Beyond Open Source Software: Solving Common Library Problems Using the Open Source Hardware Arduino Platform

Jonathan T. Younker
Head, Library Systems and Technologies
Brock University
jyounker@brocku.ca

Tim Ribaric
Digital Services Librarian
Brock University
tribaric@brocku.ca

Abstract

Using open source hardware platforms like the Arduino, libraries have the ability to quickly and inexpensively prototype custom hardware solutions to common library problems.

The authors present the Arduino environment, what it is, what it does, and how it was used at the James A. Gibson Library at Brock University to create a production portable barcode-scanning utility for in-house use statistics collection as well as a prototype for a service desk statistics tabulation program's hardware interface.

Keywords

Arduino; open source hardware; open source software; hardware development; barcode scanning; in-house use; service desk statistics

* * *

Many librarians have experience evaluating, testing, installing, and/or configuring open source software packages to accomplish a variety of library-related tasks and functions. There are open source catalogues (e.g., Evergreen and Koha), discovery layers (e.g., VUFind, Project Blacklight) as well as a myriad of library-related scripts, plug-ins, modules, etc., that underpin the everyday operations of all types of libraries. As libraries face increasingly harsher budgets, open source software may allow for greater freedom, control, and customizability, more reasonable upfront costs, and the ability to find creative solutions to common problems.

But what of hardware? While it is relatively straightforward to find and test open source software packages, especially in this era of hosted, cloud-based, and virtualized services, what if the solution involves a heretofore unrealized physical gadget? Or what

if the necessary hardware exists but at an exorbitant cost? A simple walkthrough of a trade show floor at any major library conference yields a multitude of refined and elegant library hardware solutions, such as self-check machines, automated check-in systems, laser/RFID scanners, security gates, etc. — each with a correspondingly refined and elegant price (if the prices are divulged to the inquirer at all without a sales call or local product demonstration). But what if the purchased commercial product needs to be modified to perform adequately in the library's environment? Will the company perform the modification and, if so, at what cost? Perhaps the needed hardware solution is much more esoteric or has not been commercially produced. What if there were easy-to-use, readily available, low-cost, *open* tools that would allow libraries to quickly create prototypes and production hardware solutions using freely available open source code?

Enter Arduino (<http://arduino.cc>). Developed in 2005, the Arduino prototyping platform allows beginners and experienced users alike to quickly and easily create and deploy hardware to accomplish specific tasks. Need to modify your security gates to allow the data to be inserted into a web-accessible database? Arduino can help with that. Need to create a self-check machine that interfaces with your patron API? Create a prototype with an Arduino at a fraction of the cost of a commercial product.

In this article, we will describe how the Arduino was initially conceived and developed, how it works, what it does, how it can be used in libraries, its limitations and drawbacks, and how we used the Arduino platform in two specific library-related projects: a barcode scanning 'in-house use' statistics tracking device (dubbed The Barcodinator) and a service desk statistics tabulation program's hardware interface (dubbed The Tabulatron). Along the way, we hope that staff in libraries are encouraged to broaden their skill sets to include open source hardware tools like the Arduino, alongside open source software, when confronted with challenges requiring creative solutions.

What is Arduino?

The Arduino platform is a low-cost, easy-to-use, single-board microcontroller. The Arduino excels at rapid prototyping: users of all skill levels can get started using the on-board inputs and outputs to quickly connect LEDs, sensors, motors, etc., to interact with the real-world with a low barrier for entry. Originally conceived in 2005 in Italy by Interaction Design Institute Ivrea (IDII) Associate Professor Massimo Banzi for his interactive design students, Arduinos quickly became popular with hobbyists, makers, and artists for their projects. Arduinos come in many shapes, models, and sizes, and since the project is under the open source umbrella, there are a multitude of third party kits and plans that incorporate modifications and specializations. You can buy official Arduinos (the 'Arduino' name is trademarked) fully assembled and tested, usually for less than C\$30, or you can build your own from components using the freely-available schematics. As of 2011, over 250,000 official Arduinos have been sold worldwide (Kushner n.p.).

