UNIVERSITY OF CALIFORNIA
SANTA CRUZ

LEVERAGING NOVEL TEACHING DOMAINS TOWARD BROADER PARTICIPATION IN COMPUTING

A dissertation submitted in partial satisfaction
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Emily Marie Lovell

December 2021

The Dissertation of Emily Marie Lovell is approved:

_____
Professor James Davis, chair

_____
Assistant Professor David Lee

_____
Professor Janice Pearce

_____
Peter Biehl
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures vi

Abstract viii

Dedication x

Acknowledgements xi

1 | Introduction 1

1.1 Background & Motivation 5

Informal Teaching Experience 5

Formal Teaching Experience 11

1.2 Document Overview 16

2 | Course Design for Attracting Broader Participation: Craft of Computing 19

2.1 Introduction 20

2.2 Related Work 22

2.3 Background 25

2.4 Course Design 26

Setting & Organization 27

Major Assignments 29

Infrastructure 36

Instructional Support 39

2.5 Reflections 40

Challenges & Rewards of Working in the Physical World 40

Importance of Growth Mindset at the CS0 Level 41

Leveling the Playing Field 43

Preparation for CS1 44
Broadening Participation & Perceptions Through Craft 45
Personally Meaningful Work 46
2.6 Recommendations 47
Course Logistics 47
Equipment & Materials 49
File Formatting & Exporting 50
Emphasizing Original Creative Work 51
2.7 Future Work 52
2.8 Summary 53
3 | Course Design for Retaining Broader Participation: Open Source Software Engineering 54
3.1 Introduction 56
3.2 Related Work 57
3.3 Course Overview 59
Term 1 60
Term 2 63
3.4 Student Feedback 65
3.5 Reflections & Recommendations 67
Project Choice 67
Team Formation & Tools to Support Collaboration 68
Structuring Unfamiliar Tools & Technologies 69
Professional Mentorship 69
3.6 Future Work 71
3.7 Summary 72

4 | A Case Study in Expanding Access to Electronic Textiles: The LilyTiny **74**

4.1 Introduction 75

4.2 Related Work 77

Physical Computing 77

Electronic Textiles & Computing Education Research 78

Instructional Design for K-12 STEM 79

Independent Learning Resources for E-Textiles 80

4.3 Design & Development 81

The LilyTiny 81

Companion Curriculum 84

Pilot Testing 87

4.4 Measuring Impact 89

Derivative & Follow-on Products 89

Sales Data 92

Customer Reviews and Projects 96

4.5 Future Work 103

4.6 Summary 103

5 | Conclusion **104**

Appendix A | Definitions & Acronyms **106**

Appendix B | Computing Education Seminar Resources **109**

Appendix C | Berea College Course Syllabi **119**

References **131**

List of Figures

Figure 1.1. College enrollment of women in computer science over time, as compared to other fields.

Figure 1.2. Schools with greater underrepresented racial/ethnic group enrollment, greater low-income student enrollment, and situated in non-suburban settings are all less likely to offer computer science. *Source: Code.org.*

Figure 1.3. Top row: computational craft workshops with quilters (left), scrapbookers (center), and a ceramics artist (right). Bottom row: Learning to screenprint at EnsAD Paris (for later use with thermochromic inks), participating in an outreach event for young women at Microsoft NERD, and learning how weaver Dena Molnar is integrating conductive materials into her practice.

Figure 1.4. Screenshots from my earliest e-sewing tutorial website (left) and the later LilyPond community website (center and right).

Figure 1.5. Top row: facilitating an activity with Exploratorium visitors, facilitating at an evening special event with soft circuit artists and experts Grace Kim and Syuzi Pakhchyan. Bottom row: example paper circuits created for museum display (left and center), interactive paper circuit example for museum visitors (right).

Figure 1.6. Teaching students to "think like a computer scientist" by issuing sandwich-making instructions to their instructors-turned-robots. (Pictured: Dr. Scott Heggen and myself, early in a CS1 term.)

Figure 2.1. Coursework components, with percentage of overall course grade and a short summary.

Figure 2.2. Felted circuits designed by students: a fairy house that lights up inside when the flower is placed atop it, an angler fish, and a baby bird.

Figure 2.3. Plotter designs produced by students as they learn to draw algorithmically. (Pen on paper, machine-drawn.)

Figure 2.4. Vinyl-cut stickers, which students valued enough to apply to their own laptops. (Student-produced stickers appear here in magenta, dark red, and sky blue, from left to right.)

Figure 2.5. Student final projects. Top row: laser cut coasters and a hand-embroidered pillow. Middle row: a hand-embroidered scene, a tooled leather bracelet, and a hand-embroidered pillow. Bottom row: a graduation cap, start to finish.

Figure 2.6. Halloween-themed paper circuits made by students.

Figure 2.7. Layered laser-cut paper, from an earlier version of Mini-Project #3.

Figure 2.8. The course website, as organized on Trello. An overview of current tasks and references (left) and a "card" offering guidance on the final phase of an individual mini-project (right).

Figure 2.9. Personally meaningful student work. Left: one student represents herself by coding and pen-plotting a face that is half African and half Native American. Right: another student celebrates her graduation by hand-embroidering a computational design and framing it in a shadowbox with keepsakes.

Figure 4.1. LilyTiny prototypes, from left to right: initial milled circuit board, custom-ordered factory board, final commercial product (sold by SparkFun Electronics).

Figure 4.2. Sample activities from the workshop curriculum.

Figure 4.3. Select pages from the plush monster activity, which utilizes the LilyTiny.

Figure 4.4. Derivative and follow-on sewable microcontroller boards. From left to right: LilyTwinkle ProtoSnap, Gemma, an unbranded clone, and the LilyPad LilyMini.

Figure 4.5. LilyTiny and LilyTwinkle monthly sales, showing sustained market interest over many years.

Figure 4.6. SparkFun sewable microcontroller sales, July 2012 through June 2020. Note that the LilyPad LilyMini was not introduced until 2016. Each color represents a different product family. Each pie slice represents a different product release (i.e. LilyPad Arduino 328 Main Board, LilyPad Arduino Simple Board, Firefly Jar kit, etc.). Kits are categorized by which board they include.

Figure 4.7. SparkFun sewable microcontroller ordering patterns, after adjusting for lab packs which contain multiple boards. Notice that a much greater percentage of LilyTiny orders include quantities of the board suitable for teaching.

Figure 4.8. Hobbyist projects using the LilyTiny (clockwise from upper left): an e-textile logo, embellished headbands, and a sock monkey with a glowing heart.

Figure 4.9. Art and craft projects using the LilyTiny (clockwise from upper left): an embroidered bracelet, a knit bracelet, and a mixed media art piece.

Figure 4.10. Evidence that some LilyTiny users are choosing to reprogram their boards (left) and are successful in doing so (right).

Figure 4.11. Evidence of teaching with the LilyTiny, including offerings at camps, libraries, and K-12 schools.

ABSTRACT

Leveraging Novel Teaching Domains Toward Broader Participation in Computing

by

Emily Marie Lovell

The field of computer science has long been plagued by issues of diversity – in particular, attracting and retaining those historically marginalized in computing contexts. This is a great loss to the field, to the future of innovation, and to society. Perhaps most importantly, it is an incalculable loss to those populations excluded from pursuing a passion for computing in the first place.

This dissertation chronicles a collection of projects aimed at broadening perceptions of computing, who is participating in computing, and what kinds of artifacts are created with computing. These projects leverage extensive fieldwork in the educational domains of computational craft and open source contribution; they entail (1) course design at the college level and (2) tool and curriculum design for a more open-ended audience of hobbyists and educators. The contribution of this dissertation is documentation of these design processes, along with my subsequent reflections, recommendations, and analysis.

First, I share my experience designing two courses developed while on faculty at Berea College: *Craft of Computing*, which aims to attract a diversity of first- and second-year students to computing, and *Open Source Software Engineering*, which seeks to retain a diversity of upperclassmen through graduation and into computing careers beyond. Second, I revisit my own prior work in e-textiles tool/curriculum design, sharing long-term impact analysis for the LilyTiny sewable microcontroller and accompanying workshop guide.

Evidence so far suggests that my forays into college course design successfully piqued students' interest in new domains, while positively influencing their confidence, identity, and sense of belonging. Analysis of the LilyTiny and accompanying workshop curriculum is also promising; it shows that an inexpensive and stable tool, coupled with freely available instructional resources, can indeed achieve widespread adoption in a market suggestive of novice and educational use.

To my shadow, Lyra Wenley.

Acknowledgements

It is often said that finding the right advisor is the most influential factor in one's graduate school experience... and I believe this through and through. I owe an enormous debt of gratitude to James Davis, my advisor on and off for over 15 years now. It was James who first trusted me with a course redesign and who lent me a copy of *Unlocking the Clubhouse* when I was feeling discouraged in college. It was also James who suggested I consider graduate school – and who then wrote letters of recommendation on almost no notice, who kept in touch with me while I was at MIT, and who offered me an academic home to finish my doctorate when I left. Along the way, James has been an unwavering ally, advocate, champion, and mentor; he has supported me in chasing my passions, including detouring to teach at a liberal arts college once it became clear that I thrive in the classroom. My meandering path has ultimately caused James a great deal of headache and paperwork, but he has vouched for me every step of the way. He has taught me how to stay calm under pressure, how to maintain perspective, and how to embrace life's unpredictability. He has supported me in balancing competing priorities, while always putting my health first. Thank you, James, for nurturing my growth immeasurably as a writer, communicator, researcher, and educator – and for always treating me as a colleague.

I owe great thanks as well to David Lee and Jan Pearce, for serving on my committee and enriching my work. Thank you for asking thoughtful questions, offering

constructive feedback, and wholeheartedly endorsing my focus on computing education. Linda Werner and Charlie McDowell also supported my computing education journey at UCSC; Charlie taught my first ever programming class, while Linda later sponsored a seminar of my own design. Pioneers in the field, they have offered expert insight, challenging and cheering me along my way.

My MIT advisor, Leah Buechley, provided a tremendous amount of support and guidance throughout my master's degree and early in my doctorate; in particular, she entrusted me with making the LilyTiny a reality and she facilitated bringing it to market. More recently, she obtained sales data central to my dissertation and helped to guide my ensuing analysis. SparkFun Electronics generously put together that data set and has maintained the LilyTiny as part of their product line over the past ten years.

My earlier work was also supported by others at MIT – especially High-Low Tech groupmate David Mellis, who taught me how to do PCB layout and who supported early programming workflow for the LilyTiny. Pol Pla graciously contributed to the design of the workshop guide and also photographed the pilot workshops. Amy Fitzgerald of MIT's Edgerton Center and Chris Randall of WGBH helped to pilot and give feedback on the LilyTiny and workshop guide. Later on, Jie Qi and Natalie Freed spearheaded the design of the bonus plush monster activity, which we then workshopped together.

To everyone at the Exploratorium's Tinkering Studio: thank you for letting me be a guest in your world! My time at the Exploratorium taught me so much about facilitation and inspired the needle felted circuit mini-project in *Craft of Computing*.

The Foss2serve and Teaching Open Source communities invited me into faculty circles before I had the credentials to be there – and showed me just how collaborative academia can be. Gina Likins, Tom Callaway, Heidi Ellis, Greg Hislop, Lori Postner, Darci Burdge, and so many others… thank you. Thanks also to everyone in the Mozilla DevTools community who supported my students, especially Jason Laster and David Walsh.

My Berea College department chairs Jan Pearce and Mario Nakazawa graced my time as a junior faculty member with immense freedom and respect. In addition, Nancy Gift and Leslie Ortquist-Ahrens supported me with mentorship and advocacy, while Deans Chad Berry and Matt Saderholm offered institutional flexibility in support of my health. Colleagues Dan Feinberg, Lisa Marks, Lex Lancaster, and Scott Heggen offered friendship, challenged me as an educator, and provided input on my evolving courses. Amy Nichols, Adriana Núñez, and Billy Korinko lent accountability, commiseration, and support as we all sought to balance dissertations with teaching. And to the teaching assistants and students across all of my courses… y'all are amazing! Sandra Perkins and Bria Williams especially helped get *Craft of Computing*

off the ground, after which Cody Mitchell and Jacob Hill took it to the next level. Alex Sharron helped calmly steer students through the waters of open source.

Many friends and loved ones have also supported me. My parents, Mark and Eileen, have unconditionally honored every twist and turn on my path – while my brother, Sean, has kept me humble and laughing along the way. I am grateful for my cohorts in both education and computer science, especially Mecaila Smith, Arnold Sanchez Ordaz, Ethan Chang, Priscilla Sung, Dhanya Sridhar, Ryan Compton, Brad Dettmer, and Afshin Mobramaein. (Thank you, Ryan, for inviting me to my first open source workshop!) More recently, Patrick Ray has been a steadfast source of support and the ultimate pandemic teammate, while Priscilla Sung, Mecaila Smith, and Anne-Marie Morey have championed me through the home stretch with walks, food deliveries, and co-working. (I would not be graduating if not for the hundreds of hours spent working in Priscilla's office and on Zoom with Mecaila.) My doctors have kept me healthy enough to keep going and have carefully interwoven medical treatment with deadlines; thank you Nevena Zubcevik, Mischa Grieder, Katherine Lantsman, Christine Green, and the California Center for Functional Medicine.

I have been so fortunate to learn from many experienced educators who have offered guidance and encouragement, patience and trust. They have empowered me to find my own place in the classroom and I hope to pay this forward, inspiring others to experience computing with the same curiosity, playfulness, and magic that I do.

A portion of this material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1339067. This work has also been funded, in part, by the MIT Media Lab Consortium and a UCSC Dissertation Year Fellowship.

1 | Introduction

The field of computer science has long experienced a dearth of women and other underrepresented minorities, one which has been documented by researchers and reflected in both enrollment and hiring statistics [79, 80]. This is a great loss to the field – and more broadly to humanity – as innovation depends on a diverse workforce [104]. In addition, careers in computer science afford a great deal of flexibility and financial stability, enabling upward social mobility for a broad cross-section of individuals [95].

For women, this dearth has manifested as a decline over time; women have, in fact, served as some of the field's most influential pioneers. Dating back to the 1830s, Ada Lovelace maintained notes while working on Charles Babbage's Analytical Engine that document some of the earliest known computer programs and her own broad considerations for the future of computing [86]. When ENIAC came into existence over 100 years later, serving as the first general-purpose electronic computer, it was a group of six women that served as its first programmers – and even they were selected from a much larger group of women employed as mechanical computers, using only calculators to do their sophisticated work [66]. Not longer after, Grace Hopper invented the first compiler and drove the pivotal development of early programming languages [9]. Initially driven by the constraints of wartime, the

following years presented great opportunity for women in the field of computing. However, this upward trajectory of representation shifted to a decline in the mid-1980s, as shown in Figure 1.1. We are still recovering from this decline today.

