**Taking Seriously the Material-Semiotic: critical making as practice, as theory.**

Matt Ratto ([Matt.ratto@utoronto.ca](mailto:Matt.ratto@utoronto.ca))

Faculty of Information

University of Toronto

<http://semaphore.utoronto.ca> <http://www.criticalmaking.com>

set up general symmetry and then do arduino tutorial using this precept. Then go back and reflect on what was included and not. Reparse some aspects through sociology of translation and agential cut in order to see how framings differ. Main goal is to see how genres of technical training itself needs to be reframed in order to increase critical apprehension. Digital is main site for this given important role it currently plays in understanding the line between the human and the non-human

**Introduction**

Over the last few years, the long-standing ‘firewall’ between critical thinking and physical making has been toppling, in no small part due to scholars and practitioners in design, the digital humanities, and other areas who have developed specific modes of material/conceptual engagement. While most critical work within the academy remains focused on linguistic processes and outputs, “critical making” (broadly construed) is more and more finding a place within pedagogical and research-oriented contexts.

However, in order to develop critical making, there remains a need to overcome simplistic understandings both in terms of a perceived aporia between language and the physical world and an associated assumed immediacy regarding relations between language and critique. On one hand, we need to be clear that material engagements can both support traditional process of critique and critical thinking and, at the same time, constitute forms of critique in and of themselves. But on the other hand we should not fall into realist material determinisms, naive materialisms or vitalistic perspectives where it is assumed that materials speak for and by themselves in ways that are more ‘true’ than our linguistic representations.

The pivot point upon which critical making turns is therefore nothing more or less than the old questions regarding relations between the natural world and the social world that (and perhaps I am projecting here) attracted many of us to Science and Technology Studies. The criticality of making stands upon a return to the ‘material-semiotic’, but with potentially novel resources for addressing the relations named not by either term, but by the hyphen that connects them. Unlike Latour, I do not think the hyphen is the ‘nail in the coffin’ (1999) of apparatus like ‘actor-network theory’ or ‘material-semiotic actors’ but instead constitutes the actual site of these theories’ criticality.

While there are many ways to establish the ‘critical-ness’ of making, my particular interest is to what degree material engagements gives us new modes for exploring the entanglements of the material and the semiotic and the ways these entanglements support or trouble what Marcuse called ‘the forces of liberation”. (1969)

**‘generalized symmetry’, the material-semiotic, the ‘agential cut’, and more**

A main impetus for the authors of the sociology of translation (Callon, 1986; Latour) (part of the theoretical apparatus of ‘actor-network theory’) was the principle of ‘generalized symmetry’. This principle acted as an extension of the original notion of symmetry described by David Bloor (1976). Bloor’s symmetry, linked to the ‘strong programme’ in sociology of science, simply noted that the same reasons needed to be given to explain both the success and failure of scientific theories. Instead of relating ‘false’ theories of science as due to social factors, and ‘true’ theories of science as due to ‘nature’, holding to the symmetry principle requires scholars to see both true and false claims as based in a combination of ‘social’ and ‘natural’ factors. Callon’s extension of symmetry (1986) holds that not only must the researcher not use pre-existing ‘natural’ and ‘social’ distinctions in explaining the outcomes of scientific work, but equally they need to not change registers when they move from technical to social aspects of the problem being studied. An outcome of such a commitment is that the researcher “…must abandon all a priori distinctions between natural and social events,” and see these divisions themselves as “… topics for actors’ discussions”(4). :

Instead of imposing a pre-established grid of analysis upon these, the observer follows the actors in order to identify the manner in which these define and associate the different elements by which they build and explain their world, whether it be social or natural. (Ibid)

The point here, that actors use definitions and associations between the natural and the social as part of the ways they ‘build and explain their world,’ is an extremely important one. Latour (1991) later described modernity itself as being predicated on the double moves of ‘purification’ whereby the natural and the social are made distinct, and ‘hybridization’ whereby they are reconnected. Similarly, Haraway’s original (1991) and subsequent (1992) uses of the phrase ‘material-semiotic actor’ were intended, among other things, to reconnect material and discursive articulations of nature and society, and to reestablish relations between the common-place separation of the two. For Haraway, as for Foucault (1979), the body stands as a paramount site to explore such relations. Haraway notes:

…bodies as objects of knowledge are materialsemiotic generative nodes. Their boundaries materialize in social interaction among humans and non- humans, including the machines and other instruments that mediate exchanges at crucial interfaces and that function as delegates for other actors' functions and purposes. (1992)