Despite the popularity of the platform, at the time of writing there were no articles in the library and information science literature about the Arduino microcontroller or open source hardware development in libraries, with the exception of the authors' technical construction/code article in *Code4Lib Journal*. While there are academic articles written about hardware development using the Arduino platform in a science context (Pearce, 3103), and Arduinos are available for loan at public and academic libraries (e.g., Steacie Science and Engineering Library at York University and Innisfil Public Library), there is nothing to indicate that other libraries are using these tools to create their own custom hardware solutions.

How it Works

There are two main parts to using an Arduino: the hardware and the software. Both the hardware and software are open source. The hardware is the physical Arduino board (Figure 1), which has the microcontroller chip, power/USB connections, digital and analogue input/output ports, LED, etc., while the software is what tells the Arduino what to do.

Photograph by SparkFun Electronics. Used under the Creative Commons Attribution Share-Alike 3.0 license.

Figure 1: the Arduino board

The software, a C++-like programming language, is written in the Arduino integrated development environment (IDE) (Figure 2), and is then compiled and uploaded to the Arduino via a standard USB cable which also supplies power.

Figure 2: the Arduino integrated development environment (IDE)

Arduino programs are called 'sketches' and, at their most basic, can be as simple as blinking the onboard LED (the Arduino equivalent to the ubiquitous 'Hello, World!' code in most programming languages). Once the sketch has been uploaded to the Arduino, the USB cable can be removed and the Arduino can be powered via a 9-volt battery in standalone operation.

The Arduino can also be tethered via USB to a computer (as we will see in The Tabulatron project below) to send information back and forth from the computer. In this scenario, we need another piece of software that operates on the information coming from the Arduino. This software is called Processing (<http://processing.org>) and has its own IDE. Processing sketches can be graphical and can add a level of user-interactivity to Arduino projects, although the two are independent of each other: Processing sketches can operate with or without Arduinos, and vice-versa. Processing sketches are

simply another tool to be used with the Arduino based on the needs and goals addressed by the specific project at hand.

What Can You Do With It?

Arduinos have been used to create LED cubes, robots, autonomous aircraft (<http://diydrones.com>), electronic instruments, animatronics, interactive art displays, toys, etc. In addition to the Arduino's hobbyist audience, it has slowly been gaining traction in academic environments from the laboratory to the classroom. The Arduino can be used to fabricate specific computer solutions for a variety of applications — often with a scientific focus — using a combination of off-the-shelf components available from electronics stores and specialized equipment suppliers. For example, an Arduino-based LED stimulator system for vision research was constructed at the Stem Cell and Brain Research Institute at Université Claude Bernard. The motivation for the project was “to use off-the shelf and open-source components as much as possible and to reduce design complexity allowing use of the system by end-users without advanced electronics skills” (Teikari 227). Arduinos have also been used at the Georgia Institute of Technology where an Arduino-powered laboratory apparatus was constructed for a series of biomechanical experiments. The Arduino was a strong fit for their needs as “this microcontroller development board is shown to be an inexpensive, precise, and easy-to-use tool” (Kornuta 183). The Arduino has also proven itself as an educational tool: it sits as the core component of a laboratory kit used by the Technological Educational Institute of Piraeus in Greece, at which it is “to be used in undergraduate university courses concerned with electrical engineering and physics” (Zachariadou 1599).

Why Would I Use This in a Library?

The Arduino has become popular with hobbyists and makers and has found a place in the academic environment as both a tool to accomplish specific tasks and a device that can be used for teaching purposes. What about the library environment? The most obvious place for Arduinos in the library is behind the circulation desk where they can be loaned out to patrons for their own discovery, construction, and educational needs. They can also be used as cornerstones in library makerspaces. A makerspace is an open environment where users are allowed to experiment with physical technology to learn and be creative — an environment ideally suited for Arduinos (Loertscher 45). The Arduino can be utilized by libraries to improve existing workflows or to bridge gaps in services through the creation of custom hardware with Arduinos at their core. Finally, Arduinos can be used as gateways between the analogue world and the internet. An Arduino project can be constructed to take signals from the environment and convert them into computer-ready information. This is demonstrated in the two Brock University Library Arduino projects outlined below.