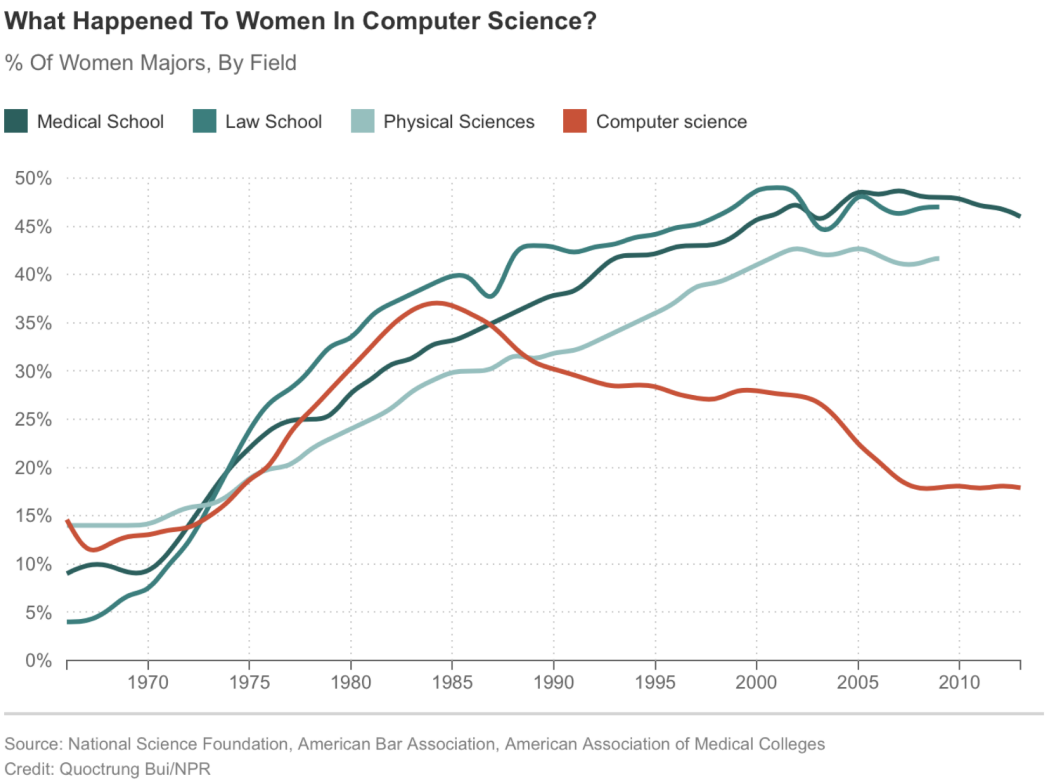

Figure 1.1. College enrollment of women in computer science over time, as compared to other fields.

Unfortunately, those from non-dominant racial and ethnic groups have been consistently marginalized throughout the history of computing [79]. This persists to present day, with students of color and low-income students being far less likely to have access to computer science courses in their high schools [12]. (As shown in Figure 1.2, non-suburban students face barriers to access as well.) These disparities

are critical, as those with access to computing in high school are much more likely to pursue it in college [92].

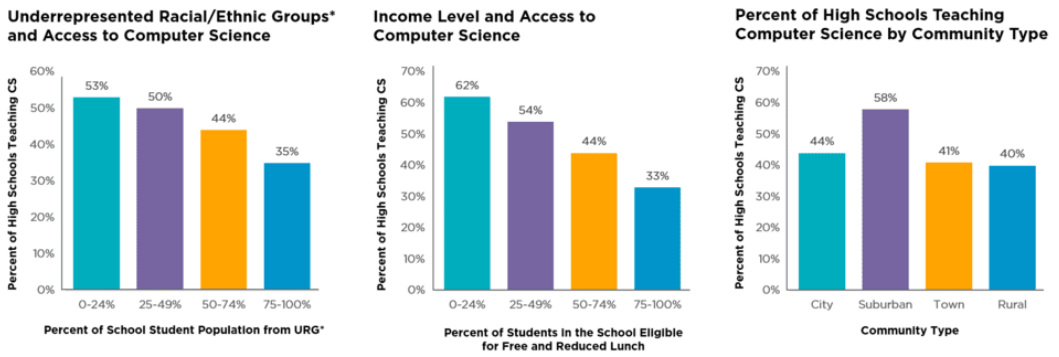

Figure 1.2. Schools with greater underrepresented racial/ethnic group enrollment, greater low-income student enrollment, and situated in non-suburban settings are all less likely to offer computer science. *Source: Code.org.*

In addition to the obvious problem of access, there are many factors thought to affect the persistence of historically minoritized students in computer science. For example, self-efficacy, or belief in one's own domain-specific capabilities, can be just as important for a student's success as the student's actual capabilities [6, 7]. Of particular note, differences in perceived self-efficacy across gender may help explain the computer science enrollment gap – and also, persistence gap – between male and female students at the undergraduate level [88]. Research has shown pair-programming to be one effective means of bolstering student self-efficacy in computing [84]. An additional avenue for cultivating self-efficacy is for a student to see it modeled by an instructor or mentor [8].

Related, a growth mindset is the belief that one can become smarter by working harder, as opposed to the belief that each person is born with a fixed amount of intelligence [27]. Computer science includes, by necessity, repeated encounters with failure — for example, though iterative software design and routine debugging. Given the constant evolution of programming languages and practices, it is also important for students to be comfortable with a shifting technical landscape [94]. Students must feel confident in their ability to learn from both hard work and mistakes, leading some researchers to promote cultivation of a growth mindset through classroom interventions [25, 124].

Additional factors that have been shown to positively influence attracting and retaining women, in particular, are: epistemological pluralism, a sense of belonging, and the potential to have positive social impact. Related research is covered in depth in later chapters [23, 80, 131].

The work described in this dissertation builds on all of the aforementioned research, seeking specifically to attract and retain a diversity of students – creating a diversity of computational artifacts – through the applied domains of computational craft and open source contribution. This is done through a combination of (1) course design at the college level and (2) tool and curriculum design for a more open-ended audience of hobbyists and educators; thus, my work encompasses research on hardware, software, and computing education. The contribution of this dissertation is

documentation of these design processes, along with my subsequent reflections, recommendations, and analysis.

1.1 Background & Motivation

I first had the opportunity to learn about course design while enrolled as an undergraduate at UCSC. As I progressed through the computer science curriculum into upper-division coursework, I noticed fewer and fewer female students in my classes — especially my introductory computer graphics class. In the year to follow, the faculty member teaching the course (James Davis, now my advisor) supported a small group of students and I in securing an instructional reform grant through UCSC's Committee on Teaching. Guided by our own diverse experiences and feedback from other students, we used this funding to draft and support a revised curriculum for the course; a new textbook, revised assignments, and partial staffing of student lab sections. When a new course, *Technology Targeted at Social Issues*, debuted in a later term, I had the opportunity to engage non-engineering students in using technology for positive social and environmental impact through working as a course assistant.

Informal Teaching Experience

I later earned my master's degree from the MIT Media Lab, where I worked as a research assistant in the High-Low Tech group [44]. Our common goal was to democratize engineering and to this end, we strove to support novice/hobbyist

communities at the intersection of craft and technology. This included documenting and disseminating our own findings — including publications, but also many tutorials on computational craft techniques, machines, and materials — as well as developing toolkits which invited participation from audiences not historically drawn to electronics or conventional programming.

Also central to High-Low Tech's mission was our engagement in informal education to support (and also, to learn from!) diverse and underrepresented groups in computing. This included numerous workshops with middle and high school students, university design students, and community artists/craftspeople, in which we would teach how to build interactive circuits using computational craft materials (such as electrically conductive fabrics, threads, and paints) and/or physical computing platforms such as Arduino [3]. Venues for these workshops included the MIT Museum, the Fuller Craft Museum, Maker Faire, SIGGRAPH, and the Computer Clubhouse International Conference. Our ultimate objective was to empower participants to feel more comfortable creating with electronics — and in some cases, to incorporate interactive circuitry into an existing art or design practice. In turn, workshop participants helped us to understand the technology barriers they faced, and also sometimes shared craft expertise in areas such as screenprinting, weaving, and ceramics. These workshops served as a valuable fieldwork component to my research, allowing for reflection throughout the iterative instructional design process. Preparation for each workshop required thoughtful consideration of personalization,

resources, activity structure, and necessary technical knowledge. Some of these workshops and collaborations are shown in Figure 1.3.

Figure 1.3. Top row: computational craft workshops with quilters (left), scrapbookers (center), and a ceramics artist (right). Bottom row: Learning to screenprint at EnsAD Paris (for later use with thermochromic inks), participating in an outreach event for young women at Microsoft NERD, and learning how weaver Dena Molnar is integrating conductive materials into her practice.

Two related projects which I contributed to during this time, but which are not detailed in this dissertation are: *CopyCAD*, enabling copy and paste of physical objects, and *The Living Wall*, a programmable wallpaper which leverages a reconfigurable magnetic Arduino-based toolkit. These projects were presented in 2010 at ACM's UIST and Multimedia conferences, respectively [18, 36].

Throughout my time at MIT, my own individual research focused on developing tools/curricula to support computational textiles at the K-12 level. In addition to organizing my own outreach workshops, mostly with young women, I also developed

resources to support informal learners working on their own; I started by developing very basic electronic sewing tutorials online (there was little freely available at the time) [39] and later collaborated with education faculty to develop an online community for the sharing of e-textile projects. Figure 1.4 shows select screenshots from these two web-based projects, which were presented at ACM's IDC and C&C conferences in 2010 and 2011 [71, 72].

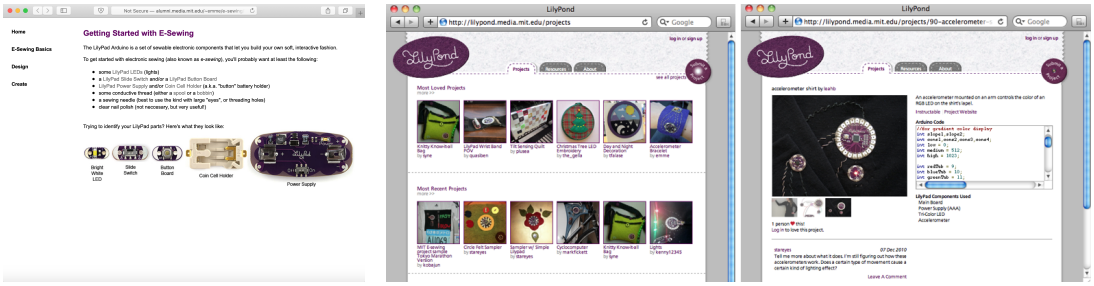

Figure 1.4. Screenshots from my earliest e-sewing tutorial website (left) and the later LilyPond community website (center and right).

It became apparent that there was a resource gap for educators wanting to introduce physical computing through sewn circuits; required physical materials were prohibitively expensive and curricula were either too introductory or too advanced. My MIT master's thesis addressed this with the pilot, development, and launch of a workshop curriculum [69, 77]. In addition, I designed and prototyped a low-cost open source sewable microcontroller known as the LilyTiny — now sold by SparkFun Electronics as part of the LilyPad Arduino product line [67]. This work is further detailed in Chapter Four, along with a recent follow-on analysis.

I have also sought to support experiential learning with computational craft through collaborating with museums and museum educators. This has included facilitating one-off activities (such as at Santa Cruz's own Museum of Art and History) and a three-month long internship with the Exploratorium's Tinkering Studio in San Francisco (offered through the Maker Education Initiative's Maker Corps program) [78, 129]. Working at the Exploratorium, in particular, offered in-depth experience designing drop-in activities, engaging with casual learners, and creating physical demos/examples robust enough for a museum floor. Figure 1.5 highlights some activities, events, and example projects from this time period.

Figure 1.5. Top row: facilitating an activity with Exploratorium visitors, facilitating at an evening special event with soft circuit artists and experts Grace Kim and Syuzi Pakhchyan. Bottom row: example paper circuits created for museum display (left and center), interactive paper circuit example for museum visitors (right).

After completing the first year of my doctorate at MIT – and following the dissolution of High-Low Tech – I returned to UCSC to continue my doctorate. While exploring potential new research directions, I was invited to volunteer at a one-day *Open Source Comes to Campu*s workshop at Hartnell College. This workshop was co-organized by CSU Monterey Bay and Hartnell's innovative CSin3 program [95], which seeks to graduate underrepresented students in only three years — and by OpenHatch, which was a non-profit organization that supported novice open source contributors through online resources and in-person workshops [103]. As a mentor at the workshop, I supported a small group of students in discussion, hands-on exercises, and making a contribution to an existing open source project.

It was both exciting and humbling having to learn on-the-fly, staying only step step ahead of students throughout the day. One of the students and I stayed in touch beyond the workshop and continued meeting on IRC to see her contribution successfully merged into a project. She was elated! Seeing a diversity of students' enthusiasm for learning real world tools (i.e. version control, bug tracking) and making a concrete contribution to a real world project sparked my own interest in using open source in the classroom as a means to attract and retain a broader cross-section of students. Chapter Three details my journey further into this teaching domain. (In parallel to all of this, I also sought to gain more personal experience contributing to an open source project. This led me to collaborate with two other UCSC computer science graduate students on interface/usability improvements to

Sahana Eden, a disaster response platform [121, 139]. We published this work at IEEE's 2015 GHTC [1].)

Formal Teaching Experience

In addition to a wealth of informal teaching experience, being a graduate student has offered many opportunities for me to grow as a classroom educator. During my time in the High-Low Tech research group, I served as a teaching assistant for a project-based graduate course entitled *New Textiles* [98], which explored the future of textiles through the combined lenses of craft and technology. After later returning to UCSC, I worked as a teaching assistant for our undergraduate *Introduction to Computer Science* course (CMPS 10; UCSC's version of a CS0 course).

I wanted to progress from working as a teaching assistant to teaching courses of my own, so my advisor supported me in taking on a graduate student instructor (GSI) position to offer my own sections of *Introduction to Computer Science* over two consecutive summers at UCSC. Although I fortunately inherited a well developed and tested curriculum — largely based on the *AP Computer Science Principles* curriculum — I put a great deal of effort into adding more active learning components, something afforded by my smaller summer class sizes. These included, for example, small group activities in which students learned about search algorithms by playing guessing games as well as discussions on provocative videos and readings.

During this time, I was also organizing an informal lunchtime gathering for UCSC graduate students interested in computer science education; we would meet monthly and talk about papers we were reading, our own ideas for research projects, and even made plans to attend conferences (such as SIGCSE) together. Students in this group expressed repeated interest in diving deeper into reading and discussion — and so my advisor, along with another faculty member, Linda Werner, supported me in developing a graduate level seminar on the topic. I put together a themed reading list which included a few must-read papers for each week, alongside a longer list of supporting literature. Examples of topics/themes that we covered include: research methods, theoretical background, active learning/flipped classrooms, broadening participation, and programming languages for teaching. I organized discussion leaders for each topic, and came up with my own weekly discussion questions to keep conversation flowing when we got stuck. I put all of these assignments/resources into a shared document online which we collectively revised and added resources to as they came up during our discussions. (A complete version of our document, as it was at the end of our time together, is included as an appendix.) Linda contributed her own extensive computing education research experience by serving as a faculty sponsor for the course, attending our weekly meetings, and helping to facilitate discussions. The course was also attended by Charlie McDowell, another faculty member whose research has been influential to the field of computing education, the

graduate students from our earlier lunchtime group, and a few more who learned of the course through flyers/mailing lists.

