open source hardware project successful than simply sharing its design files. These guidelines attempt to distill their lessons in ways that can be applied to other open source hardware efforts:

- *Use standard parts and materials (in conjunction with your open source design files).* For others to make use of an open source design, they need to be able to get the parts that it relies on, whether those are electronic components, screws, stock material, or something else. The more standard and widely available the parts you use are, the easier it will be for someone else to reproduce your design. That might require foregoing components that are convenient for you if they're not available to others. Note that this guideline is in some ways opposed to some quick prototyping techniques, which may favor the materials at hand regardless of their future availability.
- *Understand and design for the fabrication process used.* Different fabrication processes are good for different things—and they also have different processes and constraints. By designing for a specific fabrication process, you can take advantage of its strengths, avoid its weaknesses, and optimize for its parameters. Be specific: different kinds of 3D printers have very different possibilities, as do different stock materials that you might cut with a laser cutter or CNC machine. Working with a particular machine or process as you iterate on your design allows you to learn the capabilities of the machine and ensure that your designs are compatible with it. Of course, other people trying to reproduce your design might not have access to exactly the same machine or process, so try to find ways to avoid relying too heavily on individual quirks or features. Pay special attention to the tolerances of your chosen

fabrication process. Don't create designs that rely on a precision that's not possible to reliably achieve with the machine (e.g., if you have to laser-cut 10 parts to get 2 that actually work, you might want to rethink your design). Hand-soldering is not a particularly exact process; when designing enclosures for a circuit, remember that some components may not end up exactly where the design file specifies they should.

- *Pursue unique meanings, functions, and aesthetics.* The power and efficiency of mass production make it difficult to compete with this approach on its own terms. Instead, try to find unique values for your open source devices. Those might come from solving a problem that's of interest to only a small group of people, albeit possibly of great value to them. It might mean using unusual materials or aesthetics to differentiate your devices in ways that might not appeal to a mainstream consumer but might be appealing to someone looking for an alternative. Or the unique value might simply flow from finding ways to meaningfully involve individuals in the production of the devices. Take advantage of the fact that personal fabrication allows you to make devices in small quantities to find audiences that aren't well served by existing commercial products.
- *Find ways to make iteration faster, cheaper, and easier.* A key benefit of digital fabrication is that every part it produces can be different. To take full advantage of this ability, find ways to iterate on your design rapidly. Getting direct access to a laser cutter, for example, might mean you can try out a few designs in an afternoon instead of waiting a week or two to get a single one in the mail. Similarly, having the electronic components on hand to solder them to a newly fabricated circuit board will allow you test that board more quickly and update its design accordingly. Identify the biggest barrier or barriers to iteration and try to find ways to remove them, whether by getting hands-on access to a machine, using software tools to refine your design before fabricating it, or being able to modify or update a part after it's been made.
- *Open source the complements to the hardware itself.* Someone who wants to re-create or modify your design will likely need more than just a raw CAD file. Provide whatever additional information seems likely to be useful—for example, parts lists, assembly instructions, firmware, and user documentation. Furthermore, by providing the original sources for these additional resources (not just compiled binaries or hard-to-edit documents like PDFs), you enable others to update them together with your hardware files when creating new variations on a design.
- *Clearly distinguish between open source design files and the products based on them.* Selling a physical product is very different from sharing a hardware design file, even if the former is based on the latter. Someone who buys a product may have higher expectations for its functionality, reliability, and safety than someone who makes a device for himself or herself based on your design. If you make and sell products based on someone else's design, be sure to distinguish between the two, making it clear that the product is from you but giving credit to the original designer.

Questions for the Future

Even if we continue to improve our practices along the lines suggested in the general principles, it's not clear what the future holds for open source hardware and personal manufacturing. The pace at which technologies of digital fabrication and embedded computation are evolving shows few signs of slowing down (notwithstanding the impossibility of the exponential growth of Moore's law continuing forever). The extent to which these improvements will extend the capability of individuals and the possibilities for open source hardware, however, is not so easy to predict. Here are three questions about the future of open source hardware and personal manufacturing—questions that I hope will encourage us to think about the future we'd like to see and to work toward making it a reality:

- *Will the technologies that can be made by individuals keep pace with those produced by large companies?* Although technology continues to improve, it doesn't necessarily do so in ways that are accessible to everyone. As a result, it's unclear to what extent open source, DIY, and peer production will be able to keep up with the devices that are produced and sold by large companies. While the potential scope of open source hardware continues to expand as technology improves, the gap between it and proprietary products may limit the extent to which it can serve as a feasible substitute for them. We should remember that the decisions we make influence the potential scope of open source hardware. If we encourage manufacturers to make their technologies available, support open tools, make use of open standards, and make our own hardware open source, we can expand that extent to which individuals are able to create, modify, and control the technologies they use in their lives.
- *Will peer production of open source hardware improve?* Although there are exceptions, open source hardware currently seems less likely than other domains (e.g., open source software) to involve collaboration between many individuals on a centralized design or repository, in which small contributions are combined together into a complex whole. Although there are many reasons for this pattern, if open source hardware is to thrive, it seems crucial to facilitate better collaboration between large numbers of distributed and diverse individuals. This will require improved tools, more efficient processes, and, perhaps most importantly, a focus on fostering communities that have a shared interest in the development of open source hardware.
- *Will the culture of open source hardware expand to include new people and applications?* Although digital fabrication and embedded computation allow for a wide variety of activities and outputs, it's easy to get caught up in the technologies themselves as opposed to their many contexts and applications. For early adopters, an interest in the technology itself can be helpful, as its uses may not be immediately clear or accessible. Even so, this emphasis on technology for technology's sake will not appeal to everyone. Thus, as we think about the future of open source hardware, we should remember to not just play with the technology, but also find ways to make it relevant and useful to new people and situations. In part, this evolution may happen

naturally as technologies mature and we come to take them for granted but it also relies on those of us with early access to and expertise in technology to think about how to make it relevant and useful to others.

Depending on the answers to these questions, the future of open source hardware and personal manufacturing may look very different. My hope is that we will find ways to make them increasingly relevant and valuable, by expanding the technologies they can make use of, the collaborations that can produce them, and the applications and contexts to which they can be applied. If our practices can keep pace with the growth of technology, open source hardware should offer a powerful alternative to mass production for the technology in our lives.

Summary

Personal fabrication offers a potential alternative to mass production for the creation of hardware. This requires effective use of the available fabrication processes, such as 3D printing, laser cutting, and printed circuit board fabrication. As the Arduino case study demonstrates, the success of open source hardware also depends on a number of other factors, like the complements of the hardware itself and the business decision of various actors. Bringing open source hardware into our daily lives (as I try to do in my research) requires careful design of the devices themselves and the process of involving people in their production. Personal fabrication suggests new design principles. The effectiveness of these, and broader questions about the future of technology, will determine the extent to which digital technology can make peer production feasible as an alternative to mass production for hardware.