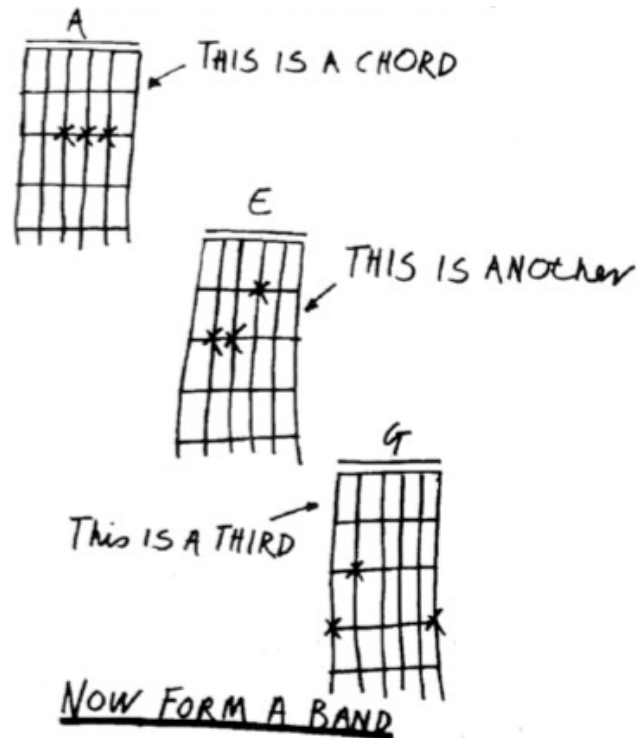


Fast, Cheap, and Out of Control:

Interactive Prototyping with Arduino

Tod Corlett, Philadelphia University

Mike Leonard, Philadelphia University



Cover of the punk 'zine *Sniffin' Glue* (1977, detail), excerpted in Banzi et al, p. 6.¹ The creators of Arduino promote a design ethos of enthusiasm and trial-and-error over efficiency and perfection, which allows it to make technology very approachable to nonengineers.

Recent developments like the iPhone demonstrate the near-future potential of software interaction with electronic systems. The screen becomes a virtual keyboard, a virtual album shelf, a virtual camera with a virtual shutter noise. However, these software-driven, highly converged, multifunctional systems also offer a sobering challenge for industrial designers. As an object, the iPhone is beautiful, but intentionally featureless. Following an increasingly popular strategy, it and its competitors offer maximum flexibility to software designers by keeping the physical interface as nonspecific—that is, as generic—as possible.

There is little question that intelligent, deeply interactive systems mediated by computer code are here to stay; the question is whether that particular part of the future will involve industrial designers in a meaningful way.

In a short time, designers may actually have to answer the question “Why should this object have physical features?” about the things they’re tasked with designing. They must lead in proposing new meanings and roles for actual physical objects, or they run the risk that their work will be reduced to simple substrates and frames for interaction that happens primarily on a screen.

If they are to retain any relevance working in this context, then, it's important that industrial design students graduate with the ability to:

- work effectively and creatively with digital interface designers, and
- propose, demonstrate and test compelling new physical interfaces with computer-mediated interaction, rather than just theorizing or simulating them.

Unfortunately, ID education today is bound by the fact that it still takes years to teach students to design traditional objects, much less intelligent systems. To teach students to effectively address the “iPhone Problem,” three issues have to be addressed:

- The Iteration Problem:

How do you allow students to “sketch” interactive systems as quickly and flexibly as they do static objects?

- Framing the Issues:

How can one convince students to uncover new potential in the technology, rather than borrowing or replicating existing solutions?

- Meaningful Collaboration:

How can a student multidisciplinary team blur disciplinary boundaries to create new value, rather than simply defining a hard line between their own areas of expertise?

The balance of this paper will be devoted to showing how each of these questions can be effectively addressed with a powerful new teaching and design tool: the Arduino microcontroller.

A Background of Frustration

In February of 2007, after several months of sourcing, preparation, and faculty training, I and my colleagues Mike Leonard (Industrial Design) and Sherman Finch (MS Digital Design) brought our students together to launch a collaborative project with digital design and industrial design students working on collaborative teams to design interactive software/hardware systems.

This sort of project had been tried before by some of us, several times, and it had never worked well.