In my third year of doctoral studies at UCSC, around the time of my advancement, I was recruited for a faculty position at Berea College. Berea is a small liberal arts college – and also a work college – located in Berea, Kentucky. The college's mission is to serve students of great academic promise, but limited economic means; all students attend on full scholarship and participate in a campus labor program, through which they work part-time for the college while pursuing their degrees. The Berea student body is exceptionally diverse; 40% are students of color, 55% are first-generation college students, and 11% are international students (representing 76 different countries, mostly in the developing world). It is a unique and impactful setting in which to consider broadening participation in computing, given that students from historically underserved communities comprise most of the student body.

While fortunate to teach some of my own classes as a graduate student at UCSC, I seized this opportunity to join a small computer science department as a faculty member, viewing it as a chance to do truly immersive fieldwork. Berea, in particular, offered me the freedom to develop and teach my own courses at the boundaries of computer science and to work closely with underrepresented students. In addition, the college values innovative teaching pedagogy, including active learning and a flipped

classroom approach. (Figure 1.6 shows my early participation in one such instructional activity, in which students must verbally "program" their instructors-turned-robots to make a peanut butter and jelly sandwich.)

Figure 1.6. Teaching students to "think like a computer scientist" by issuing sandwich-making instructions to their instructors-turned-robots. (Pictured: Dr. Scott Heggen and myself, early in a CS1 term.)

I taught at Berea for a total of two and a half years, during which my research and teaching interests were nurtured and my position converted from temporary to tenure-track. I designed two new computer science courses at the lowest and highest level offered by Berea (100-level and 400-level), both of which are detailed in this dissertation. I also had the opportunity to take over Berea's CS1-equivalent, *Software Design & Implementation*, teaching three sections in parallel in my final term. I left

my newfound home (and career) in rural Appalachia due only to ongoing challenges with my health and the difficulty of completing a dissertation while teaching full time. I am so terribly grateful for having had this experience, which, in turn, became central to my dissertation.

Collectively, all of these teaching experiences — informal and formal — allowed me to better understand the perspective of various learners, from K-12 students to university students (sometimes, peers) to craftspeople to museum visitors. I learned how to translate ideas and concepts into curricula, how to appeal to a diversity of learning styles, and how to design learning experiences for different settings; a 90-minute course meeting and 10-minute drop-in museum activity contrast greatly in their challenges and affordances! These experiences also gave me the opportunity to directly observe barriers to learning about computer science and electronics, such as low technological self-efficacy and a fixed mindset, which I wrote about as part of the 2014 ICER Doctoral Consortium [70]. These early teaching experiences also gave me the chance to observe resource and opportunity gaps, some of which I sought to address in my later work.

1.2 Document Overview

This remainder of this dissertation chronicles a collection of projects aimed at broadening perceptions of computing, who is participating in computing, and what kinds of artifacts are created with computing. All of these projects are situated within the landscape of applied domains which are not yet commonplace in teaching, namely computational craft and open source contribution.

Chapters Two and Three summarize two major curriculum-design projects undertaken while serving on faculty at Berea College over a two and a half year period of time. The majority of this work was published at IEEE's 2021 Frontiers in Education Conference [74, 75]. Syllabi for my final offerings of these courses are also included as an appendix.

More specifically, Chapter Two reports on a CS0-level computational craft course added to Berea College's departmental offerings in hopes of further broadening participation. I summarize the course design and structure, which emphasize algorithmic design (using Processing), handcraft, and digital fabrication. I share examples of creative computational work and feedback from students, as well as reflections on the course's efficacy within Berea's funnel-style curriculum. Early evidence suggests that the course offers a highly personal and creative entry point to

computing – and one that is effective at engaging a diversity of students while ensuring a smooth transition to CS1.

Meanwhile, Chapter Three reports on my experience scaffolding student success in the uncertain landscape of open source. Following participation in faculty workshops on the subject, I spent two consecutive terms developing, teaching, and revising an upper-division open source software course. The difference between the two course offerings was astounding; students enrolled in the second iteration made more successful project contributions, spent more of their own time working outside of class, and felt a greater connection to both the project and the developer community of which they were a part. I detail my experiences, with particular focus on the importance of project selection – as well as the revisions I believe to be most responsible for improvement: additional mentorship, supplemental in-class tutorials, more dedicated class time for teamwork, intentional team groupings, and access to large screens for collaboration.

Chapter Four presents follow-on analysis of work from my master's thesis. The LilyTiny sewable microcontroller was created ten years ago – as part of that thesis and in collaboration with my advisor at the time, Leah Buechley – in an effort to make electronic textiles more accessible. At the time, e-textiles was gaining traction as a means to invite more diverse participation in computing, but financial and instructional barriers stood in the way of broader adoption. In addition, there existed a

scaffolding gap between projects involving lights, batteries, and thread – and those requiring programming (i.e. leveraging the LilyPad Arduino and/or additional sensors or outputs). In an effort to expand access to electronic textiles, I designed the LilyTiny, an inexpensive, pre-programmed sewable microcontroller which controls assorted LED patterns, and which later became available for purchase through SparkFun. Alongside the LilyTiny, I released a free workshop guide for educators which details five low-cost activities that can be taught without any prior electronics experience. This chapter summarizes my prior development of the LilyTiny and companion curriculum – and then reflects on whether I met my stated goal of expanding access to electronic textiles in the decade since. I share and discuss various measures of impact, including: a survey of derivative products, a multi-year analysis of sales data from the LilyTiny's sole distributor SparkFun Electronics, and a sampling of customer reviews and projects. The majority of this work has been accepted for publication at ACM's 2022 CHI Conference on Human Factors in Computing Systems [73].

2 | Course Design for Attracting Broader Participation: *Craft of Computing*

My job interview at Berea College concluded with the department chair asking me over dinner, "If you had complete freedom to develop a new course, what would it be?" Intrigued by the idea, I suggested something of a mashup between introductory programming in Processing, computational craft, and algorithmic design. I had enjoyed teaching students to program in Processing at UCSC, had a lot of experience with computational craft (and it's creative, diverse possibilities) from my time at MIT, and had always wanted to experiment more with algorithmic design. All of these approaches also held promise as avenues for broadening participation. When offered the position, I was invited to design exactly this course – the same one I started imagining over Indian food that night.

My only constraint was to situate the course, which we named *Craft of Computing*, within an existing selection of CS0-level courses designed to invite diverse participation. These courses target both (1) non-majors who are curious about computing (and/or are seeking to fulfill a college-wide general education requirement) and (2) computer science majors who may not have much prior experience with computing. Accordingly, most seats in Berea's CS0 courses are reserved for freshman and sophomores. Unless exempted by instructor permission,

students are required to take at least one CS0-level course before proceeding to Berea's CS1 equivalent.

Because of this structure – in addition to Berea's broader institutional context – this course offered a unique opportunity to impact student perceptions of computing; it was a chance to share with students how computing can be personal, creative, and applied, and to do so at a pivotal moment in their academic journey and identity development. My department unequivocally supported these efforts in terms of space, equipment, materials, mentorship, and teaching support – without which this work would not have been possible.

2.1 Introduction

CS0 courses can offer students with little-to-no computing background the opportunity to explore computer science before committing to a major [136]. What's more, CS0 courses can help to level the playing field for these students by the time they enter CS1 alongside peers who may have taken computer science courses in high school and/or have a stronger mathematics background [13].

Berea College, among other institutions, has adopted a CS0 "funnel" approach to attract minoritized students; that is to say, Berea offers several CS0 courses on topics of interest to the general student population [108]. This approach has since been adopted and found to be effective elsewhere [42, 136]. Upon my hire, I was invited to

design and teach a new CS0 course in my domain of expertise – computational craft – in hopes of further broadening departmental demographics. Computational craft has been studied as a successful avenue for attracting and retaining groups historically excluded from computing, particularly women [14, 58]. This course, entitled *Craft of Computing*, covers core CS0 concepts including computational thinking, variables, loops, functions, etc. The course also showcases the creative possibilities of computing when paired with handcraft, digital fabrication, and algorithmic design. I taught *Craft of Computing* five times over a two-year period, during which it was deemed so successful as to be added to the college's permanent catalog.

In this chapter, I detail the course structure, learning goals, and major assignments – complete with many compelling examples of student work; *Craft of Computing* students have created personally relevant and meaningful artifacts, often displayed in their dorm rooms or given as gifts. Examples include: a needle-felted fairy house with embedded LED lights, light-up paper circuit valentines, and beautiful recursive geometric patterns – first generated in Processing and then realized in the form of plotter drawings, vinyl-cut laptop stickers, and laser-cut wooden coasters.

Early analysis suggests that the course offers a highly personal and creative entry point to computing – and one that is effective at engaging a diversity of students while ensuring a smooth transition to CS1. My personal observations are supplemented with student feedback in the form of interviews, informal course

reflections, and end-of-term course evaluations. I also provide insights and recommendations for others looking to adopt a craft-themed CS0 course.

In sum, the primary goal of this chapter is to document *Craft of Computing*, a novel undergraduate course designed to broaden *perceptions* about computing – which in turn, influences who *participates* in computing [22, 81].

2.2 Related Work

My work builds upon a body of prior research on computing education pedagogy, broadening participation in computing, CS0 course design and outcomes, the Processing programming language, and computational craft.

The course design draws upon a number of existing pedagogical approaches that are well-researched both within and beyond computing education. For example, I sought to support affective learning in this course – doing so through encouraging informal social interaction, challenging students to bridge the digital and physical worlds, and portraying computational craft in the light of "hard fun" [106, 110]. (In other words, computational craft *is* challenging, but if actively engaged in one's personal and creative pursuit, one is less likely to mind.) The class is also built around an active learning approach, which is known to enhance student learning and motivation, heavily interspersing hands-on activities with short periods of instruction or tutorial [38, 82, 111]. Overlaying the design of the course itself, my own teaching relies

heavily on guided discovery; when a student asks a question, I respond with a series of questions to lead them to an answer – rather than merely supplying the answer up front [2]. These techniques and approaches were foundational to the design of the course, many of them being used widely at liberal arts colleges due to their positive effect on student learning and overall experience.

Turning to broadening participation, the dearth of women and other minoritized groups in computing is well documented, as are some of the factors critical to attracting and retaining them [65, 79, 80]. Especially relevant to this chapter are a sense of belonging and engaging in work that is personally meaningful and/or culturally relevant [49, 64, 89, 132]. Building technological self-efficacy and a growth mindset can further support these goals [25, 77, 94, 124]. Creative computing, in particular, has been shown to effectively increase growth mindset and decrease computer anxiety [68]. The design of *Craft of Computing* builds upon this knowledge, including the incorporation of specific recommendations – for example, encouraging students to pursue identity-affirming creative work and curating a physical space in which students of diverse identities feel welcome [23]. The course also leverages pair programming throughout, which is well documented as supporting broader participation [83, 84, 134].

There is also ample research on the importance of CS0-level courses as an avenue into computing, especially for students with little-to-no prior programming

experience. Prior work also affirms the importance of CS0 as a way to broaden student perceptions of computer science [136]. To this end, a "funnel" curricular model has been well-documented, in which several different themed CS0 courses are offered as a means to invite diverse participation [42, 108, 136]. My work expands existing practice by offering yet another novel entry point to computing, through a domain which is both creative in nature and stereotypically "softer": craft.

The Processing programming language was designed to support exactly this kind of creative work [40, 119, 120]; it was developed by designers and meant to be more accessible than it's closest counterpart, Java. Processing has since been leveraged in university-level offerings of "CS Principles" and CS0-style courses, both as an approachable text-based language and a means to create art [4, 137]. It is for exactly these reasons that I chose to teach *Craft of Computing* using Processing.

Computational craft – especially the field of electronic textiles – has been established as an effective means to broaden participation in computing, especially at the K-12 level and in after-school settings [16, 56, 58]. My work expands these initiatives to the undergraduate context, in which students are critically deciding upon and pursuing a field of study to propel their careers. My work also infuses a stronger software component (via programming in Processing), in hopes that this may prepare students for CS1.

Lastly, the design of this course was especially informed by the work of CU Boulder's Craft Technology Lab and my prior research group at the MIT Media Lab, High-Low Tech [44]. Both of these now-defunct groups have laid a strong foundation of tutorials and tools which enable making, hacking, and programming rooted in craft. Their missions focused on democratization of engineering and the radical inclusion of diverse populations – and their research and teaching has very directly inspired my efforts to formalize a course at the undergraduate level.

2.3 Background

My course joined an existing selection of themed CS0 offerings at the college, all of which funnel into a singular CS1 course. Examples include: *Intro to Robotics, Storytelling with Alice, Intro to Game Design*, and *Building Better Apps*. While *Craft of Computing* debuted as a "special topics"/elective offering of this variety, it was added to the college's permanent catalog within one year.

I taught *Craft of Computing* five times over a period of four consecutive 15-week academic terms. This chapter includes examples of student work across all of the terms in which the course was offered. However, in detailing the structure of this course, this chapter will focus on the latest iteration unless otherwise noted. There were some notable changes made over the two year period in which the course was

developed; those are discussed toward the end of this chapter in the context of recommendations for others.

2.4 Course Design

Craft of Computing shares the same learning goals of many other CS0-level courses, namely to teach core computer science competencies while showcasing applications of computing (in this case, creative ones), and to lower the barriers to entry for students with little or no programming experience [136]. I leverage craft as a context because it is both relatable and provocative when considered in juxtaposition to computing – while also exposing students to creative applications of programming. In terms of domain-specific content, *Craft of Computing* exposes students to the following:

- *Programming in Processing* – including coverage of computational concepts such as loops, variables, and functions (facilitating the creation of computational art)
- *Some basic electronics* – including simple textile and paper circuits
- *Handcraft* – including needle felting and embroidery (allowing students to realize their algorithmic designs in a traditional craft medium)

- *Digital fabrication* – including use of a plotter, laser cutter, and vinyl cutter (allowing students to realize their algorithmic designs in a computer-mediated craft medium)

The course also includes some coverage of the *maker movement*, in particular, discussion around accessibility and inclusivity of maker culture.

Setting & Organization

I taught *Craft of Computing* in a lab space shared by Berea's electronics course, which allows access to all of the relevant tools, materials, and equipment, as well as a sink for cleanup and a safe place for students to leave projects-in-progress. This classroom is also the setting for the department's evening lab hours which are open to students in all computer science courses, and where students may drop in for help or to work on projects. An assortment of TAs staff this space Sunday through Thursday night each week, and an effort is made to have one or more TAs from each class scheduled on any given night. Students feel great ownership over this space.

The course meets for long class periods – 110 minutes – twice per week. One day per week focuses on programming or computational concepts, such as Processing syntax, coordinate systems, and programming fundamentals. These class periods include informal whiteboard "mini-lectures" covering bite-sized concepts such as variables, loops, and functions, one at a time. These are interspersed with exercises from the textbook which are completed in pairs, sharing one laptop, as dictated by pair

programming practices. Students have reported really liking this format; as one student commented in a course evaluation, *"This is a class that doesn't work well with a lecture style and she knows that and taught the class accordingly, small part lecture and then hands-on work."*

The second day each week focuses on circuits, handcraft, and digital fabrication as mediums for computational art and design. During these class periods, a document camera and equipment/materials are used to do live demos on handcraft techniques and also technical topics such as how to: use a multimeter, design a simple circuit (with a battery and a LED), prepare files for digital fabrication, and use CNC equipment. These demos are interspersed with long periods of unstructured hands-on work time, during which students are encouraged to move around, work in clusters, socialize, and ask myself or one another for help as needed.