For Haraway, “…accountability for techno-bio-politics in the belly of the monster.” (Ibid. 298) requires recovering the status of bodies as ‘materialized’ in and through social interactions that include humans and machines. A central conceit by Haraway as for Callon and Latour is that an assumed dividing line between material/nature and social/discursive is a site where where individual bodies and facts are made to conform, where knowledge is legitimated, and where worlds are built. Materialist feminist theorist Rosemary Hennessey has put it clearly:

“a rigorous materialist theory of the body cannot stop with the assertion that the body is always discursively constructed. It also needs to explain how the discursive construction of

the body is related to nondiscursive practices in ways that vary widely from one social formation to another” (1993, 46), quoted in Barad, 2003: 810)”

Barad’s contribution to this issue focuses attention on the performance of the ‘agential cut’, the specific moment when the ontological indeterminancy between the natural and the social is resolved in order to produce ‘objectivity’. Rather than speak of hybrids or networks of ‘humans’ and ‘non-humans’, Barad directs our attention to the ‘apparatus’ through which both types of entities are temporarily produced. Barad’s use of this term signals her connection to scientific practice and the ‘material-discursive’ ‘apparatus through which measurements take place – such as Bohr’s use of a diffraction grating apparatus for measuring the wave quality of light. But it also performs a relation to Foucault’s articulation of ‘apparatus’ as an ‘analytic of power’ that is linked to a specific historical moment .(Foucault, 1980).

**How Matter comes to Matter**

How can and should we examine these relations and the ways in which they work to build worlds (Callon), create accountability (Harway) or link to social formations (Hennessey)? It is certainly the case that critical studies in STS, Feminist, queer studies, materialist studies, etc. have been doing exactly this for some time. What remains to be done? Suchman’s ‘reconfigured’ version of her original text ‘Plans and Situated Actions’ (2006) provides some potential direction. First, Suchman notes how ‘non-human agency’ has often stood in for questions regarding the relations between nature and society. Following Casper (1994) she asks us to reframe such questions “…from categorical debates to empirical investigations of the concrete practices through which categories of human and nonhuman are mobilized.” (1) Suchman thus provides a link between nature/social and human/non-human that mirrors similar trends in the work of Haraway, Barad, and others. For me, this link explains my personal attention to computational technologies as sites where human/non-human relations are currently articulated in ways that matter. While this is certainly historically contingent, and such a focus may soon be displaced by other sites of techno-scientific practice (such as that of genetic engineering), computational technologies and their deployment into many novel aspects of the life world are an important critical focus.

Equally, Suchman asks us to focus in on the ‘concrete practices’ through which divisions between human/non-human are made to stand. This prompt follows many similar imperatives within STS at least, to ‘follow the actors’, to study everyday activities, to address mundane practices. But Suchman gives an additional prompt for what we might call a more ‘participatory’ approach:

Engaging in a series of iterative attempts to enact a practice of small-scale, case-based codesign, aimed at creating new configurations of information technologies, has left me with a more concrete and embodied sense of both problems and possibilities in reconfiguring relations and practices of professional system design. (6)

The original text to Plans and Situated Actions primarily involved ethnographic observation and examination, watching people at work, taking notes, and analyzing the results. Here in this quote, Suchman describes some of the work she has done in the intervening years. She highlights processes of creating ‘new configuration of information technologies’ and the ways in which such practices have left her with a more “embodied sense’ of the difficulties and the potentials involved in creating alternatives to current practices. I see in Suchman’s quick description clear links to others within design, the digital humanities, or STS engaged in similar practices. Johanna Drucker’s ‘speculative computing’ (cite), Carl Disalvo’s ‘adversarial design’, or my own work (cite) all share an emphasis on reflection through material engagements that reconnect the lived experiences of making technologies with a critical/ conceptual register. I see the term ‘making’ as having appropriate resonance here, given its somewhat mundane sensibility - despite the more recent neoliberal overtones of the term (CF Doughterty, Anderson).

**Critical Making: An exercise**

We are used to scholarship on technology that does not seem to require much knowledge of the material characteristics of technical systems, replacing such knowledge with what I have in the past called ‘textual doppelgangers’ (Ratto, 2012; 2014). ‘Critical makers’ (as a general category) have begun to incorporate more technical background and understandings within their work. But for the most part, such work requires that they learn how to engage with technical systems using the pedagogies and practices drawn from technical fields. In particular, the genre of the tutorial or the technical instruction carries with it particular ways to parse the world into technical and non-technical domains, including what counts as human and what as non-human. Given the critical importance of such divisions, it is important to look at such trainings in order to better understand how current engagements with the technical may be hindering critical approaches.