In previous years, the project brief had been very specific and product-oriented from the beginning, to control the size of the design task. For instance:

Design a hardware/software system to do the following:

- allow retrieval of a contact from a database
- dial a number
- take a picture
- email it to another contact

Although everyone agreed that existing systems that performed these tasks were cumbersome, thoughtless and inelegant, and although everyone, students and faculty, wanted to create radical new ways to work, the student solutions proposed tended to converge rapidly toward conventionality as the project progressed.

- Screens were made rectilinear. This was easier to delegate to the digital designers, and it was difficult to test the advantages of other shapes or displays, so they were voted out.
- User choices were represented by menu items arranged hierarchically. The only way to “sketch” the interaction without building it was to diagram it, and tree structures were simple to diagram.

- Controls were momentary-contact buttons. Auditory feedback was “beeps.” In engineering parlance, these were established standards; in more literary terminology, they were clichés. Lacking tools to prove or demonstrate the potential of alternatives, established standards were seen as safe choices.
- Once these conventional menu structures, controls, and experiences were adopted, in many cases the industrial designers quickly found themselves left to design Scroll Up, Scroll Down, Select and Back, plus a couple of context-determined option buttons, a screen bezel, a charging socket and a battery hatch. Everyone’s good intentions had led to another completely conventional object.

While one could argue that it was instructive for students to recapitulate the history of bad choices that led to so many of the objects we use every day, it was hoped that Arduino could help in bypassing these design pathologies which made the previous projects so unproductive. We particularly wanted to try an approach called “sketching in hardware.”²

Rapid Interactive Sketching: The Hardware

Design, being an iterative process, requires that potential solutions to a problem be tried or tested and assessed, so they can be improved. In designing interactive systems, this means designers need to build and program actual electronic hardware, and then change the hardware and software swiftly and inexpensively as needs and ideas shift. These changes should be in the hands of the students, rather than the instructors; they should be able to happen at two in the morning, when inspiration strikes. In the words of MIT’s Rodney Brooks, the process needs to be “fast, cheap, and out of control”³.

Of course, this is difficult to achieve, given the limited time and financial resources of most ID programs. In the past, systems for prototyping electronic interaction fell into two categories:

- Closed proprietary systems, such as Lego Mindstorms, Teleo, and VEX. While offering easy access to the world of programming, sensors, and computer-mediated interaction, these were essentially high-end toys. They offered limited opportunities for customization and were generally expensive, with a simple starter package starting at more than \$300. Modifying the capabilities of the “kit” involved cutting up and breaking these expensive components, something starving students were reluctant to do.
- “Engineering-based” solutions, which required students to gain extensive knowledge of electronics and programming before they could begin to design with standard “production-level” electronic components.

Arduino is an alternative—a cheap, open-source microcontroller which can be used to quickly prototype interaction with electronic systems. ID students can come up to speed in a few days, and the system can be used either to communicate with computers and Internet-based applications, or to create standalone “smart product” prototypes.

For interaction prototyping, Arduino offers a couple of key advantages:

- Very low cost. The microcontroller itself costs approximately \$30 US. A complete kit of parts, with which students could create an intelligent mobile robot, had they need to, can be assembled for \$110.
- Extreme flexibility. After a few lessons in basic digital electronics, students can begin adding and changing the system with parts from Radio Shack, or for that matter from their own pockets.

The Tools

Arduino is a system with two major parts:

- A small (2 x 3 inch) circuit board. This board can be programmed to act (and react) on its own, or it can act as a customizable input or output for another computer.

- A software environment and programming language, which allows users to tell the board how to act. When connected to a computer, either by USB cable or wireless link, the board can also communicate with interactive development tools like Flash, Director, Max/MSP, and an interesting open-source graphics environment called Processing.

Arduino was developed by a multinational team of academics and physical-computing enthusiasts, and it's open source, which means no one person or corporate entity controls the intellectual property. The software and hardware work with almost any computer platform; students can download and install the software without charge. The boards are made and sold by several vendors, which helps keep prices down and improvements frequent. New versions of the software are released every few weeks or months, as issues are resolved and features are added. (Open-source software can initially seem like a risk to skeptical university IT departments, but reliable open-source applications like the Firefox browser have recently given them more confidence.)