Categories of student work and assessment are outlined in Figure 2.1. Assignments and quizzes focus on building core craft and programming competencies, while mini-projects and the final project offer the opportunity to integrate and apply these skill sets. (Mini-projects and the final project are detailed in the following section.) Homework includes readings from the textbook, work on mini and final projects, and sometimes finishing an in-class craft or programming exercise (although the bulk of this work happens *in* class).

Assignments	20%	Includes written reading responses, programming exercises, and outside-of-class research. Intended to broaden both craft and technical skill sets.
Quizzes (x5)	25%	Cover technical content and allow the instructor to better understand which topics would benefit from more coverage. Quizzes are reviewed in class, with focus on revisiting topics students may have struggled with.
Mini-projects (x3)	30%	Through these special assignments, students learn to interface between the digital and physical worlds — for example, by making something on a vinyl cutter which was first designed on a computer using Processing.
Final project	25%	Invites students to blend craft and computation in a more open-ended context than the mini-projects. Once again, students bring a design of their own into the physical world — but they do so through a medium and technique of their choice.

Figure 2.1. Coursework components, with percentage of overall course grade and a short summary.

Major Assignments

The major assignments of the course, mini-projects and the final project, emphasize blending handcraft with electronics, the creation of original vector designs in Processing, and realizing those designs in physical form. Although the majority of these assignments are structured around a piece of digital fabrication equipment, the emphasis is on what creative possibilities the equipment enables, rather than simply learning how to use it.

Mini-project #1: Felted Circuits

The first mini-project invites students to blend needle felting with textile circuitry, as inspired by the work of artist Moxie Lieberman [93]. (Lieberman was an artist-in-residence in the Exploratorium's Tinkering Studio during my internship there.) Over a couple of class periods, students are taught how to needle-felt, how to design a simple circuit (with a light, a battery, and an LED), and considerations for working with electronic textile materials like conductive sewing thread. Each student sketches a three-dimensional felted object, along with plans for how they will embed a sewn circuit into its structure – with care and attention given to LED placement,

battery pack placement, and keeping the various threads from accidentally making contact and short-circuiting. Students who are interested in optionally adding a switch are encouraged to do so, and instructionally supported in modifying their sketches to accommodate this. Once students bring their sketches to me for revisions and approval, they collect the necessary materials and bring their circuits to life. Examples of student work appear in Figure 2.2.

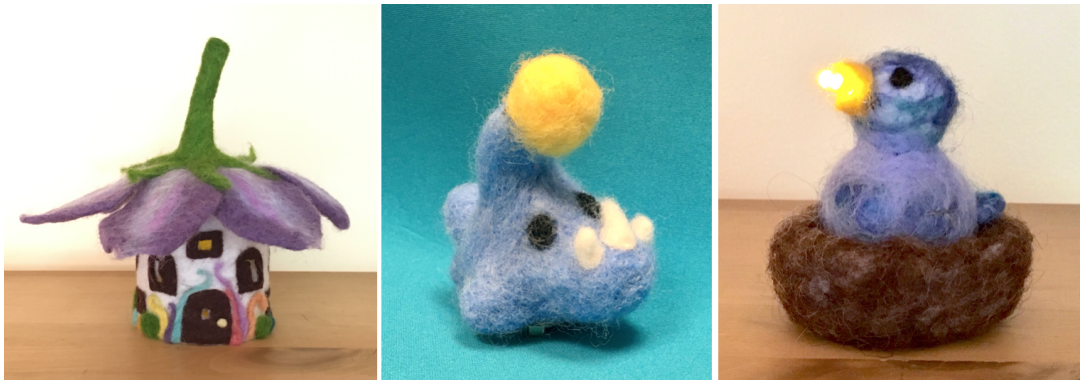

Figure 2.2. Felted circuits designed by students: a fairy house that lights up inside when the flower is placed atop it, an angler fish, and a baby bird.

Mini-project #2: Plotter Drawings

The second mini-project invites students to blend code-driven/algorithmic design (done in Processing) with vector path *drawing*. Students are asked to create an original single-frame/non-animated vector design in Processing, which must also meet the following technical requirements:

- Use of 2+ drawing commands/shapes (`line`, `rect`, `ellipse`, etc.)
- Use of *variables* whenever possible.
- At least one *loop*, to create visual repetition.

- Appropriate use of comments throughout.

Students first submit a draft of their code to Moodle (Berea's LMS) and present their draft to the class for feedback using an overhead projector. While students are working outside-of-class to incorporate any suggestions or revisions, class time is used to demonstrate how to export and format Processing PDFs for vector plotting and how to use a Cricut machine for drawing. Students submit a final draft of their code/design, output their design on the Cricut using pens or markers, and compose a written reflection about the experience. They present their final physical drawing in class alongside their code, again utilizing an overhead projector to do so. Examples of student work appear in Figure 2.3.

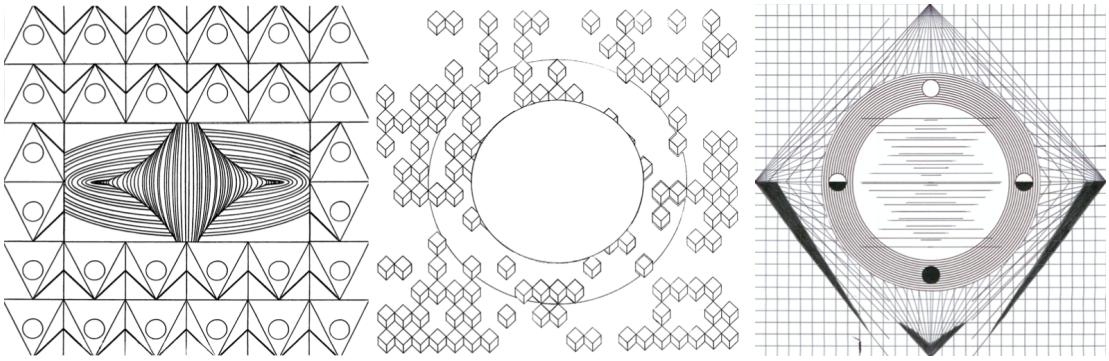

Figure 2.3. Plotter designs produced by students as they learn to draw algorithmically. (Pen on paper, machine-drawn.)

Mini-project #3: Vinyl-cut Stickers

The third mini-project invites students to blend code-driven/algorithmic design (done in Processing) with vector path *cutting*. This assignment follows the same draft-revision-fabrication structure as the plotter mini-project, but includes the added

challenge of designing for cut paths instead of drawn lines; designing vinyl-cut stickers requires students to think about positive/negative space and open/closed shapes. Each term, an early observation and point of discussion is that overlapping lines in a design will generate vinyl confetti instead of a single, unified sticker.

Students must submit an original design that is significantly different in composition from their plotter design. While students are working outside-of-class to revise their sticker designs, class time is used to demonstrate how to work with a Roland vinyl cutter and how to carefully transfer a cut sticker to a surface. For their final presentations, students are required to show their sticker adhered to a surface; even a piece of paper suffices, but most students choose to affix their sticker to a bicycle, laptop, or other personally meaningful object. Examples of student work appear in Figure 2.4.

Figure 2.4. Vinyl-cut stickers, which students valued enough to apply to their own laptops. (Student-produced stickers appear here in magenta, dark red, and sky blue, from left to right.)

Final Project

The final project invites students to blend code-driven/algorithmic design (done in Processing) with handcraft, vector path drawing, vector path cutting, or a mix of these. Essentially, students are asked to generate a more complicated vector design than they have done prior and to realize it in any of the physical mediums covered in the course. They may also use any art/craft medium with which they have prior experience or wish to explore on their own.

This project follows the same draft-revision-fabrication structure as the two prior mini-projects, but with the added technical requirement that students must use functions in their code. Students are also expected to engage in independent research and planning in terms of creating a design that is suitable for their chosen medium(s), envisioning how they will realize this design in physical form, and requesting any necessary physical materials for their project. The fabrication part of this assignment must either incorporate handcraft in some way or demonstrate a more complicated application of a tool from an earlier assignment (for example, using a plotter with two pens/colors or using a laser cutter with a new material). Examples of student work are shown in Figure 2.5 – but students also used many other mediums such as charcoal, acrylic paint, and 3D printing.

Figure 2.5. Student final projects. Top row: laser cut coasters and a hand-embroidered pillow. Middle row: a hand-embroidered scene, a tooled leather bracelet, and a hand-embroidered pillow. Bottom row: a graduation cap, start to finish.

Bonus Activity: Paper Circuits

Each term, one class period is spent teaching students how to make light-up Halloween cards (Fall term) or valentines (Spring term), as inspired by the work of Jie Qi [114, 115]. This activity is ungraded, as students take their cards with them at

the end of class, but it is intended to reinforce earlier learning about circuits and to demonstrate yet another creative technical application.

During this class period, which always takes place after the felted circuit mini-project, I teach how to create circuits on paper using copper tape, lights, and batteries. I also teach students how to solder – and they have the opportunity to practice what they've already learned about using a multimeter to measure continuity. Some examples appear in Figure 2.6.

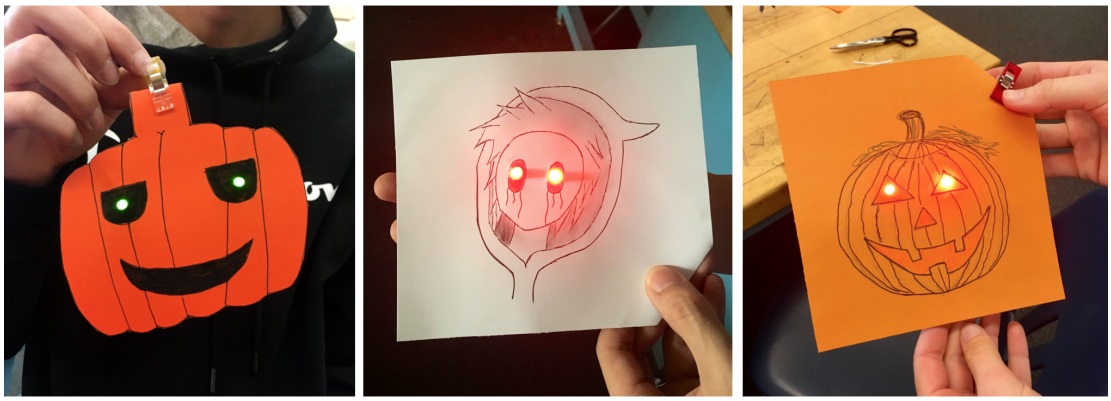

Figure 2.6. Halloween-themed paper circuits made by students.

Deprecated Assignments

In addition to the above, earlier offerings of the course included a textile sensor mini-project that followed the felted circuit mini-project. Inspired by Hannah Perner-Wilson's work [50, 109], this opened up discussion of resistance and the differences between a "dimmer" and a "switch". We used neoprene, Velostat, and conductive thread to follow Perner-Wilson's Instructable on the topic [52]. Students added their own flair to the assignment by making their sensors in creative shapes like

dinosaurs and hearts – and enjoyed comparing how this and other variables affected their resistance, using a multimeter to investigate. Although students appreciated this more advanced electronics project, it was removed from later course offerings so that we could go more in-depth on frequently-requested, more advanced topics in computing, like translation and user interaction.

A couple of earlier terms of the course also used a laser cutter in lieu of a vinyl cutter for Mini-Project #3; this was simply dependent on whether we had the necessary equipment access, and the project requirements and challenges were comparable across the two different machines. During these offerings, students were also able to use the laser cutter for their final projects. Examples of student work from these deprecated assignments appear in Figure 2.7.

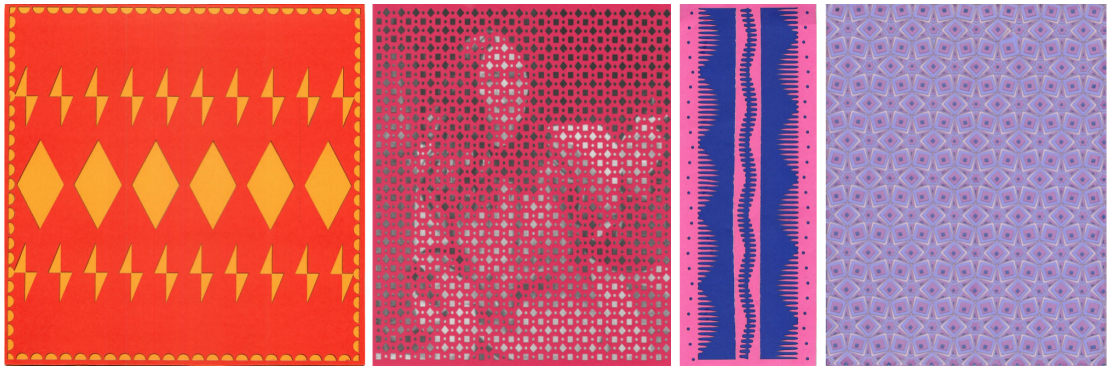

Figure 2.7. Layered laser-cut paper, from an earlier version of Mini-Project #3.

Infrastructure

The choice of using multiple infrastructure tools for teaching versus consolidating into a single platform has an effect on the tone of the class. Although there is a

learning curve to students using multiple tools in tandem, this is common in the computer science workplace and I emphasize their utility with regard to preparation for post-graduation employment.

The course is taught entirely in Processing, aside from a Blockly-based warm up assignment. I chose Processing because it was originally developed by artists and designers to enable the creation of creative work by those with minimal programming background [120]. To scaffold our journey through learning to write Processing code, all students are required to purchase *Learning Processing* and it serves as the course textbook [123]. *Craft of Computing* covers the first three chapters of the book in great detail, with select topics from later in the book covered by request.

A Trello board serves as our course website [24, 130], which colorfully organizes upcoming assignments/due dates, requirements for each assignment, and references such as the syllabus and equipment documentation. Students submit coursework on Moodle, except for physical artifacts which are submitted in person. An example from one of the course offerings is shown in Figure 2.8.

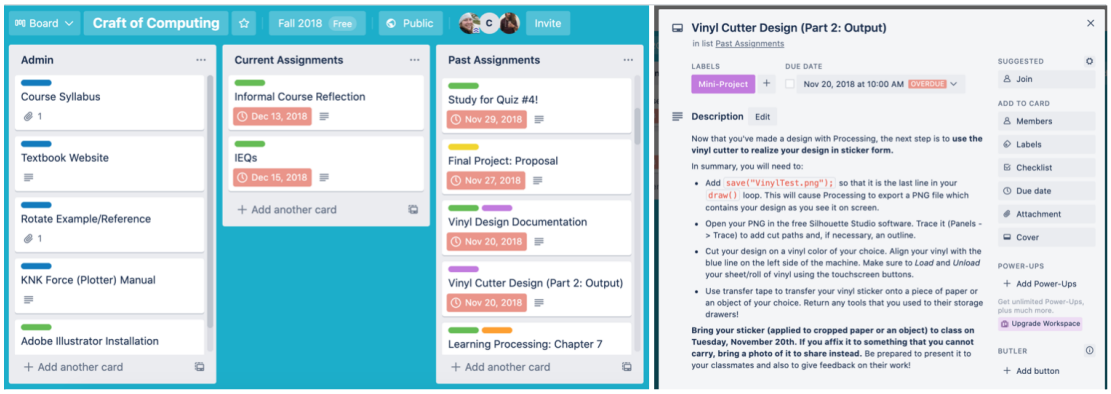

Figure 2.8. The course website, as organized on Trello. An overview of current tasks and references (left) and a "card" offering guidance on the final phase of an individual mini-project (right).