In my teaching and research I frequently make use of the arduino microcontroller system as do many others interested in physical computation systems and pedaogogy (cites here recommended by reviewers.) But I often struggle with available tutorials around the Arduino that typically focus on learning some ‘basic’ skills. As has been made clear by others (CF Agre, 1997) moments of technical training are rife with what has been called ‘socialization.’ (cites) We might in fact go so far as to describe technical training as a primary site when standard ideas of technology and concommittment values of instrumentality and rationality are reified. To reparse this based on some of the concepts described above, we can say that moments of technical training are sites where particular divisions between ‘nature’ and ‘the social’ are produced and then naturalized. It therefore becomes important as scholars to start a critical process within technical training, rather than see such training as an a-priori part of critical forms of making. Among other elements, it seems crucial to open to consideration the moments when particular ‘cuts’ are made which work to reproduce humans and non-humans.

In the remainder of this section I will walk through a basic tutorial on the Arduino system, providing the information necessary to carry out the specific steps that make the arduino operate. I will move between two different registers within this process (pace Callon!), typographically separating them in order to call attention to the shifts. I see the writing below as a sort of experiment. While still retaining the training element that will result in readers without previous experience to be able to ‘make the arduino work’, I also want to selectively pull out moments when particular articulations of the social and the natural are being drawn and when specific cuts are being made that work to (re)produce the arduino as apparatus.

**Arduino Tutorial**

*Many Arduino tutorials start with some historical background and description in order to help situate the novice user within the system. Such histories typically involve an origin story, a description of the device and its user base, and an overview of what is it and how it works. Such background descriptions provide important clues for new users who often need guidance in parsing not just the functional capacities of the object but also how it fits into past and present developments. I draw the story below from a range of similar histories, including those written initially by its original developers.*

Originally developed by interaction designers looking for a less expensive way to prototype digital/physical interactions, the arduino is a physical electronics board and accompanying software development environment that facilitates connections between the digital and the physical world. It allows developers to easily connect input components such as temperature sensors and buttons or output actuators like motors and lights and to program how these function using software code. The arduino can act as a stand-alone device or it can serve as an input/output device for a desktop or laptop computer. It has been widely adopted by artists and designers looking for simple ways to create interactive objects, spaces, and exhibitions, but also by engineers and scientists for prototyping and early development.

The arduino system is typically considered as having three main technical components; first the hardware board design that provides reliable power, a microcontroller chip, and a USB interface for programming and access; second, a software IDE or Integrated Development Environment, that provides a way to write and debug code and upload this code onto the arduino hardware; and third, software code running on the arduino hardware called a ‘bootloader’ that makes it possible for the IDE to program the arduino hardware. Importantly, the developers of the Arduino have licensed the electronic design of the arduino hardware as open source (Creative Commons CC-SA-BY License), the IDE software is licensed under the GPL and the bootloader under the LGPL free software licenses.

As many have noted, the arduino’s technical capabilities are matched or exceeded by a range of alternatives. Before the arduino became popular, many computer science and electronic engineering experimenters used the BASIC Stamp, a microcontroller and environment developed by Parallax, Inc. that provided many of the same capabilities as the arduino. Similarly, both the Lego Mindstorms (link) and the Phidgets (link) systems have been used for digital/physical experimentation. Explaining the arduino’s popularity, despite these alternatives helps unpack it as more than simply the sum of its technical components.

There are a number of reasons for the popularity of the arduino platform. First, the hardware and software design make learning and use fairly easy - the hardware is mostly based on commodity parts and is inexpensive, all software is available online, and many tutorials are available to help beginners get started. Second, the focus by the original developers of the arduino was on encouraging adoption by designers rather than engineers. Therefore, early example code and the creation of the IDE itself focused on simple functions rather than technical sophistication. Third, the community of current users is very active and, due in part to the open source nature of arduino, many tutorials and sample projects, including wiring diagrams and code are online and available to be repurposed.