All this technology is educationally important for two primary reasons:

- It both affords and enforces reality in the design of interactive systems. If previously all sounds were “beeps,” Arduino allows students to explore the experiential difference between a trill, a moan and a chirp by constructing each sound. As a bonus, at the end of the exercise they've fully specified the sound in terms engineers would understand.

- It's fast and easy to change, like a sketch. One of the first things demonstrated in learning Arduino is programming the board to flash an LED on and off. Within a few minutes, as soon as they understand the principle, students immediately begin to modify durations and intensities, fading lights up and down, blinking them in accelerating and decelerating patterns. The “sleep light” on Apple computers, so celebrated at the time of its invention, quickly appears obvious to Arduino-using students. Why would one *not* specify an LED blink that “feels like sleep” for a sleep function? What else would one do?

Design, especially interaction design, is based in experience; without being able to create experience, design languishes because it's impossible to tell good ideas from promising bad ones. With Arduino, students really learned the difference it made in their project when the system waved a flag rather than blinking a screen, and they were able to implement the change in about an hour.

Having taken these steps to ensure that the students could prove out what they proposed, we then put some procedures in place in hope they could propose ideas really worthy of building.

Framing the Issues: The New Process

Rather than a very specific project goal, we chose a nebulous one: to convey emotion through gesture. This overall goal was broken up into subprojects, each with its own brief.

First of all, it was realized that interesting, iconoclastic design work would only result if students felt safe and confident making unusual choices. Thus, we structured the project with initial exercises that built confidence, then ones which forced and then rewarded unusual thinking; only then did we turn to the central design task.

Phase 1, Workshops: The nine participating digital design graduate students were allowed to form seven teams, as were the 21 students in the industrial design classes. Each team was issued a kit of parts including an Arduino board, a solderless breadboard, and other small electronic parts.

In class, they were taught to:

- Solder and connect electronic components
- Mix colors using LEDs
- Compose and play tunes through a speaker
- Sense distances with an IR ranger
- Control servos through software
- Drive motors
- React to switches
- Detect varying light levels
- Communicate with Flash, Director, and Processing
- Control lights on an Arduino board from another computer
- Control a program on a computer by rotating knobs attached to the Arduino board.

The Arduino technology workshops occupied class time; it's a testament to the accessibility and simplicity of the system that this only required three weeks. Class lessons, with pictures and software code, were accessible on the course web site, so students could always reiterate anything that had seemed unclear in class. Outside class time, the teams were asked to think about the project brief by defining and conveying an emotion through gesture, in the form of a 30-second edited video. Then they were asked to edit the video to three seconds, losing as little of the power of the experience as possible.

Phase 2, Do Something Cool:

In this assignment, the teams were asked only to demonstrate "something cool"; to use the Arduino hardware to do something they'd never seen before, something that would surprise and amaze the rest of the class.

Projects included:

- An electronic flute that detected the shadows of the player's fingers
- A toy gun that detected, located, and fired at passersby.
- Telepresence miniature golf.

The Arduino system allowed students to really try things out, and this unleashed tremendous creativity. It also opened up the potential for serendipitous discoveries by doing things the "wrong" way. One team got a music signal mixed up with their motor-control current, and fed it to the motor anyway; the motor windings resonated, and resulted in a usable motor that actually whistled a tune while it worked.

After only a few weeks of work, the students were able to realize, as working systems, things they would have called science fiction before the beginning of the project. This had a salutary effect on their confidence and their brainstorming; they were much less prone to dismiss ideas as unrealistic, having proved to themselves and each other how ambitious they could be without failing.

Once the seven digital design teams and the seven industrial design teams had demonstrated their projects, they were allowed to combine into seven multidisciplinary teams, based on common issues addressed in their projects, and take on the next assignment.

Phase 3, Link Gesture and Emotion Interactively:

In this assignment the students were asked to work together for the first time, and to choose an emotion. They could then choose to construct a system that would either:

- Evoke that emotion in the user through gesture, or
 - "Understand" and respond to the emotion when it was expressed by the user.
- The resulting systems went through two rounds of iteration, growing more effective and reliable. With Arduino, the students were able to "release early and often," and evolve experiences by testing and changing them.

Phase 4, Propose a Real-World Application:

Having developed an effective piece of interactive emotional technology, the teams were now asked to propose a practical application for it. Polemical applications and games as well as practical applications were allowed, but they all had to have written criteria detailing what user reactions were to be counted as successful.