Students are also required to join the *Craft of Computing* channel on our department's Slack team [125], which was created and is managed by the department's teaching assistants (TAs). This is where any announcements or clarifications are made in between course meetings, for example, deadline extensions or on-demand examples to clarify content. Students are also asked to post on Slack rather than emailing me, as this enables a quicker response from the myself, course TAs, or other students in the course. Students are also encouraged to send *direct* messages to me and/or the TAs on Slack if they would like to inquire about grading, ask a question specific to their code, or anything more private.

Complementary to the above, Google docs is used to host software and equipment documentation and Calendly is used for students to schedule time on specific equipment [21].

Instructional Support

Teaching assistants have been absolutely instrumental in the success of *Craft of Computing*. Abundant teaching support is a unique benefit of teaching at a work college, but undergraduate tutors and graders could be tasked with similar support at other institutions. For this course, TAs staff evening lab hours and monitor the course Slack channel. They have also worked independently to create invaluable documentation that persists across terms; TAs have authored detailed step-by-step tutorials on each piece of equipment as well as file conversion processes. All of these feature annotated screenshots, lots of encouragement, and a sense of humor. These walkthroughs have been vital in terms of reducing student confusion.

TAs also learn how to use each piece of equipment in advance of associated assignments and they supervise equipment time slots, which students can reserve both during and outside of evening lab. TAs also handle all of the signups and scheduling for this.

Finally, TAs grade all of the quizzes and meet with me to grade final projects as a group at the end of each term. This is very helpful, as they have a window into each student's process and any barriers that they have encountered. On the whole, I have found it tremendously helpful to have my TAs' ongoing feedback on what topics students struggle with in lab and any issues that arise with equipment or materials.

2.5 Reflections

In this section, I summarize some of my own observations and reflections about the course.

Challenges & Rewards of Working in the Physical World

Working with physical materials and equipment can be time-consuming and frustrating, as echoed in students' course evaluations, especially as it contrasts with the software/digital focus of most computer science coursework. However, once students get in the habit of planning ahead and signing up for equipment time slots, they often express a sense of pride and triumph in what they make; a sentiment echoed through many student evaluations is, *"The more effort I put into a project, the more enjoyment I got out of it."* Students have especially enjoyed making things to display in their dorm rooms, to personalize their belongings, or to gift to loved ones – sometimes requesting items be returned early from grading in time for a birthday or holiday.

I have additionally observed students bonding over the above-mentioned frustrations; for example, how an earlier-used plotter would sometimes quit halfway through drawing or how awful the laser cutter smells after cutting wool felt. Students have also built a strong sense of community around our presentation days. Upon their suggestion, I stocked the lab with tea and second hand coffee mugs – and these class meetings came to be known as "CriTEAque Days". Sharing creative work is

vulnerable and I believe that together we have cultivated a safe space for students to provide constructive feedback and try new things. This is echoed by a student evaluation: *"[The instructor] is very passionate about the topic and encourages us to try new techniques, or to apply these techniques differently to see the outcome."* Students look forward to our CriTEAque class periods, taking great interest in one another's creative and technical growth.

Students have also reported appreciation for learning craft techniques – as a creative outlet, as a destressor especially around midterms and finals, and as a means to be more self-sufficient. One student writes, *"I hadn't concerned myself prior with sewing or felting, but I have a fair deal of interest and respect for the creative applications of both after having taken the course."* Another reflects, *"This course taught me the basics of programming and basic sewing and embroidering techniques that not only I can use in a career but for life skills too."*

Importance of Growth Mindset at the CS0 Level

The importance of a growth mindset is well-documented, and the focus on creativity and craft in this course seems to support this objective. Most introductory computing courses have all students complete exactly the same assignments. This can offer the temptation to ask another student how they solved a bug, rather than struggling with it on one's own. The focus on creativity in this course means that every project is unique, and neither other students nor myself are likely to know the answers

immediately. This allows abundant opportunity to practice debugging and growth mindset towards a goal the student is personally invested in. Student evaluation comments reflect this experience:

> *"She was very relate-able and talked through struggles. She also did not just give us the answers to problems in our code. She just spotted the spots where they were and it was like hide and seek or find Waldo. It was great to hear the encouragement of knowing that she had found the bug in the code and then we had to work on finding them ourselves."*
>
> *"She welcomes questions and mistakes on work and assignments and teaches us that sometimes mistakes aren't terrible things but can add on to our projects."*
>
> *"The instructor does a good job of leading you into fixing code you have a problem with instead of outright telling you what is wrong or what you need to add so that you can actually learn something."*
>
> *"Whenever students are stuck she just doesn't give them the answers she asks them questions for them to start "thinking like a computer scientist" as she would say."*
>
> *"She has helped me to learn that coding is no harder than solving a puzzle."*

Leveling the Playing Field

CS0 courses strive to provide an entry point for students with little-to-no programming background, yet often enroll students with a wide range of preparation. For example, some students may opt to take CS0 as review or because they are interested in the specific topic/theme that is featured. This presents a unique challenge for CS0 instructors, as they strive to balance approachability for less experienced students with keeping more experienced students engaged.

Given my experience with this course, computational craft is, in fact, very well suited to this challenge. Between the creative aspect of every assignment, the variety of tools/materials/techniques at hand, and the extensibility of Processing as a programming language – I have seen students of all levels remain engaged over the course of each term. Student evaluation comments support this:

> *"[The course] allows students to brainstorm and create their own unique projects while ensuring the students learn the content and the projects follow the specifications."*
>
> *"... individuals in the class were at a variety of skill levels and [...] everyone was able to learn despite that challenge."*
>
> *"Anytime that we finished a certain part, she would challenge us with extra tasks that gave a better understanding."*

Preparation for CS1

CS0 courses aim not only to offer an appropriate entry point to computer science, but also to prepare students with minimal programming background for CS1. Student evaluations reflect success in these areas as well:

> *"Even though I had no background of Computer Science, [...] this course [was] very accessible to me."*
>
> *"I recognize that this would be an ideal first computer science course."* (This comment was made by a student with some programming background already.)
>
> *"I would recommend anyone who is thinking about doing computer science to take this course. It's a perfect preparatory course for [CS1] and [CS2] by softly introducing key concepts that are crucial to the major. Anyone who takes this course will have a huge leg up in [CS1]."*

Despite the non-traditional computing topic area which may be perceived by some as "softer" or less difficult, *Craft of Computing* is one of the only CS0 courses in the department to use a text-based programming language; most CS0 courses in Berea's funnel utilize block languages. A couple of students posited in interviews that this makes for a smoother transition to our CS1 course taught in Python; *Craft of Computing* students already have familiarity with compiler errors, nuances of written

syntax, and data representation (e.g. ints vs. floats). *Craft of Computing*'s coverage of Processing also uniquely exposes students to debugging, libraries, and programming in different coordinate systems.

Broadening Participation & Perceptions Through Craft

The goals I was most passionate about for this course were to invite participation from a diverse cross-section of students and to vastly broaden students' perceptions of what computer science is "good for".

A total of 69 students enrolled in *Craft of Computing* over the two-year interval, many of which were first year students. At Berea College, first year students are placed into Fall Term courses by an advisor, while all students self-enroll for the Spring Term. Looking only at Spring Term (self-enrolled) students, 19 men and 15 women enrolled in the course. While a small course like this is hardly suitable for reporting statistics, this is 44% female enrollment, as compared to under 20% of computer science bachelor's degrees being awarded to women nationwide [96].

I am also very encouraged that student evaluations reflect success in broadening perceptions of computing:

> *"Every assignment is flexible; the criteria can be met with an incredible number of solutions of varying complexity, and every assignment feels like it has the potential to be an art project."*

"... the course did well to link the computational and physical areas, generally broadening the scope for which I might consider programming."

"I learned so incredibly much! Before this course, I didn't know the first thing about programming, but now, I'm coding simple video games in my free time, and I have even decided to minor in computer science. I loved this course so much that I applied for, and was granted the opportunity to work as one of the two TAs for the course in the fall."

"... her ability to relate the computing information to things in the real world, and to combine digital work with analog work is simply astounding."

"This was a great opportunity to learn new things in a new field. I liked being able to bring my code to art."

Personally Meaningful Work

Lastly, *Craft of Computing* student projects frequently reflect students' identities, relationships, and milestones. Students have created homages to best friends and parents, gifts for their children, and decor for their dorm rooms. Many students have chosen to integrate their projects into their everyday lives – for example, affixing their vinyl cut stickers to frequently used items, sharing their projects on social media, and asking for work to be graded ahead of schedule so that it can be given as a gift. This seems especially promising, as research has shown that engaging in

personally meaningful work can attract historically marginalized students. A couple of examples of student final projects celebrating personal identity and accomplishment can be seen in Figure 2.9.

Figure 2.9. Personally meaningful student work. Left: one student represents herself by coding and pen-plotting a face that is half African and half Native American. Right: another student celebrates her graduation by hand-embroidering a computational design and framing it in a shadowbox with keepsakes.

2.6 Recommendations

In this section, I make concrete recommendations to others who may be interested in offering a similar course at their own institution.

Course Logistics

Longer course periods really do offer more time to engage with physical materials, fabrication equipment, and handcraft. They also allow for more meaningful discussions, both on the topic of readings and on days that students are presenting

their work. I recommend scheduling a course of this type during extended time blocks, if your institution offers this option.

The earliest offerings of the course were taught in a couple of different classrooms that did not have storage for materials or student projects – nor did they have the equipment used in the class. I spent a lot of time shuffling both materials and students between locations, to ensure that we had access to everything we needed. If at all possible, I recommend scheduling a course of this type in a lab space – ideally one in which students feel at home.

In terms of resources, students adored *Learning Processing* as a textbook. They appreciated the author's conversational tone and the workbook-style exercises, which we leveraged both for homework and in class. Students had no trouble skipping ahead to specific topics of interest on their own if they were looking for a challenge; in fact, one of the most common pieces of feedback received regarding the textbook was simply a desire to have covered more of it. Some students also independently sought out the textbook author's accompanying videos which explain and demonstrate key concepts, and told us how helpful these were. In short, a textbook and related videos that *directly support student activities* worked well.

Students initially complained a bit about all of the infrastructural pieces (Moodle, Trello, Slack, Google docs, etc.). I emphasized Trello and Slack as the most important resources to keep track of, encouraged students to configure Slack notifications to

their phone or email, and took great care to appropriately link between platforms. Ultimately, many students did find Slack to be a helpful touchstone throughout the term. I believe messaging on Slack feels closer to a text message than an email, and that this allows students to very quickly get in touch without worry over formality, etiquette, or perfect phrasing. Thus, I do recommend using Slack – or another platform emphasizing approachability, to facilitate announcements and peer support between course meetings.

Equipment & Materials

As mentioned in my earlier reflections, working with physical tools and materials is uniquely challenging – especially within a computer science context, where students are used to having everything they need to complete assignments right on their laptops. Over time, I have found a few strategies that helped to ease this.

I let students know in the course description, and again on the first day of class, that this course will require visiting the evening lab and/or scheduling separately with equipment. This helps to set student expectations early on and to redirect students who may have an incompatible term schedule.

I also make sure to offer plenty of unstructured time to work on projects during course meetings. This is especially helpful when learning about circuits, doing any kind of handcraft, and when students are pursuing open-ended final projects. Because of the demands of students' labor schedules, this helps to ensure that students have the

necessary time, support, and access (to equipment, materials, and instructional staff) to achieve course goals. Depending on your institutional context, you may choose to do this as well.

To keep the class on schedule, I recommend purchasing any necessary equipment and materials before the start of the term. The one exception to this is students' final project materials, which students request through a Google spreadsheet. If your department does not have funding available to cover these needs, you may consider charging a materials fee for the course.

When selecting equipment, simple is best. Early course offerings incorporated a laser cutter and vinyl cutter housed in a neighboring department, plus a powerful-but-experimental plotter. By the time of the course's latest offering, I had scaled back to mostly relying upon a Cricut machine (which can both cut and draw). The Cricut software is also easier for students to learn and to stick with over multiple assignments, especially compared with learning one application for a laser cutter and another for a vinyl cutter.

File Formatting & Exporting

It can be confusing for students to envision how the display output of their coded designs will translate into the physical world. For each Processing-based mini-project, it is important to be clear that the goal is generation of static images – not animations – because only vector data will be used to generate machine output. Students are

welcome to embellish their code with color and fills on their shapes, but it is important to emphasize that only the line data will be used.

Although Processing does allow for exporting vector designs to PDF, some additional formatting is necessary. Most notably, Processing exports duplicate paths for each shape: one to represent lines and one to represent fill, even when `noFill()` is specified. The first time an assignment requires formatting student designs for a piece of equipment, I recommend doing a live demo of the steps required to achieve this. I include a quick overview of vector versus raster file formats and a brief-but-targeted dip into Adobe Illustrator.

My TAs have assisted with creating written documentation of this process for students to follow along with on their own. In addition, I provide some simple example files for students to practice this process on. I recommend ensuring that you have tested the entire workflow and that your students have access to any necessary software – even one lab computer with vector graphics software (e.g. Adobe Illustrator) installed will suffice. A dedicated lab computer can also be very useful for running the digital fabrication equipment, rather than having students “print” to this equipment from their laptops.

Emphasizing Original Creative Work

From the course's inception, I had intended for students to create original designs for every assignment, although I was open-minded about what that meant. However,

some students dedicated quite a lot of time to recreating familiar imagery with Processing, while technically meeting each assignment's coding requirements. In some cases, this meant spending hours plotting out an existing logo or character coordinate-by-coordinate, failing to leverage the built-in Processing functions I had wanted students to learn about – and missing out on the joy and creativity of algorithmic design.

In later terms, I added an explicit requirement that assignments consist of original creative work rather than anything derivative. This yielded better progress toward programming learning goals and also more interesting outcomes. Showcasing beautiful examples of prior student work, as they were accumulated, helped greatly with this as well.

Finally, from a student course evaluation: *"I suggest she buy more needle threaders."* This is indeed a great thing to keep in mind when teaching with textiles!

2.7 Future Work

Evidence so far – seen in student work, enrollments, and course evaluations – is promising. Further analysis of student course evaluations and institutional data can paint a more detailed picture of who enrolls in the course – not only students' gender, but also their declared major and reason for enrolling. The next step after that will be to leverage institutional data to understand if (and how) enrollment in *Craft of*

Computing impacts students' choice of major and/or path through the major. Finally, many *Craft of Computing* students have voiced interest in an upper-division level of the course… and it would be a wonderful experience to design and offer an advanced elective counterpart!

2.8 Summary

In designing *Craft of Computing*, I had hoped to further broaden participation within my own undergraduate department, and to expand students' perceptions of computer science. Anecdotal evidence suggests that the course was effective in doing so; the course enrolled students across a diversity of majors (including art, theater, and applied design) and students report a broader understanding of the field in their course evaluations. The work that students produced – as exemplified in this chapter – is uniquely creative and personally meaningful, piquing students' interests in the creative and varied possibilities of computing.