*These aspects help explain the popularity of the platform but also reveals the ‘arduino’ as more than just a singular technical artifact. Instead, it can better be understood as the focal point of a number of technical, political, discursive and social moves, as a network, assemblage, or constellation. (Latour, Callon, Akrich) Actor-Network Theorists originally noted the multiplicity of objects and the ways in which they necessarily involve a range of ‘actants’, including both human and non-human elements. Importantly, ANT scholars also highlighted the instability of objects and demonstrated the ongoing work that is required for them to cohere and maintain the semblance of durability. Latour, for example, initially described this process as a ‘trial by strength’ in which ‘actor-networks’ compete to ‘enroll’ others and thereby secure their own consistency as a ‘fact’ or ‘artifact’. (Latour and Callon) Alternatively, from the viewpoint of agential realism (Barad, 2003) we might describe the arduino as a phenomena and its particular material instantiations as the result of material-discursive practices that work to ‘cut’ it in ways that highlight particular attributes.*

*Such insights would not come as any sort of surprise to the majority of arduino users. There is, I believe, a strong recognition of the multiplicity/complexity of the arduino by its users. Few would claim that the arduino would be the arduino if licensed under a more restrictive license or without the strong community of developers and their willingness to share. Most recognize that the arduino is not simply what we might reductively consider its technical parts - electronic components and software code - but include social attributes as well. Reading the arduino through the lens of ANT or of agential realism is simplified by the public nature of the work involved in maintaining it as an object. Unlike many other technical systems which actively seek to hide the ongoing labor of their constructed nature, the work of managing the objectness of arduino and its appearance of consistency is done at open conferences, in public forums, and in fierce debates where motives, licenses, political discourse, and technical facts are deployed with little differentiation.*

To carry out this tutorial, you need to purchase or otherwise secure an Arduino device, a USB cable, a red LED, and a green LED. If you went to a local or online electronics store you might have noted that there are many different arduinos, compatibles and clones made by a variety of different vendors. The original developers of the Arduino have licensed the electronic design of the arduino as open source (Creative Commons CC-SA-BY License) and made the actual design files available online. This has allowed many others to develop their own often customized versions. There are many such versions, developed and sold (often alongside the original arduinos) by various vendors. For example, Sparkfun electronics (<http://www.sparkfun.com>) sells approximately 16 different arduino and arduino compatible boards, including boards developed and manufactured by the official arduino developers as well as derivatives developed and made in house. Similarly, Adafruit (<http://www.arduino.cc>) sells both official arduinos and their own arduino compatible versions. Both are considered official distributors of arduino and are listed as such on the distributor page at <http://arduino.cc/en/Main/Buy>. Any Arduino will do for this tutorial.

*The openness of the arduino electronic design files has also allowed a number of other companies to develop direct copies of the official Arduino boards. In a blog post titled ‘send in the clones’ from 2010, one of the original developers, Massimo Banzi distinguished four different ‘types’ of arduino variations that are not an official arduino board, clones, derivatives, compatibles and counterfeits. While encouraging the development of derivatives such as the Sparkfun and Adafruit boards noted above, Banzi highlighted problems with the three other variations. Banzi and the other developers have maintained trademark control over the Arduino name, logo, and graphic design on the boards in an attempt to differentiate the quality of their boards from others and to prevent pure clones and counterfeits of their work. They specifically discourage others from reproducing the graphics and logo in an attempt to pass clone boards off as originals and will leverage trademark law in order to do so. Importantly, Banzi also described what constitutes an actual arduino:*

*An Arduino is a board which*

*it’s directly supported by the official Arduino IDE*

*it follows the Arduino layout we have standardised*

*it’s properly documented on our website*

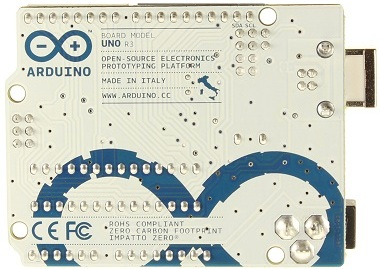
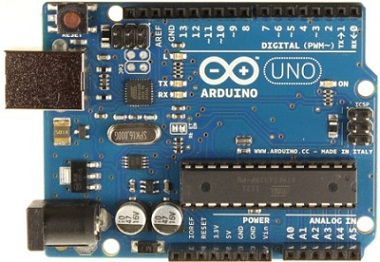
*it’s properly licensed to bear the Arduino name and logo*

*it’s made by authorized manufacturers (Banzi, 2010,* [*http://blog.arduino.cc/2013/07/10/send-in-the-clones/*](http://blog.arduino.cc/2013/07/10/send-in-the-clones/)*, accessed July 29,2013)*