Phase 5, Build:

The students had two weeks to build working autonomous demos of their systems, with hardware durable enough to survive a four-hour end-of-semester exhibition. On the day in question, April 30, all seven worked. They included the following:

- Real Mario World: A recorded version of the side-scroll Nintendo game in which the user plays as Mario, running to run, jumping to jump, and watching the game landscape roll by on a life-sized screen. Exhausting, but fun.

- **Watching:** Users explore a landscape made entirely of texts on surveillance, while others watch surveillance video of these users, processed so the images themselves are made up of text. Spooky and much more visceral than an afternoon talking about Derrida.
- **Graffiti Ball:** Users can throw splashes of paint, giant refrigerator magnets, or “bombs” at a projected wall surface by bouncing a sensor-equipped ball off the wall.

With a few exceptions, the technology involved in these systems was simple, in the final analysis. Most of the hardware boiled down to switches or motors of one kind or another, and the software was a combination of a few sensor-tuning algorithms and garden-variety Flash. The technology was, however, used very effectively. The fact that the technology was all available and had been repeatedly demonstrated made the students confident in employing it wherever necessary; conversely, the fact that they knew they had to follow through with their own engineering, rather than handing it off to someone else, made them prudent and efficient in deploying electronics.

Meaningful Collaboration: Mind the Gap

In previous interdisciplinary projects, one of the most troubling failures was the tendency of the industrial designers and digital designers to divide up the work immediately, creating disciplinary “silos” and not really collaborating effectively. This was understandable, on the principle that strong fences make good neighbors. The industrial designers saw themselves as designers of objects, and the digital designers saw themselves as designers of linked on-screen graphics, and the two disciplines were failing to learn very much that was new from being placed together. We found Arduino to be helpful in reducing this resistance to collaboration.

- **It’s All in the Code: Working Backward to the Objects**

Because much of the effort in making an Arduino-enabled project work lies in writing the software that makes it behave correctly, and because coding was new to both the industrial and many of the digital designers, the teams quickly found themselves having to work on common tasks that lay in between their disciplines. They also realized that the code determining the performance of the system had to be put in place, tested and evolved before the appearance of the physical objects or screen graphics could be addressed, so an unprecedented amount of time was spent on real, meaningful collaboration.

- **A New Worldwide Community**

One of the factors that makes Arduino so flexible and effective as a learning tool is the global community of other students, professional designers, hobbyists and dilettantes who are also using the system, expanding its capabilities, and, most importantly, posting their results, circuit diagrams and source code on the internet. If a student wants to make an Arduino board control a recycled stepper motor, all she needs to do is type “Arduino stepper” into Google to find a full tutorial. This has the effect of further speeding development. Students don’t have to do all the work themselves; they can find something as similar as possible somewhere else, and modify or expand it to suit their needs. A system for controlling Google Earth by stepping on floor switches⁴, for instance, can easily be turned into hardware for computerized dance lessons.

Perhaps more important, however, contact with this larger group of Arduino users has allowed and encouraged the industrial and digital designers to think of themselves as part of that community, as “physical computing designers” with much more to gain than to lose by looking at, working on and improving the ideas of others. Far from being just a circuit board, Arduino is a movement, and it’s one that offers some profound life lessons to many design students and designers still working in isolation.

Further Reading

Kurt, Tod E. “Spooky Projects: Introduction to Microcontrollers with Arduino” parts 1–4. Online PDFs. www.todbot.com/blog/spookyarduino

References

(1) Banzi, M., et al. "Getting Started with Arduino- Beta Version". Online PDF.
www.arduino.cc/en/Booklet/Homepage

(2) *Sketching in Hardware*. Conference theme, Henry Ford Museum, Dearborn, MI, June 24-25, 2006.
<http://sketching06.com/>

(3) "Fast, Cheap, and Out of Control: A Robot Invasion of the Solar System"
Journal of the British Interplanetary Society, Vol. 42, pp 478–485, 1989
Online PDF at
people.csail.mit.edu/brooks/papers/fast-cheap.pdf

(4) Glaeser, T. "Google Earthwalk with Arduino at Berlin Design Mai 2007" Online video.
www.youtube.com/watch?v=zoNwJ931aql