I hope that sharing my experiences and recommendations emphasizes the importance of diversified CS0 offerings and can, in particular, enable more courses of this variety at other institutions.

3 | Course Design for Retaining Broader Participation: *Open Source Software Engineering*

Shortly after my first experience teaching open source, via the OpenHatch workshop at Hartnell College, I discovered a vast body of existing research on student involvement in free and open source software (FOSS) projects. This research was motivated by many of the same characteristics I observed that day: open source projects offer students a means of developing a practical skillset, building a portfolio of work, participating in a community of practice, and using computing to impact society in a meaningful way. Much of this work was published by faculty involved with Foss2serve [37], a special interest subgroup of the Teaching Open Source community [140]. An established working group of educators and researchers, Foss2serve supports student involvement in humanitarian open source projects, specifically because of their social impact. I also learned of POSSE (the Professors' Open Source Software Experience), which is a multi-day professional development workshop offered by RedHat and Foss2serve to support faculty new to teaching open source [28, 91, 113]. Over the following two years – and while exploring dissertation directions – I participated in multiple POSSE-related workshops, during which I gained experience with open source tools and helped to develop curriculum and activities for college classroom use. I also learned from faculty at a variety of

institutions who were integrating open source contribution into their computer science courses and began envisioning how I might teach such a course of my own.

Upon starting my position at Berea College, I was invited to do exactly this. Shortly after my hire, I was offered the opportunity to take over Berea's upper-division software engineering class, which had most recently been taught in the context of open source software. This version of the course had previously been run one or two terms, taught by a Berea faculty member who had also attended and who I had met through POSSE workshops. Informed by POSSE best practices, feedback from students who had taken the earliest iterations of the course, and my own teaching experience, I decided to redesign the course with a focus on scaffolding student success despite an inherently unpredictable context: the wilds of open source contribution. In doing so, I leaned heavily on the resources and mentorship afforded by my involvement with Foss2serve. I have remained involved with this faculty research community ever since, most recently collaborating with Lori Postner and Darci Burdge to offer a workshop at the 2018 Grace Hopper Celebration of Women in Computing on the topic of candidate project evaluation for student involvement.

While *Craft of Computing* offered the opportunity to impact student perceptions of computing at the entry level, designing an open source elective afforded the chance to do so at another critical moment – as students considered whether they would pursue

a career in computing post-graduation; effectively, it offered an opportunity to *retain* a diversity of students who had already been *attracted* to computing.

3.1 Introduction

Teaching open source software development has gained traction in undergraduate curricula for many reasons: students learn to use real-world tools/processes, build portfolios of project contributions, and function within a distributed professional community [30, 47]. Open source also provides a clear avenue for students to have a positive and tangible impact on society, something that is known to be relevant to broadening participation in computing [26, 80].

Open source contribution also showcases an application of computer science that students may not have been aware of prior. At the upper-division course level, this is especially important, as students are about to decide whether they will seek a career in computing post-graduation [138]. The issue of post-graduation retention is especially critical at Berea College, where many students are working to overcome socioeconomic disadvantage.

Over the course of two consecutive terms, I developed, taught, and heavily revised an upper-division course entitled *Open Source Software Engineering*. I leveraged a wealth of existing activities and resources in designing the course [37], was supported with real-time mentorship as I ran the course (from other faculty involved with

POSSE), and secured sustained professional mentorship for my students (from the Mozilla DevTools project).

The difference between the two terms was tremendous. Most notably, student project contributions increased from a 25% success rate in the first iteration to 100% in the second iteration. As a result of the course's success, it was retained in the department's permanent course catalog. In this chapter, I detail my experiences across these two terms and the revisions that I believe to be responsible for the improved student experience and outcomes in the second iteration.

3.2 Related Work

There exists a growing body of research on undergraduate engagement in free and open source (FOSS) projects, especially humanitarian free and open source projects (HFOSS). It has been well established that involving students in open source communities offers a valuable opportunity for students to learn within a community of practice, gain experience with practical tools (e.g. version control systems, bug trackers), build a portfolio, and contribute to a real-world project [29, 31, 46]. In addition, teaching with open source reaps the benefits of project-based learning – for example, helping students cultivate "soft" skills such as teamwork, communication, and project management [87].

HFOSS projects, in particular, have been a deliberate choice for many educators because these communities are typically welcoming and supportive to newcomers [48]. It is suspected that these communities also attract participation from women and other underrepresented minorities, due to their social impact [112]. Additionally, the altruistic nature of humanitarian open source contribution lends itself nicely to service-learning [90]. Teaching open source also offers instructors the chance to model a growth mindset and to foster a sense of belonging within a professional community – also of great relevance to broadening participation [49, 64, 132].

Much like *Craft of Computing*, this course also leverages more general pedagogical techniques with a track record of supporting student learning and broader participation – namely, active learning and pair programming [38, 82–84, 111, 134]. Also as with *Craft of Computing*, my teaching style embodies a guided discovery approach, meeting student inquiries with my own series of questions, designed to lead them incrementally to the answer or resource they may be seeking [2]. In this course, I especially make a point of offering process-oriented praise, as student learning is not always reflected in project contributions – and nonetheless I aim to support their development of a growth mindset [27].

A number of successful open source courses pre-date my own course design, situated within a variety of institutional contexts [11, 48]. These range from single term courses, featuring a taste of open source, to immersive year-long capstone courses or

those aimed specifically at broadening participation [10, 55, 133]. In designing my own course, I sought to translate existing best practices to fit within our liberal arts upper-division elective context. Given that so many Berea students come from historically marginalized communities, I also sought to showcase a socially impactful and community-oriented application of computing, in hopes of retaining students in the field post-graduation.

Despite all of its promise, teaching open source presents many curricular challenges: community leadership can take unexpected turns, projects vary in size and complexity, and student learning can be difficult to assess [30]. In addition to putting the aforementioned research into practice, my work reports on what I have learned; scaffolding student learning in such an unpredictable context is challenging, but thoughtful planning and revision can have a dramatic positive impact on outcomes.

3.3 Course Overview

Although my course is titled *Open Source Software Engineering*, the emphasis is much more on open source than on software engineering practices. The first half of each term is spent on history, etiquette, culture, and tools – and the second half is spent diving into an active open source project.

The first offering consisted of 16 students (14 male, 2 female) and the second offering consisted of 12 students (10 male, 2 female). Both classes represented a wide range of

experience, as some students had only taken CS1 and CS2 while others had completed a variety of upper-division coursework. An overview of the two offerings follows.

Term 1

I leveraged the Foss2serve library of activities for the first half of the course, guiding students through licensing, candidate project evaluation, version control with git, communication tools (such as IRC and Slack) and more. Students completed these exercises in pairs, in class. I also required students to make GitHub accounts and, after an in-class crash-course on HTML and CSS, students practiced fixing up a buggy GitHub Pages site that was created in advance. I generated several GitHub "issues" for students to claim and work on, to help learn the GitHub workflow and to practice HTML/CSS. (This activity was borrowed from OpenHatch.)

Weekly reading was assigned from either *The Cathedral and the Bazaar* [117] or *The Art of Community* [5]. Students were required to create blogs and to post reading responses there. A few group discussions were held in class on related topics.

For the second half of the term, I embedded all students in the same open source project rather than each team selecting their own project. Students weren't very excited about this approach, but I felt that it would be easier for me to support them – and for them to support one another. I solicited suggestions through the Teaching Open Source [140] mailing list and learned of others' positive experience engaging

with the Mozilla Firefox DevTools community [35] and, more specifically, with the debugger.html project [53]. (Mozilla originally developed this debugger as part of the Firefox Developer Tools, although it now works in both Firefox and Chrome.) I connected with two other faculty members teaching with DevTools, Heidi Ellis and Darci Burdge, and together we worked with the debugger.html community to identify candidate bugs for our students. (I knew both Heidi and Darci through POSSE, as they were both a part of the core group of faculty organizing and hosting the workshops, Foss2serve.) Motivated by a desire to engage students in HFOSS, we selected bugs under the umbrella of accessibility.

I divided students into teams of four. I grouped students according to their own preferences and who I thought would work well together. One consequence, however, was that most teams reflected a broad spectrum of prior experience. I instructed students to assign themselves relevant homework between class meetings – e.g. tutorials, testing, or bug research – and asked that they reserve class time for team collaboration. Students completed bi-weekly team evaluations, in which they ranked themselves and each of their teammates on metrics like regular attendance, leadership, and attitude. (This tool was shared with me by Heidi Ellis.) These were treated as confidential and allowed a window into any interpersonal challenges early enough to intervene.

Students were required to join the *Open Source Software Engineering* channel on the department's Slack team, which was created and is managed by teaching assistants. Each team was also asked to create *their own* Slack channel, in which they would briefly report out in writing at the start and finish of each class. I joined each of these channels as well. This practice was inspired by scrum/standup meetings; each student had to share what they accomplished outside of class and what they would spend class time working on that day. This gave students experience with industry practices and tools and also helped guide me as to which teams needed help getting unstuck. This practice also held students accountable, as teammates would be disappointed if someone had not done any work between course meetings.

During class, teams worked – sometimes altogether, sometimes in pairs – to make progress on their chosen bug. This often involved posting to communication channels used by the debugger.html project, including Slack and GitHub. Through our community interactions, it became apparent who a couple of particularly helpful Mozilla developers were, and we leveraged their support through the rest of the term.

The debugger.html project is built in React [118], which is not covered anywhere in our departmental curriculum. No structured support was provided for learning React; instead, each team sought out materials to learn the basics, and sometimes students shared resources across teams.

Three out of four teams got so far as to submit pull requests on GitHub. However, only one team's contribution was accepted and merged. The other teams got stuck in the review process – or in one case, were unable to even fully solve/address the bug they had been working on all term.

Term 2

I made significant changes to the course, both in response to student feedback and my own observations. I also hired a TA who had previously taken the class and was able to provide feedback based on his experience as a student in the course. Below, I summarize the major changes.

Students reported getting little out of the readings from *The Art of Community*, so I dropped that textbook. Because discussions had been sparsely participated in, I spent less class time on them and instead asked students to reflect deeper in their blog posts. These changes won more class time for working in pairs or teams.

Students from the prior term struggled with learning React and expressed frustration with each team discovering the same resources on their own. In response, I asked the course TA to develop and lead a walkthrough in which students built a barebones blog using React. I also created a shared virtual bulletin board (using Trello [130]), where students posted resources that their classmates might find useful.

I chose to involve students in the same project as before: debugger.html. This time, students were grouped with those of *similar experience level*, allowing each team to choose an appropriately challenging bug to tackle. Students were also grouped in *teams of three instead of four*, as I suspected this might lead to more consistent communication within teams. Teams did collaborate more effectively this way, with each member contributing more equally to conversations and to code.

The Mozilla developers that we encountered in Term 1 brainstormed with me about how to better support students through the contribution process. We decided to identify smaller issues – or even subtasks of issues – for teams to claim, even if it meant shifting focus beyond accessibility. We also established a separate Slack channel on the DevTools team, which both the students and the developers joined. This offered a less intimidating venue for students to ask questions. Students were required to cross-post their scrum reports in this channel, so that the developers could track their progress in greater detail. I believe that this gave students' self-assigned homework a greater weight, as they were reporting to real-world developers, and not just to their college instructor and classmates. I also added in-class standup meetings on a weekly basis, in which teams would report out to one another on their progress and share learned expertise.

The teams of three grappled initially with how to collaborate during class; no longer could they divide-and-conquer by splitting into pairs. They began making use of large

portable screens in the classroom, which they wheeled to their desk clusters and took turns connecting their laptops to. This facilitated much richer discussion about each team's progress, as teams could analyze code, write code, or sift through resources together. Instead of watching pairs head-down at their laptops, I saw teams engaging in lively discussion, moving around and using the screen as a prop. This also made it easier for me to circulate throughout the classroom and monitor each team's progress, joining their discussions when helpful.

Towards the end of the course, the Mozilla developers who were supporting my students offered to schedule a video call during class time. We structured this call as a standup meeting, during which each team reported out on their weekly progress and had the opportunity to receive real-time feedback. Students were also able to ask the developers about their personal experience getting into open source.

Each of the four teams made at least one successful contribution to the project. One team made three, spanning both code and documentation!

3.4 Student Feedback

I asked for informal feedback throughout both terms in which the course was offered. In Term 1, students expressed a large degree of frustration and confusion (although they responded to it with a constructive attitude), while students in the second term openly and enthusiastically affirmed that they were having a positive learning

experience. Unsolicited, I received the following from a student via email, about halfway through Term 2:

> *"So far I am genuinely enjoying the course. The work is not too overwhelming and it feels manageable. I really like how we are encouraged to try things and learn on our own. It is building my confidence as a woman in computer science."*

I also received valuable feedback on the course through students' course evaluations, submitted at the end of each term. A couple of comments following Term 2:

> *"I did learn a lot. I feel much more comfortable with my computer, with web development, with open source, with communicating, with teamwork, and everything we touched on in class."*

> *"I learned more about open source development than I even expected to in this course. I think the idea of having students contribute to a real piece of software is amazing and it is a piece of software that millions of people, including myself use. Interacting with the Firefox community was very educational both in a coding aspect and in a... well, community aspect."*

Comparing quantitative evaluation data, students from Term 2 spent more hours per week on the course, reported learning more, and rated the course higher overall.

3.5 Reflections & Recommendations

It may seem obvious that a course should improve in its second offering, due to the instructor's increased familiarity with the material/structure and access to an experienced TA. However, this course improved dramatically, and despite a continually shifting context. Below, I summarize what I believe to be the most influential factors.

Project Choice

Project choice is arguably the most foundational factor in the success of any class structured around open source contribution. I recommend, when possible, embedding all students in one project/community; this allows the instructor to understand and support student progress while also staying in touch with a single set of community leaders. As is emphasized by Foss2serve, I also recommend verifying that the community is highly active and welcoming to newcomers; this will ensure that students receive timely, constructive responses to questions and pull requests. In our case, debugger.html's active Slack channel also meant that students could observe community norms before wading in themselves. The ideal solution will vary widely given the number of students in a course; for example, it might be overwhelming to embed a class of 50 or 100 students in a single community.

Selecting a project that is well-known and/or humanitarian in nature allows students to have real-world impact. Additionally, selecting a very active project with clear

documentation and a welcoming atmosphere can help cultivate a sense of belonging. A project which leverages current/relevant tools or languages – and which uses a major platform to track contributions (such as GitHub) – will also help students to develop professional skills and a visible portfolio.

Team Formation & Tools to Support Collaboration

After attempting two different strategies for assigning teams, I feel strongly that it's best to group students with others of similar experience level. This way, less experienced students do not fall behind or lose confidence – while more experienced students can take off and run with a more difficult problem. I also observed that teams of similar experience level naturally gravitated towards bugs within reach of their expertise; this further allowed all team members to engage equally in the process and to reach an affirming outcome.

As time permits, the more experienced teams can also provide support to those that are stuck. This is easily facilitated by the addition of class-wide standup meetings, during which stalled teams can solicit help. I also recommend team-specific Slack channels for communication/reporting, along with a class-wide channel for students to ask questions and share resources between class meetings.