*Here again we can see the public nature of the socio-technical work that stabilizes arduino as arduino. For Banzi and the other developers, the arduino is both a technical object that follows a standardized arduino layout and utilizes official arduino software, but is also named (by them), ‘properly licensed’, and made by manufactures who have been authorized. Banzi’s description of the Arduino can be parsed via the material-semiotic ontology described above - note his language here - ‘follows the Arduino layout’ not ‘’has’ or ‘contains’ the layout, “properly licensed’ not ‘has’ or ‘contains’ the arduino license. In each case, Banzi is describing material-discursive practices that work to stabilize attributes by ‘cutting’ the phenomena of ‘arduino’ into two aspects - the arduino and the layout, and the arduino and the license. While it is possible to view the Arduino as a network that ‘contains’ hardware, software, licenses, layout, and documentation, such a description overly stabilizes each element and does not do credit to the intra actions through which they are constituted.*

*In deciding what arduino board to purchase, you are participating in the process by which the arduino is stabilized. The choice of what to obtain does not involve simply ‘technical’ attributes, but engages you, actively or passively, in decisions that engage social, political, and legal considerations. Do you purchase an non-authorized arduino clone that, even if it is legal according to trademark law, is seen by the original developers as ‘not an arduino’? Do you purchase an actual counterfeit - or more importantly, do you spend the time to make sure the arduino you are purchasing is not a counterfeit?⁠2 Even in the sourcing of your components you are participating in processes of stabilization (or unstabilization.) We might carry out a similar process of reflection regarding the other parts of this exercise, including the USB cable and PC but we might hard pressed to do so. Both of these objects are equally the result of complex legal, discursive, and technical engagements, though unlike the Arduino example they are much less available to us for review and critique.⁠3*

Once you have decided and obtained your arduino (and the other equipment), take a few minutes to look at it. (For simplicity’s sake, my comments in the remainder of this experience will focus on the Arduino UNO. Your version may differ in some ways from my description.)



You will see a blue-green (front) and white (back) circuit board, with a number of different components on the front of the board. The most important part is the AVR microcontroller, the black rectangle located at the lower slight right of the board. This is the main component of the arduino, its ‘brain’ which provides the logic, memory, and input/output functions. A microcontroller is like an entire computer but in a single chip. Like a laptop of desktop computer, a microcontroller contains a processor core, static memory (like a hard drive) and volatile memory (like RAM), as well as ways to accept input (like a keyboard or mouse) and provide output (like a screen.) Other important parts of the arduino board include the barrel connector located on the bottom left (black cylinder) which allows you to connect battery or wired power connector, and the USB port (silver rectangle) which allows you to connect it to a computer for programming and other communications. Finally, the top and bottom of the board have female headers (rows of black sockets) which allow you to connect input devices such as switches, buttons, and sensors, or output devices such as lights or motors.

The board is also covered on the front with white letters, numbers, and logos, intending to, among other things, provide instructions to users. In particular, the top row of female headers is labeled ‘Digital’ and each hole is associated with a number or other label. Similarly, the bottom role of female headers is labeled ‘Power’ and ‘Analog In’ with labels such as ‘A0’ or ‘GND’ associated with each hole. Other visual elements include the Arduino logo, a web link, and the words ‘made in italy’. On the back, you can see the silver solder points which each of the components on the board are attached to the board traces that connect everything together. These traces are themselves visible on both the front and back of the board as a series of lines running between components. Also, words such as ‘open source electronics prototyping platform’ and ‘ZERO CARBON FOOTPRINT’ declare the environmental status of the device.

*Even in such mundane details as the visual elements on the Arduino board we can see the hybrid social and technical attributes that characterize it. Visually, solder joins and pin descriptors merge somewhat seamlessly with socio-legal positions ‘open source electronics’, nationalistic agendas ‘made in Italy’, and concerns for the environment. By incorporating these attributes as part of the Arduino, the developers are reparsing what counts as ‘technological’. AVR microcontrollers, environmental sensitivity, national (or at least anti-off shore) manufacturing are ‘cut together’ in ways that redefine what counts.*

**Downloading and Installing the Software**

Go to <http://arduino.cc/en/Guide/HomePage> and follow the ‘Getting Started’ instructions.⁠4 Note that there are different procedures depending on whether your PC is running Windows, Linux, or MacOSX. These instructions will walk you through the following steps:

1. downloading, installing the software and any necessary drivers, launching the IDE application.