The Barcodinator

The Barcodinator: Initial Request

In the summer of 2012, Circulation staff at the James A. Gibson Library at Brock University were embarking on an in-house use statistics program. Instead of relying solely on check-outs to judge the usefulness of physical items, staff were interested in adding statistics to our integrated library system (Millennium by Innovative Interfaces Inc.) when a user had taken an item off the shelf for perusal/consultation. In-house use statistics collection programs are common in libraries but had never been implemented at Brock. In order to facilitate the process when Circulation staff travelled from floor to floor scanning these items, Library Systems and Technologies (Systems) staff were asked to provide a laptop (with a high-capacity battery), a barcode scanner, and a wheeled cart. This presented some immediate challenges as we had no viable laptops available, and the purchase price for a new one (as well as the requisite Kensington lock to guard against theft) and a new scanner would be prohibitive, especially considering the amount of time per day this utility would be used and the Systems staff time required to set up and maintain the laptop. Furthermore, since the goal was to be as efficient as possible for Circulation staff to travel between floors collecting and scanning items waiting to be reshelfed, this large, cumbersome, elevator-only cart with hundreds of dollars of equipment on it seemed extremely inefficient. One of the first options considered was using a mobile application to accomplish the same task. We quickly realized that there were no applications available that would accomplish the needs outlined above, and once the time and costs for custom application development were factored in (not to mention the purchase price of a mobile device for staff use), the costs were prohibitive.

The Barcodinator: Proposed Solution and Prototype

Systems staff had been interested in Arduinos for a number of years, and Jonathan Younker (co-author of this article) had already purchased the hardware components necessary to handle certain aspects of this project, namely the Arduino, an LCD display, the interface between the PS/2 laser scanner and the Arduino, and an Arduino-compatible SD card shield. These components were initially purchased with the intent of creating a hand-held off-line Circulation utility. This utility was to be for use by service desk staff during power failures to record user and item barcodes for uploading into the integrated library system (ILS) when power was restored.

Using code from the Arduino Playground code wiki (<http://playground.arduino.cc/>), we had successfully interfaced the PS/2 laser scanner with the Arduino, and it could read barcodes from our items' Codabar barcodes.

Armed with most (but not all) of the pieces and knowledge needed, we suggested to Circulation staff that we could prototype an equivalent using an Arduino instead of the requested laptop-based 'mobile' barcode scanning utility. The thinking was that if we could not create a functional prototype by a certain date, we would fall back to

purchasing the equipment and resources needed for the laptop-based solution. We internally dubbed this project 'The Barcodinator.' The goal was to create a hand-held, low cost, easy-to-use platform which would scan barcodes using the surplus PS/2 scanner, store them on an SD card, display the last scanned barcode on the two-line LCD, and operate a week at a time between battery charges. Staff would navigate between floors with The Barcodinator, scanning items that had been pulled off the shelves, then upload the barcodes into the ILS once they had returned to the main floor.

Of course, the clock started ticking as soon as the suggestion that we use an Arduino-based solution was accepted. Instead of creating a proof-of-concept utility, we were now creating a prototype of a hardware device that would be used in production on a daily basis (and needed to be ready as soon as possible). We are not electrical engineers, nor have we been trained in electronics. The creators of The Barcodinator (the authors of this article) are librarians with a hobbyist-level interest in Arduinos and open source hardware. Nevertheless, by using freely available code via the Arduino site and elsewhere, some small portions of custom code, a few off-the-shelf hardware components, and a barcode scanner fished out of our redeployment pile, we had a working prototype ready for testing in a matter of days (Figure 3).