Finally, a team size of three – along with access to large shared screens – encourages lively discussion and equitable collaboration within each team.

Structuring Unfamiliar Tools & Technologies

Although having to learn new technologies – React, in this case – was not the insurmountable obstacle I expected it to be, it really helped to provide some structure around this in the second term. I recommend providing infrastructure for students to share resources, as I did with Slack and Trello. I also recommend offering project-specific demos and/or walkthroughs for students to build experience with any required tools or technologies. I believe that these things empowered students to more efficiently and confidently jump into working on their bug/issue. (Note that this did not deprive students of the opportunity to feel "productively lost"; there was still plenty of independent learning to be done!)

Professional Mentorship

Professional mentorship was a vital thread running through the entire course experience. I believe this to be, perhaps, the most influential factor in the course's improvement. When the Mozilla developers became more involved in Term 2, students responded with greater motivation and a stronger sense of accountability. Although many students initially found it intimidating to communicate directly with the developers, doing so pushed them to practice communicating with professionalism and specificity. These developers modeled a growth mindset; not always having the answers, but coaching students through finding resources and

learning on the fly. They provided continuous and timely feedback via Slack and GitHub, doing so with proficiency, patience, and encouragement.

Students responded especially positively to the standup video call that we did towards the end of Term 2. Knowing the call was on the horizon motivated them to make progress as a team and to generate interesting questions. Additionally, they valued seeing that the developers were people that they could relate to; approachable individuals who once had very little experience with open source themselves. In this sense, the developers that mentored my students became very effective role models for them.

I also benefited from mentorship – both from the developers (with whom I could check in about student progress and impact on their community) and from other faculty teaching open source (who offered mutual support and years of experience). For most of Term 1, I had a standing weekly call with the two other faculty members embedding their students in the debugger.html project – and I kept in close contact with the Mozilla developers via Slack throughout both terms. This mentorship helped *me* to maintain a growth mindset as I guided students through unfamiliar content and processes; I often needed to remind both myself and my students that my role in the course was to *guide* rather than to *instruct* them.

For the above reasons, I emphatically recommend reaching out to others teaching with open source, as well as securing mentorship for your students within your chosen project.

3.6 Future Work

Although I am confident that students learned more in the second offering of the course, assessing actual student learning in open source is challenging. In the case of this course, each student entered with a different level of experience, and it was important to me that students make progress relative to their own starting points. Blogs allowed a window into each student's process, but most students did not seem motivated to complete these assignments thoughtfully nor in a timely fashion. Students echoed these sentiments in their course evaluations, along with an explicit desire to be assessed on their technical contributions. Adding an assessment of students' concrete technical contributions would serve as a good motivator in future offerings. Students, in fact, wrote openly about this in their course evaluations:

> *"It's a lot easier, psychologically, to work hard on the blog posts and written assignments, because they were graded. It's hard to get working on the coding and researching because it's not directly graded, and it wasn't hard to do a little bit and then write an enthusiastic blog post that gets full points."*

> *"When choosing between a graded assignment in one class and an ungraded assignment in [this course], it's very hard to not choose the graded one. The only way to combat this is to somehow make the open source work graded. I don't know how this could be done. But as long as the only graded work in this class is the blogs and writing assignments, it will be too easy to slack off on the open source work."*

Although I perhaps define success and learning in broader ways than my students, it seems that adding an assessment of their concrete technical contributions would serve as a good motivator for them to expend time in that arena.

Given the positive response from our single standup scrum video call in Term 2, I believe scheduling those meetings more frequently would be beneficial – and the developers volunteered to do so in a future course offering.

Finally, a term-length course is barely long enough for students to dip their toes into an open source project. Many students have indicated interest in continuing their involvement in debugger.html and I would love to advocate for this to become an option for satisfying the department's senior project requirement.

3.7 Summary

Open source is an uncertain and constantly shifting landscape within which to situate an undergraduate class – but the benefits are vast when well-executed. It is tricky to

get right; despite my participation in professional development workshops and connection with more experienced colleagues, students in the course's first offering were less successful than I would have preferred. The second offering resulted in substantially higher student success and I believe these gains were attributable primarily to thoughtful team formation, structuring unfamiliar tools and technologies, and professional mentorship. I hope that these findings are of use to others setting out to teach similar courses.

4 | A Case Study in Expanding Access to Electronic Textiles: *The LilyTiny*

Ten years ago, and as part of my master's thesis, I designed a simplified sewable microcontroller based on the LilyPad Arduino toolkit and released a companion project-based e-textile curriculum along with it. This work was motivated by the inaccessible cost and complexity of teaching introductory electronics and programming at the time, despite the potential for these activities to appeal to historically minoritized populations and potentially help build self-efficacy.

The resulting circuit board is known as the LilyTiny and is now commercially available through collaboration with SparkFun Electronics. Taking advantage of the resistance inherent in conductive thread, the LilyTiny simply breaks out each pin of an ATtiny85 microcontroller which is preprogrammed with a variety of light behaviors. Depending on how they are connected, the LilyTiny can drive an LED to blink, randomly twinkle, fade on/off in a heartbeat pattern, or fade on/off in a breathing pattern. The LilyTiny may also be reprogrammed by the user, thus expanding its utility to teach both circuit-building *and* programming skills.

In the time since the release of the LilyTiny (as a commercial product) and an accompanying workshop guide, I turned my focus to other projects, doing little to promote their adoption. My dissertation research returns to this body of work,

specifically to examine what happened during that time period; did the release of these resources "into the wild" improve access to e-textiles, as I had hoped?

4.1 Introduction

Electronic textiles, also known as “e-textiles” or “soft circuits", are electrical circuits created using flexible conductive materials (such as conductive threads and fabrics) in conjunction with discrete electronic components (such as lights, batteries, switches, and sensors). This domain has long been gaining traction as a creative and approachable avenue into computing; utilizing craft materials and techniques, it invites diverse participation, broadens perceptions of what electronics and computing are "good for", and supports the creation of a very different kind of artifact when compared with traditional electronics prototyping materials [17].

The LilyPad Arduino was introduced in 2008 as a commercially available e-textile toolkit, enabling anyone to build their own soft, wearable, sewn – and programmable – circuits [14, 15]. In the years to follow, Adafruit released a similar toolkit, known as the Flora [128]. In addition to supporting individual artists and makers in realizing personal projects, these toolkits also opened up the possibility of teaching electronics and programming with e-textiles. Indeed, research has found this to be a fruitful avenue for broadening participation in computing, teaching electronics and

programming, and inspiring a new class of beautiful, computational, and personal artifacts [56].

Despite these successes, I observed critical resource gaps preventing widespread adoption of e-textile learning activities, especially at the K-12 level. In particular, I noticed that many educators did not have access to the budget required to secure relevant tools and materials at scale. Additionally, I noted a lack of instructional materials to support educators in preparing for and facilitating such activities.

I also noticed a scaffolding "valley" between simple projects involving only lights, batteries, and sewn connections – and more advanced projects leveraging the programmable LilyPad Arduino. I designed the LilyTiny in an attempt to bridge this valley; each LilyTiny is pre-programmed with several LED behaviors, inviting conversation about the power of computation without requiring students to write (or even understand) code.

This chapter summarizes my experience developing a low-cost sewable microcontroller, known as the LilyTiny, and a workshop guide to support it – work undertaken to address the aforementioned resource gaps in hopes of broadening access to e-textiles. I also share the results of my inquiry into the impact of this work, several years having elapsed since I created the LilyTiny – now a commercial product sold by SparkFun Electronics. My investigation includes a survey of derivative

products, a multi-year analysis of sales data, and examination of customer reviews and projects.

4.2 Related Work

The development of the LilyTiny was made possible by years of prior research in physical computing, electronic textiles, and education.

Physical Computing

In the realm of physical computing, two projects in particular directly paved the way: the Arduino electronics prototyping platform and, later, the sewable LilyPad Arduino. Arduino was initially developed to enable rapid prototyping without specialized engineering expertise [85]. The LilyPad toolkit extended this functionality to a textile context, thereby inviting participation from diverse populations as well as enabling the creation of soft, beautiful, computational artifacts [14, 15, 17]. Both of these projects pioneered the now-ubiquity of physical computing – not only by their very design, but also by their mass availability and pricing suitable for hobbyists, artists, and students. They both leverage an open source hardware (also known as "open hardware") model, allowing others to modify the PCB layouts for personal use or derivative products. My work extends these efforts, attempting to make e-textiles more accessible and affordable to a broader audience.

Electronic Textiles & Computing Education Research

The LilyPad Arduino has been extraordinarily successful in leveraging handcraft practices and materials to draw in demographics historically excluded from engineering (most notably, women). This has been evidenced by a much larger proportion of the LilyPad Arduino market share being female purchasers when compared to the classic Arduino – and by an emerging design community at the intersection of aesthetics, craft, and computation [17].

Significant work has also gone into the development of curriculum to support adoption of the LilyPad Arduino [14, 16, 43, 58, 59, 116]. This work affirms how highly I valued developing curriculum to support the LilyTiny hardware.

Ngai, et al. have developed two modular platforms for wearable computing, TeeBoard and i*CATch, to bring computational textiles into the classroom and teach basic programming [99–101]. More recently, Hill, et al. introduced the ThreadBoard, for rapid prototyping of e-textile circuits [45]. These projects represent critical strides in the mission to expand educational access, although these tools are not yet available to the general public.

In parallel to the development of new e-textiles tools and kits, there has been a great deal of research into the impact of teaching with e-textiles. For example, studies have demonstrated the utility of e-textiles as a means to develop students' STEM/technological self-efficacy, teach debugging, develop computational thinking,

experiment with aesthetics, and create culturally relevant artifacts [34, 57, 58, 60, 77, 122]. This body of work has inarguably established the value of e-textiles as an avenue for effectively broadening participation in computing, especially at the K-12 level and in after-school settings [19, 56]. Broader impact of this work has been limited, in part, by the funding required to secure necessary tools and materials, as well as access to a variety of instructional resources to support educators. I directly sought to address these limitations.

Instructional Design for K-12 STEM

Experienced educators and organizations have been disseminating resources for STEM learning long before e-textiles activities came to be. In particular, WGBH (a PBS affiliate, now known as GBH) has a long history of publishing K-12 activity guides for use in classrooms and at home. (The Design Squad guides are an excellent example of this [107, 135].) The National Center for Women & Information Technology (NCWIT) also offers "in-a-box" programming on topics including computer science "unplugged" (in-person, off-screen activities), outreach, and pair programming [97]. The design of my workshop guide drew heavily on the format of these successful resources, expanding their domain coverage to include e-textiles. (NCWIT has since released an "e-Textiles in-a-Box" program [33].)

Independent Learning Resources for E-Textiles

In addition to resources for educators, there has been an explosion of resources for individuals to independently learn new skills or complete projects related to making, crafting, and prototyping. At the time that I developed the LilyTiny, a handful of project-based e-textiles books had been released: *Fashioning Technology, Switch Craft, Fashion Geek, and Open Softwear* [32, 63, 102, 105]. Around the same time, MAKE Magazine – and the shorter lived CRAFT Magazine – were gaining popularity as monthly publications, containing example projects, relevant news/products, and profiles of prominent makers/crafters. Since the development of the LilyTiny and accompanying curriculum, two additional DIY e-textiles books have been released: *Make: Wearable Electronics* and *Sew Electric* (the latter containing an activity featuring the LilyTiny) [20, 41].

In addition to print resources, the internet has been host to a number of free, digital DIY resources over time; websites like Soft Circuit Saturdays and How To Get What You Want have reflected independent efforts to share e-textiles resources [50, 126], while structured tutorials have offered guidance to independent learners in the craft/technology realm [61, 62, 71]. Instructables has served as a valuable platform for many of these, especially as leveraged by prominent e-textiles artists/makers like Becky Stern and Hannah Perner-Wilson [51, 52]. SparkFun Education has expanded

these offerings in recent years, in particular supporting the LilyTiny with detailed documentation and tutorials [127].

My work builds on the success of many of the aforementioned projects, with an emphasis on lowering prevailing barriers of cost and know-how, while uniquely striving to support educators guiding many learners in parallel.

4.3 Design & Development

The LilyTiny and accompanying workshop guide were created to address known barriers to broader adoption of e-textiles in educational settings. In designing these materials, I sought to overcome challenges of cost, know-how, and also to provide a bridge to integrating computation and learning about microcontrollers without having to program. It was my hope that this work would expand access to electronic textiles as a creative way into computing.

The LilyTiny

My goal for the LilyTiny was to create a sewable microcontroller at a much lower price point than the LilyPad Arduino, and one which arrives pre-programmed, allowing users to incorporate computation in their projects without writing code. I designed around the ATtiny85 microcontroller because it is very inexpensive, yet is powerful enough to support pre-programmed behaviors such as light patterns. My breakout board was based on the LilyPad Arduino accelerometer board layout, which

is open source and available under a Creative Commons License. The LilyTiny is about the size of a quarter.

I used a milling machine to make the first prototype of the breakout board (see Figure 4.1). ATtiny chips were soldered by hand to each milled board, after which the broken out pins ("petals" in LilyPad terminology) were color-coded with permanent markers. I programmed these early prototypes one-by-one using an early prototype of SparkFun's Tiny AVR Programmer which attached to the petals of each board using alligator clips.

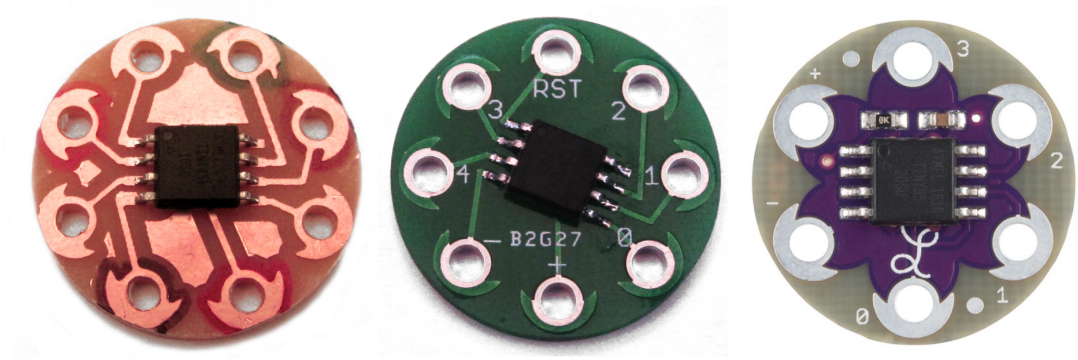

Figure 4.1. LilyTiny prototypes, from left to right: initial milled circuit board, custom-ordered factory board, final commercial product (sold by SparkFun Electronics).

After these boards were manually tested and successfully used in a pilot workshop, I placed a custom order with a circuit board manufacturer. This version included appropriately labeled pins and was more reliable than the first. This time, I used batch reflow soldering to affix the ATtiny chips, after which the boards were again programmed individually with a Tiny AVR Programmer prototype.

Following testing and an additional pilot workshop, we partnered with SparkFun Electronics to release the LilyTiny commercially as part of the LilyPad Arduino toolkit line of products. (My advisor, Leah Buechley, and labmate, David Mellis, guided this process.)