2. connecting the board and setting the IDE environment correctly

5. Opening example code and uploading it to your board.

First, the process for installing the software and drivers can be somewhat difficult. If you are running Windows, you must follow the driver installation process to the letter. This requires letting Windows try and fail to find and install the appropriate drivers before moving on to a manual process. If you need additional assistance, try looking at a different tutorial for installing the drivers, such as the one at <http://learn.adafruit.com/lesson-0-getting-started/installing-arduino-windows>. Also, I have found that many MacOSX users fail to copy the application to their ‘Applications’ folder and, instead, try to run it from the disk image or desktop. While this mostly works, you may sometime encounter strange errors including an inability to upload code to your arduino board.

*The issues related to Microsoft Windows and Arduino drivers point to continuing controversies over what counts as legitimate technologies. Current versions of Microsoft Windows software (Windows 7, 8, and 10) require that the drivers for hardware devices are authorized by Microsoft through the use of a ‘digital signature’. For reasons related to the ‘open’ nature of the Arduino platform (open both in terms of hardware and software design as well as open in terms of potential use), such a digital signature has not yet been obtained. This means that making the Arduino work on the Windows system typically requires additional steps to turn off the need for digital signatures for installed software. While framed as a technical debate about security, both the technical and the discursive outcomes work to frame what is ‘proper’ and ‘appropriate’ behavior.*

Second, after launching the Arduino IDE, it is important to correctly set the Arduino IDE to your board and serial port. As noted above, there are many different Arduino and Arduino compatible boards. In order for the IDE to work properly, it must be told what board you are using. This step is described in the guide listed above, but is a step that is sometimes missed. Similarly, the IDE needs to be told which serial port on your laptop or desktop is connected to your arduino board. It can sometimes be difficult to figure out which of the serial ports listed under the ’Tools/Serial Port’ menu item in the IDE corresponds to your Arduino board. The simplest way I have found to discover which it is, is to look under the Tools/Serial Port menu in the IDE **with your Arduino unplugged,** and make a note of what is there. Then, plug your Arduino into the USB and look again at the Tools/Serial Port menu item. Select the new item that you see there and you should be good to go. If you do not see any change, check to see if you properly installed the drivers.

*A recent controversy has risen between the original developers of the Arduino platform and the original manufacturer over who owns the rights to the Arduino name. The result has been a series of lawsuits, and a range of both discursive and material actions. Depending on where you sourced your ‘official’ Arduino, you may encounter one such move at this stage of the tutorial. If you purchased your Arduino from Arduino.org (or from one of their vendors), upon plugging in you will see a pop-up window with the following text:*

*This is not an official arduino – (replace with actual text)*

*The developers of the Arduino IDE – Arduino.cc – have added commands within the IDE to look for and respond to hardware ID numbers that are associated with the devices produced by Arduino.org. This move stands in addition to marketing materials and public presentations aimed at reinforcing the authority of Arduino.cc as related to that of Arduino.org. Unlike these discursive moves, the pop-up window feels as if it conveys greater authority, in no small part due to the fact that it appears to be the IDE – the technology itself – giving you this warning.*

Third, how do you know whether you are successful? Following the instructions in the guide requires you to open example code - File > Examples > 1.Basics > Blink and, following successful configuration, to click the ‘upload’ button (the right facing arrow icon). This kicks off a complex process involving the following steps;

1) the example code you selected (the ‘blink’ program) is compiled, e.g. translated from human-readable source code into machine code;

2) the arduino IDE sends a command over the USB cable to the bootloader software running on the Arduino board asking it if it is ready to receive a program;