Figure 3: the first prototype of The Barcodinator

The Barcodinator still needed an enclosure, and a quick visit to a local dollar store yielded an easily-modified plastic snap-lock food container that fit all of the components inside (padded with Styrofoam) and could be easily opened if anything needed to be adjusted (Figure 4).

Figure 4: the completed Barcodinator

Total costs for the hardware of The Barcodinator were less than C\$50. We tested it with as many barcodes as we could find in Systems and then gave it to the Circulation staff member responsible for in-house use statistics.

The Barcodinator: Issues Encountered and Revisions

The Barcodinator worked. And then it did not. The Circulation staff member came back with the unit, letting us know that it had started 'squealing,' and that it was extremely unreliable when scanning barcodes. We were initially at a loss as all the connections seemed secure, and it did not exhibit the same symptoms when we were testing in the Systems department. Ultimately, we were able to narrow down the squealing to a battery-related issue: we were powering the Arduino and the laser scanner with a regular alkaline 9-volt battery, and the power draw was too much for both to run simultaneously for longer than a few minutes. The squealing noise was the barcode

scanner's initiation beep that was resetting multiple times per second. We purchased a lithium-ion battery and small charging circuit board to connect to the Arduino at an additional cost of approximately C\$30. The power problems immediately went away (along with the squealing); the unit was now functional for roughly a week between charges (an initial goal of the project), and staff could now charge the battery via a standard cell phone charge cable.

The second issue — barcode scanning unreliability — was a little more difficult to diagnose. Everything seemed functional when we would test it, but the staff member was unable to scan more than one book before encountering difficulties. Luckily for us, the problem turned out to be simple: the barcode scanner had a coiled wire, and as the staff member would walk from floor to floor in the Library, the movement and the weight of the coiled wire would unseat the cable's connection. A liberal application of hot-glue inside and outside of the enclosure to secure the cable solved the problem.

The Barcodinator: Production

The Barcodinator has been in use on a daily basis since September 2012. It has accounted for over 3,000 items scanned from September 2012 to January 2013. Although not all staff use The Barcodinator (some prefer to collect materials on the upper floors and then scan them all at once on the main floor, which requires the books to be carted back to their respective floors after scanning), it accounts for approximately 60% of our in-house use statistics. We still consider it a prototype, and the pen-style laser scanner that is used is not ideal for scanning older or copied replacement barcodes, but this was a fantastic project to learn how to leverage the rapid prototyping ability of the Arduino for a real-world library application.

The Barcodinator: Looking Ahead

As the Arduino platform allows different code sketches to be uploaded to the same hardware, this project could be adapted in a number of ways. The same utility could function as the originally-conceived offline circulation utility. It could also be adapted to serve as a self-check unit, communicating via an Ethernet or WIFI shield using the SIP2 protocol to a library's ILS. It could also be leveraged to scan staff/student card barcodes for workshop/class registration or for follow-up after information sessions. This utility could also accommodate both magnetic stripe cards and barcodes, or could easily be adapted to accommodate radio frequency identification (RFID) enabled items via readily-available Arduino shields. Because Arduinos are considered prototyping platforms, we opted not to solder any of our connections; however, using open-source software tools like Fritzing (<http://fritzing.org/>), it would be possible to create a custom printed circuit board (PCB) to reduce the size of the platform and to securely solder all of the connections. At that point, it would also be possible to sell or distribute the product as an assembled solution or a user-buildable kit, although issues surrounding patent investigations, commercialization, distribution, etc., are well beyond the scope of this article.