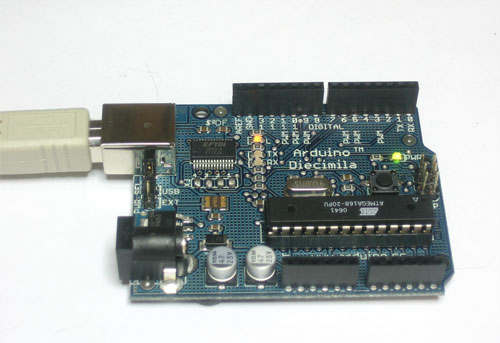
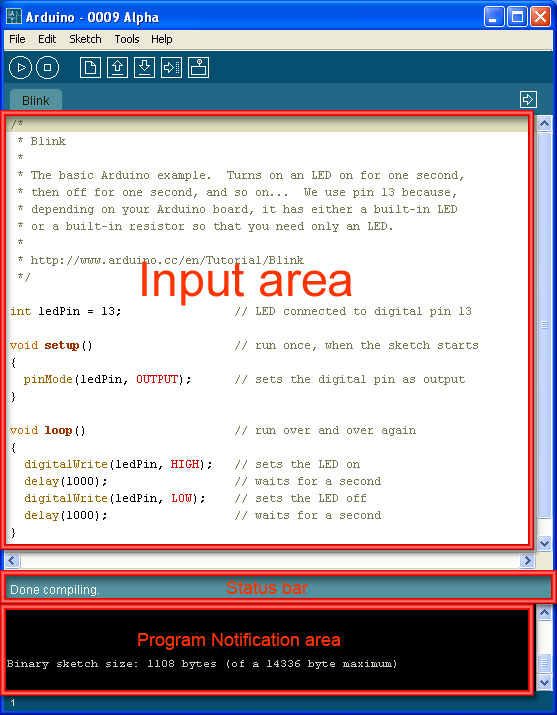
3) the bootloader responds that it is ready;

4) the Arduino IDE streams the machine code to the arduino board;

5) the bootloader saves the code to static memory, notices when it is done, and reboots the arduino;

6) the arduino board executes the code.

Traces of these steps can be seen if you look carefully. The status bar will show ‘done compiling’ and the program notification area will respond with the size of the machine code as ‘Binary Sketch size” if the compilation and uploading is successful. Also, you can see the upload request, response, and the streaming of the machine code by watching the serial transmit (TX) and serial Receive (RX) lights on the arduino board.⁠5 Note here that tracing this process requires you to move between the software IDE and the hardware board, to look for error messages on a screen as well as physical changes (lights going on and off) in order to understand what is going on.



*If you have successfully installed and uploaded code to your arduino board - congratulations! The steps involved are somewhat complex and involve operations and activities that are not entirely familiar to non-developers. It can be difficult to trouble-shoot problems when they occur, since the systems works at a number of different levels - software IDE and drivers, hardware board and serial ports - and making the whole chain work together requires you to think across these levels and to figure out where the problem occurs. Success can be as difficult to recognize as failure and the process of observation requires moving between the parsing of debug messages from the software as well as close observation of the hardware. You are obviously interpolated within this system since much depends on your ability to figure out where you should be looking and what requires your attention.*

**Conclusion**

What might you have learned from this encounter with the arduino system? First, you hopefully have a better sense of the complexity of technological objects. As the arduino example highlights, technical features alone do not determine a technological artifact. Instead, they are better addressed as complex phenomena and as resulting from and maintained by material-discusrive practices through which the political, the technical, and the social are depicts and deployed. Further, you have participated in this process through the decision-making you engaged in when sourcing your arduino, an experience which highlights our complicity even as consumers in the complex arrangements through which objects are stabilized and maintained.

Although it seems obvious to state, your own engagement with the arduino was markedly different if you actually obtained the parts listed above and attempted to carry out the above steps, rather than reading them and imagining yourself doing so. The experiences of finding an arduino and the other parts, downloading and installing code, and working through the example code make use of different physical and cognitive resources than the apprehension and act of imagining yourself engaged in such operations. Even if your experience matched act for act and cognitive operation to cognitive operation the depiction of the process in the text above, your phenomenological experience would be different. Of course, your experience probably did not match all that well, given the reductive way the text and images above depicted what you were expected to accomplish. You might have borrowed an arduino from a friend and had a heated discussion regarding its status as ‘official’ or not, or ordered one from an online vendor through a complex process involving drop down menus, credit cards, and postal codes. The phrase ‘obtain an arduino’ could not hope to capture all the possible complexities attending such a process. You might have had difficulty connecting the arduino to the USB cable and to your PC because of the resistance of the cable and the way it can flip the arduino around. The phrase ‘plug it in’ could not hope to capture the complicated physical moves you needed to carry out to accomplish this.