The Tabulatron

The Tabulatron: Identified Problem

As patrons interact with library staff, the usual last step of the process is for staff to tabulate the interaction. The data captured varies between institutions, but the general idea is that the type of interaction with the user is recorded: traditional reference question, directional question, referral to a librarian, etc. This data is then aggregated and used for reporting purposes to library organizations (such as CARL) or used to aid decision-making about staffing levels at service desks. This tabulation is often done as tick marks on paper sheets affixed to the surface of the desk. Some institutions tabulate these transactions directly into an electronic database, foregoing the paper tally sheet. The open source Libstats package (<https://code.google.com/p/libstats/>) is an example of this type of configuration: a web page can be opened by desk staff, and the transaction is tabulated by picking the appropriate values from a web form, with all results then sent to a database. Both the paper and the electronic methods have limitations and drawbacks. The paper form — while highly visible and accessible to the staff member — requires manual processing time to transfer the data to a database or spreadsheet for further analysis. The software-based solution does not require any further processing, but during busy periods it can easily be overlooked, especially if the software has been minimized or is covered by other software running on the computer. The Tabulatron was constructed as a hybrid solution to this problem: it would have the ease of use of the paper form with the immediate data collection of the software-based alternative.

The Tabulatron: Proposed Solution and Prototype

The goal of The Tabulatron was to create a physical box attached to a PC by a USB cable (for power and communication) with a series of buttons that staff can push to record service desk transactions. The physical construction of The Tabulatron was straightforward and simply required an Arduino, four push buttons, an LED, and an enclosure to contain all of the pieces. Each push button was connected to a different input pin on the Arduino and the LED was connected as an output. This device is then connected to a computer via a USB cable (Figure 5).

Figure 5: The Tabulator

The Arduino is programmed with a sketch that simply listens for a button push and then communicates with the connected computer with the number of the corresponding button that was pressed. For feedback, the LED blinks following the button press. The computer that is connected to the Arduino is running a graphical Processing sketch that is waiting for signals from the signaling device (Figure 6).

Figure 6: The Processing sketch. The green label indicates that it just received a button press for 'Technical' help.

This Processing sketch then increments a value in a database that corresponds to the type of patron question. For prototyping purposes, the device was coded to aggregate values in a Google Spreadsheet. As the Brock Library uses a custom solution for tabulating patron transactions that functions in a very similar way to the LibStats software package, the move to production with the device would require that the Processing sketch communicate with the Brock Library's software via an application programming interface (API). For more technical details, please see the authors' *Code4Lib Journal* article.

The Tabulatron: Construction and Costs

The process and cost of constructing a Tabulatron is very minimal. The most extravagant component in our construction was the prefabricated enclosure. Many electronics stores sell these boxes in various sizes to create a polished look for the hobbyist project. This type of box is not required; an inexpensive plastic container would suffice. A more ambitious library could even drill holes in the top of a service desk and run the appropriate wires to a hidden Arduino board. In our construction, a small amount of soldering was necessary to attach the wires to the leads of the push button switches and the LED. As the contacts of the switches and the diameter of the wire were fairly

large, no intricate work was necessary. The push buttons and LED can easily be purchased for under a dollar apiece. Total cost of fabrication, including enclosure and components (minus the Arduino), was approximately \$20.

The Tabulatron: Looking Ahead

The Tabulatron is a fairly versatile device that can be programmed to accomplish just about any manual counting that is required. It could be modified to be a manual head counting device to keep track of occupancy in a given space. The Tabulatron could also be scaled up by adding up to six buttons — the largest number the Arduino UNO used in this project can support. Larger Arduinos can be purchased that could support up to sixteen different buttons.

In the Brock University Library, interest has been expressed by Circulation staff to have a Tabulatron device available at that service point as a pilot project. The Circulation desk receives a large amount of activity and would benefit from the quick-fire nature of The Tabulatron. With minimal effort, a staff member could quickly enter data via The Tabulatron without the need to be using a computer with a keyboard and mouse to collect the information: they simply press a button.

What it is Not

While discussing the possibilities and successful projects undertaken using Arduinos, it is important to realize that the Arduino platform has limitations. It is not meant to function as a standalone computer system; it is not possible to hook up a keyboard and monitor to use it for general computing tasks — a stark contrast to the Raspberry Pi (<http://www.raspberrypi.org>) which has recently become very popular. The Raspberry Pi is meant to create a full computer experience on a low-cost, highly portable device. It can be loaded with a traditional operating system and used to play games and surf the Internet. The Arduino, on the other hand, should be seen as an appliance that can be programmed (repeatedly, in different configurations, if necessary) to complete a very specific task. This task almost invariably involves collecting signals from sensors connected to the Arduino, computing values and transmitting results to a connected computer (or to other physical output devices). The available onboard memory of Arduinos is typically in the kilobyte range, meaning that this nimble little device can only be pushed to do so much.

Conclusion

There are certainly many different way to accomplish some of the goals outlined in this article. For The Barcodinator, a mobile application could have been developed to scan barcodes and use WiFi to interact directly with the ILS. Raspberry Pi, BeagleBone or other single-board microcontroller platforms could have been used instead at a similar price point and learning curve. With The Tabulatron, simply assigning keyboard shortcut keys could create the same end result, albeit with less flexibility. The advantage of using the Arduino environment for these and similar projects is that there is a tremendous

amount of freely available open source code to address any sort of real-world application. Massimo Banzi's goal of creating an easy-to-use, low cost, and compact platform for prototyping has resulted in thousands upon thousands of Arduino-based projects, from hobbyist-level to academic scientific applications, many with how-to guides and downloadable sketches.

In the library world, we have largely relied on commercial solutions or simply done without. There has been a surge of open source software coming out of all types of libraries over the past decade. While there are different challenges when developing and maintaining open source projects versus their closed, commercial equivalents, the open source hardware platforms (like Arduino) are at a stage now where custom hardware development should be an option to all libraries considering a significant hardware technology purchase.

At the very least, the Arduino platform is a fantastic way for all interested individuals (not just library technologists, makers, and hackers) to go beyond the 'someone should build this' stage into the 'I can build this' phase. A few lines of code, an Arduino, an idea (and a cup of coffee or glass of beer, of course) may yield a solution that could have profound service and budget implications for libraries. Get out there and make something; the future is open.

Resources

- The official Arduino website: <http://arduino.cc>
- The Arduino Playground: <http://playground.arduino.cc>
- Processing: <http://processing.org>
- Fritzing: <http://fritzing.org>
- Instructables (search for 'Arduino'): <http://instructables.com>
- Make Projects (search for 'Arduino'): <http://makeprojects.com/>
- O'Reilly books and resources: <http://oreilly.com/arduino/>

Works Cited

Kornuta, Jeffrey A., Matthew E. Nipper, and J. Brandon Dixon. "Short Communication: Low-Cost Microcontroller Platform for Studying Lymphatic Biomechanics In Vitro." *Journal of Biomechanics* 46 n.d.: 183-186. Web. 26 Feb. 2013.

Kushner, David. "The Making of Arduino." *IEEE Spectrum* 26 Oct. 2011: n.p. Web. 26 Feb. 2013.

Loertscher, David V. "Maker Spaces and the Learning Commons." *Teacher Librarian* 39.6 (2012): 45-46. Web. 27 Feb. 2013.

Ribaric, Tim and Younker, Jonathan. "Arduino-Enabled Patron Interaction Counting." *Code4Lib Journal* 20 (2013): n.p. Web. 5 May 2013.

Pearce, Joshua M. "Building Research Equipment with Free, Open-Source Hardware." *Science* 337 (2012): 3103. Web. 25 Feb. 2013.

Teikari, Petteri, et al. "Basic Neuroscience: An Inexpensive Arduino-Based LED Stimulator System for Vision Research." *Journal of Neuroscience Methods* 211 n.d.: 227-236. Web. 26 Feb. 2013.

Zachariadou, K., K. Yiasemides, and N. Trougakos. "A Low-Cost Computer-Controlled Arduino-Based Educational Laboratory System for Teaching the Fundamentals of Photovoltaic Cells." *European Journal of Physics* 33.6 (2012): 1599-1610. Web. 26 Feb. 2013.