One way to parse these differences is by noting the differences between tacit and explicit knowledge and, following Polyani and others (Polyani, 1958, 1966; Lam, 1996, Nonaka and Takeuchi, 1995; Collins, 2001) highlight the difficulties in formally and explicitly articulating even simple activities such as using a hammer to hit a nail. More over, we might emphasize the need for a ‘knowing subject’ and practical and shared experiences in order to engage in activities requiring tacit knowledge. We can also affirm the importance of an individual’s commitment and relation to the context of action (Nonaka, 1994:21) as other key elements in making sure we can move easily between these two levels. We do not need to defend the importance of tacit knowledge or the value of explicit articulations - that has already been done for us. But critical value can be found by examining how these levels are established and maintained, what kinds of information are made explicit within relevant communities, and how such practices work to normalize certain activities and denormalize others. But leveraging definitions of tacit/explicit do little to explain the relevance and importance of the differences between your experiences and the textual and pictorial descriptions above, particularly since we are not interested in coherence between explicit descriptions and tacit experience. Instead, we need to focus on the differences and incommensurable aspects, given our goal is to better understand how the physical, embodied engagement supplements and extends critical reflections that are typically considered through explication and formal means.

Despite my best attempts, the arduino critical making experience described above does not entirely steer clear of reductive separations of ‘social’ and ‘technical’ due in part to a desire to make the process comprehensible - requiring a certain obedience to genres and processes associated with technical work. I have noted above that Arduino developers and users do recognize the politics and social nature of the Arduino object and that they engage in this work as part of the process by and through which the Arduino is constituted as Arduino. However, it is clear that the ‘social and political’ work of Arduino is kept separate from the ‘technical’ work. The tutorials linked above and, in fact, all tutorials that I have seen that teaches how to use and develop with the Arduino, provide resources for carrying out technical operations - writing code, installing drivers and libraries, and assembling hardware. Breaking out of this domain to engage in subjective reflection (e.g. ‘How do you feel when…’) or political critique seems foreign precisely because of “the whole tacit system of intellectual procedures” (Agre, 1997) associated with the technical disciplines.

Most experiences with the arduino are bounded in part by this reliance on a genre of technical writing - the tutorial which instantiated realist commitments, leaving little room for social reflection or thought that does not fit within an instrumental logic. Without acknowledging the limits of this genre as it is typically performed, it is difficult to move beyond the strong social and technical separations that are a core component of ontological realism. However, upon reflection one might use our experiences butting up against this limit to analyse the obdurancy of material-discursive practices and to start to analyze how social formations and objects are co-constructed through specific ‘agential cuts’ (Barad) that create and normalize relations between discursive and non-discursive practices. In this sense, the obdurancy of the ontological commitments of technical work and related constructivist educational forms acts both as something to be overcome but also as a prompt for better understanding social formations marked by these forms of work.

The above tutorial engages but also extends upon the typical technical tutorial. It walked you through a technical process, makes use of example code, and even uses the ubiquitous ‘hello world’ of physical computing, the blinking of an LED. However, it also intersperses this technical focus with prompts for other kinds of reflection, asking you to think about the social and political work associated with the Arduino licenses and to examine your own feelings and emotions regarding your engagement with the device. Such extensions into the realm of the affectual and the political demonstrate that it is possible to break genre conventions and associated realist commitments to open the ground for alternative ontologies.

This seems critically important. In our move to incorporate making as part of a critical repertoire we need to avoid the material romanticism and individualism that often colors this work. You may now have some knowledge with this issue, potentially having experienced the heady glow of successful Arduino blinking. It is tempting to bask in the power of technical work and to be captured by the sense of control and agency that it offers. But making in and of itself is not enough (for a critical maker.) Instead, one must remain cognizant of the ultimate purpose of the work and be ready to move outside the individual moment of making to engage with other scholars and thinkers. Like Gauntlett I believe that all making can be critical making, but it requires moving outside the material romanticism often associated with making to address the larger socio-material issues.



1 I will return to the value of the maker movement for critical making later in this chapter. For now, suffice it to say that the openness and willing to share expertise

2 This turns out to be relatively difficult to do, given the ways most arduino counterfeits mimic the design characteristics of official arduinos. See here for guidance by the official developers regarding counterfeits - <http://arduino.cc/en/Products/Counterfeit>.

3 Though note the emergence of this complexity when controversy arises. For example, USB as a social, technical, and institutional standard is revealed by cases involving the emergence of new standards and the ways tech companies jockey for power regarding these developments, e.g. Intel and USB 3.0 specifications.

4 If you need additional assistance, there are a variety of excellent online tutorials that can walk you through the process of installing the Arduino software. My personal favorites are the tutorials up at http://learn.adafruit.com.

5 images pulled from arduino.cc and adafruit guides.