All versions of the LilyTiny prototype were programmed with the same Arduino code, allowing a user to access four different light patterns depending on which output pin/petal they sew an LED to. These include: blinking on/off, a breathing pattern, a heartbeat pattern, and a random twinkle pattern. I chose to pre-program the boards in this way to invite discussion of computation without the user having to write or understand code, meanwhile offering out-of-the-box access to creative and computationally interesting behaviors. This filled a gap at the time between lower-tech projects involving only LEDs and batteries – and more complicated projects leveraging a LilyPad Arduino which must be programmed before use.

For more advanced users, the LilyTiny offers a lower-cost means of incorporating computation into a project, as it can be reprogrammed using a Tiny AVR Programmer and the Arduino software.

The LilyTiny debuted for sale through SparkFun for about $10, but its price has hovered closer to $5 for the majority of the years since introduction. This makes it possible for educators to consider purchasing in bulk for workshops or classrooms.

Companion Curriculum

To support adoption of the LilyTiny, especially amongst a target audience of educators, I developed and self-published a companion workshop curriculum entitled *Getting Hands-on with Soft Circuits*. I made this curriculum available for free on the internet and also for ordering in hard copy format.

This curriculum was designed as a standalone resource, providing just enough on-demand information for educators/facilitators to guide students through an informal activity. This includes necessary know-how relating to both sewing/crafting and to electronics.

The curriculum includes a series of five workshop activities leveraging e-textiles as a means to explore circuits and computation, some of which are shown in Figure 4.2. These activities are designed in sequence, such that each activity builds on the concepts of those preceding it – but also such that one could choose different activities to workshop, depending on prior experience. Activities 4 and 5 make use of the LilyTiny, with the preceding activities building foundational e-textile skills.

Figure 4.2. Sample activities from the workshop curriculum.

Each activity includes a photo of an example project, a list of tools and materials, a summary/overview, a list of learning goals, and directions on how to prepare for and facilitate the activity. When relevant, activities also include support materials, such as templates or handouts that can be given to students. Each project was designed to be doable in a two or three hour session, with the exception of the final activity which is better suited to a half-day workshop.

Activities 1 through 3 build foundational e-textile skills: an introduction to circuits with conductive thread, a primer on switches and how they control electrical flow, and an overview of parallel circuits and how they enable one battery to power multiple lights.

Activities 4 and 5 provide a high-level introduction to microcontrollers and the concept of programmability – without having to read or write code. In these activities, participants create light-up patches, using the pre-programmed LilyTiny to control the behavior of an LED (blinking, fading, twinkling, or heartbeat). This is first done individually, and then as part of a collaborative electronic patchwork quilt.

All of the activities were designed around low-cost, easily obtainable materials. These include craft notions (acrylic felt, sewing needs, snaps, beads, etc.) as well as off-the-shelf electronics components (such as through-hole LEDs, coin cell batteries, and battery holders). These items can all be sourced for less than 50 cents apiece. The

only tools required for these activities are readily available, such as needle nose pliers, scissors, and hot glue guns.

The workshop guide also includes a troubleshooting flowchart, a curated list of low-cost tools and materials, and pointers to additional print and online resources relating to soft circuits.

After the release of the workshop guide, I co-developed one more LilyTiny-powered activity with collaborators Natalie Freed and Jie Qi. This activity is entitled *Plush Monsters: Creatures with Character*. Originally developed for a large-scale workshop at the 2011 Grace Hopper Celebration of Women in Computing, we self-published this activity online afterwards [76]. The activity may be used as an add-on or independent of the workshop guide, as it includes its own curricular materials as shown in Figure 4.3. (The layout of the workshop guide is very similar.)

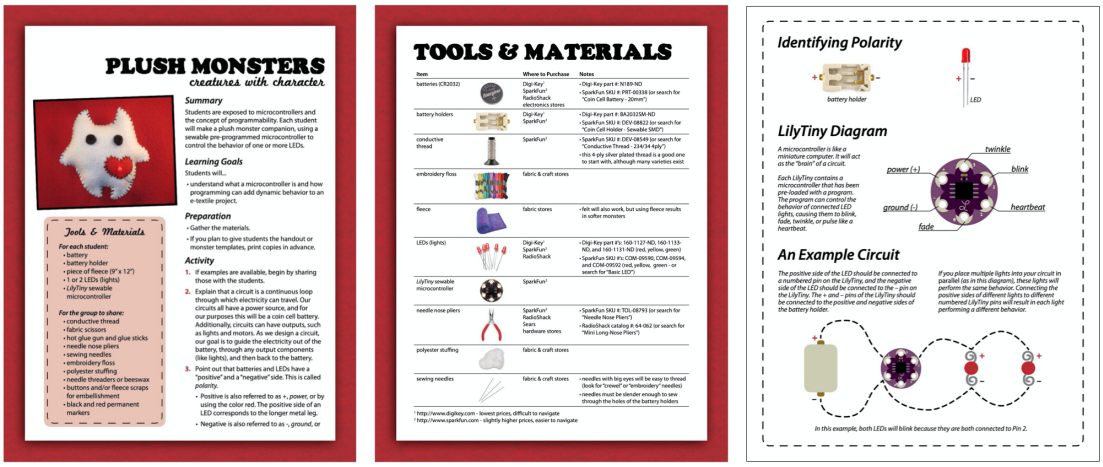

Figure 4.3. Select pages from the plush monster activity, which utilizes the LilyTiny.

Pilot Testing

Design of the LilyTiny and curriculum were guided by two pilot workshops run in parallel with the development process. These workshops were arranged in collaboration with an outreach center on MIT's campus and enrolled volunteer homeschool students who were already familiar with basic circuits. Over a two-hour session, facilitators taught students to create a light-up patch with an LED whose behavior is controlled by a LilyTiny, using Activity 4 from the workshop guide. Students participating in these workshops were familiar with basic circuits and electronic components – a similar level of understanding to that which is covered in the guide's first three activities.

I taught the first workshop using the earliest milled version of the LilyTiny. A total of 16 students between the ages of 11 and 16 participated (10 female, 6 male). 12 of the 16 students were successful in getting their LilyTiny to control an LED. Two of these students finished early and added additional lights to their circuits in parallel configuration. All of the students were offered the option of taking conductive thread and/or additional LEDs home to complete or augment their projects.

This workshop revealed a few areas for improvement – for example, sourcing more durable materials and fine-tuning techniques for novice sewing with conductive thread. These informed a revision of the curriculum and during this time, I also procured the second version of the LilyTiny circuit boards.

In order to test the usability of the materials by a third party, the second workshop was taught by an outside educator. I provided all of the physical materials, but asked her to teach the workshop using only the curriculum as a guide. 10 students participated in this workshop, between the ages of 11 and 14 (4 female, 6 male). I was present to observe this workshop, during which all participants were successful in sewing a patch containing a LilyTiny-controlled LED. Additionally, several students went beyond connecting one light to their microcontroller, adding additional lights with alternate behavior. This second workshop was reassuring that the curriculum adequately supported an accessible and scalable learning experience using the LilyTiny.

In addition to my own observations, I solicited extensive feedback from the educator who taught the second workshop, as well as from an expert STEM activity guide developer. This was invaluable in making revisions.

I also solicited feedback from students through surveys at the end of each workshop. Responses indicated consistency across the two workshops in terms of length, difficulty, and pace. This was preliminarily indicative that the instructional materials were transferable. When asked about future workshops, several students indicated specifically that they would like to learn how to program the microcontrollers themselves, or suggested projects that would likely require programming.

4.4 Measuring Impact

In an effort to understand the impact of the LilyTiny in the years that have elapsed since it was first introduced, I conducted a followup study involving a survey of derivative products, analysis of sales data, and a sampling of product reviews. Although SparkFun has repackaged the hardware in various ways, I have done little to promote adoption of the LilyTiny; thus, I believe my findings to be a true reflection of whether these research innovations met an educational need and had impact in the wild.

Derivative & Follow-on Products

I surveyed the marketplace for low-cost sewable microcontrollers released over the past ten years. A handful of related and derivative products are shown in Figure 4.4.

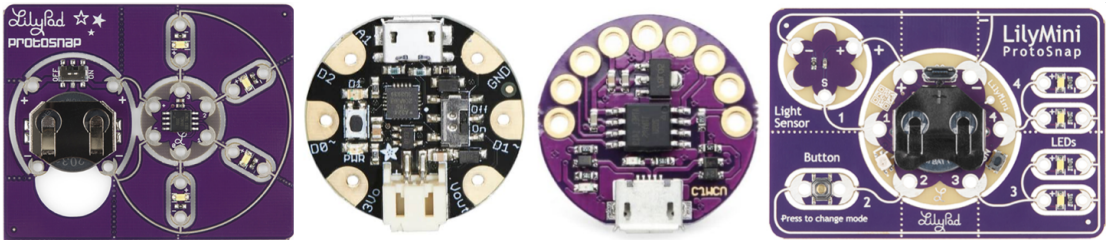

Figure 4.4. Derivative and follow-on sewable microcontroller boards. From left to right: LilyTwinkle ProtoSnap, Gemma, an unbranded clone, and the LilyPad LilyMini.

At the time that the LilyTiny came to market through SparkFun, a sister product was also released, known as the LilyTwinkle. The LilyTwinkle hardware is identical to that of the LilyTiny; the only difference between the two products is that the LilyTwinkle ships with a different Arduino program. Instead of each Lily petal

offering a different light behavior as with the LilyTiny, all of the petals twinkle lights at different rates. While the LilyTiny was designed to invite conversations about computation in educational settings, the LilyTwinkle is nicely suited to creating sparkling wearable projects – presumably appealing to a broader audience. In addition to these standalone products, SparkFun bundled the LilyTwinkle into a few different kits and form factors, including: a Firefly Jar kit to create a twinkling felt mason jar, a ProtoSnap kit allowing testing of the board prior to sewing, and an E-textiles Basics Lab Pack to support classrooms.

A little over a year after the release of these two products, Adafruit released the Gemma sewable microcontroller [54]. Like the LilyTiny, the Gemma also aims to be a smaller, more affordable version of it's full-scale, higher-priced counterpart, the Flora. The Gemma has undergone several revisions, evolving to focus on reprogrammability and now featuring an upgraded chip, mini-USB connector, on-board on/off switch and RGB LED. It currently retails for about twice the cost of a LilyTiny, at around $10. The pre-loaded code is not well-documented nor marketed as a selling point, but it does ship with example code.

More recently, in 2016, SparkFun released the LilyPad LilyMini, another small sewable microcontroller which arrives pre-programmed, uses an upgraded chip, and includes an on-board coin cell battery holder. Although the program it ships with

offers more interactivity than the LilyTiny and LilyTwinkle, it is sold at a higher price point ($16) and is much more difficult to reprogram.

A number of other sewable ATtiny85 breakout boards have been released in recent years. These boards are similarly bite-sized and typically manufactured using purple solder mask, like the original LilyTiny. However, these products feature a somewhat different arrangement of pins/petals and an on-board USB connector. They are sold under a variety of unbranded names such as "LilyTiny ATtiny85 Development Board", "MicroUSB LilyTiny", and "CJMCU LilyTiny". Although I was not involved in their development, the choice of naming leads me to believe they were directly inspired by the LilyTiny. These boards do not necessarily ship with any example code installed, requiring the user to make some modifications to the Arduino IDE in order to initially program them. These boards retail for $1-15 and are widely available from a variety of sellers on eBay, Amazon, and Alibaba.

The LilyTiny was born out of open source hardware development, as were all of the aforementioned related boards. While it is not uncommon for someone to clone or create a derivative version of a useful circuit board, I believe that the number and variety of products following in the footsteps of the LilyTiny are testament to a market need for a small, low-cost, sewable microcontroller – especially when compared with the more full-featured LilyPad Arduino and Flora. It is worth noting, however, that the only boards advertised with their pre-loaded programs as a feature

are the LilyTiny and LilyTwinkle. I believe this to be a particular asset and selling point for educational settings, as out-of-the-box functionality makes teaching time-constrained workshops/activities much more feasible. This feature also allows the introduction of computational behavior without the requirement to write code or navigate the Arduino upload/reprogramming process.

Sales Data

Next, I set out to understand the LilyTiny's impact on users. I use sales data as a proxy for adoption and investigate LilyTiny's position within the market; whether it has been successful since its commercial debut, and whether this has shifted with the release of similar products. My MIT advisor and creator of the LilyPad Arduino, Leah Buechley, helped me to obtain eight years of sales data directly from SparkFun, dating from the release of the LilyTiny and LilyTwinkle in July 2012 through the start of this investigation in June 2020. Because SparkFun is the only manufacturer of the LilyTiny and all LilyPad Arduino products, this data encompasses all sales, including those made direct-to-consumer and those made to distributors/resellers.

I first wanted to check whether our hardware has sold well as a commercial product. Indeed, it has; over this eight-year period, a total of 81,227 of our breakout boards (LilyTiny and LilyTwinkle combined) were sold. This includes boards sold individually as well those sold as part of a kit or lab pack. Our hardware shipped to 80 different countries across nearly 10,000 orders. The United States generated the

highest number of orders, followed by Australia, Canada, and the United Kingdom in that order. Both products leveraging our hardware have sold steadily as shown in Figure 4.5, each averaging over 5,000 units sold per year. I think these numbers make clear that our breakout board is satisfying a real user need – and continuing to do so long past the introduction of competitor products.

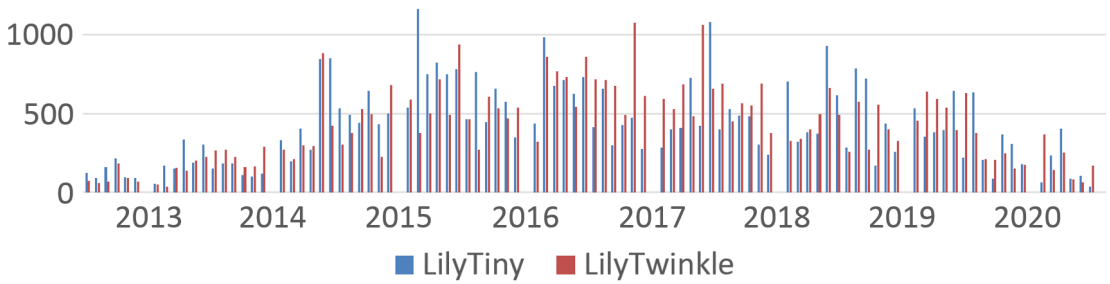

Figure 4.5. LilyTiny and LilyTwinkle monthly sales, showing sustained market interest over many years.

Second, I wanted to check the hypothesis that a very basic board with pre-installed software is a useful intermediary between simple circuits and more complex boards requiring programming. To do this, I looked at sales data across the entire set of sewable microcontrollers offered by SparkFun. Figure 4.6 shows market share for each individual product, kit, or lab pack. To my surprise, the individually packaged LilyTiny was the single most ordered sewable microcontroller during the eight year time period that I examined. However, many products are related to one another through upgrades or repackaging, and thus I grouped these products into conceptual families. Even after grouping, the LilyTiny/LilyTwinkle board was purchased as often as boards in the much more capable LilyPad Main family, with the

LilyTiny/LilyTwinkle board representing 46% of sales. This seems to validate that a cheaper simpler board has value to a substantial number of users.

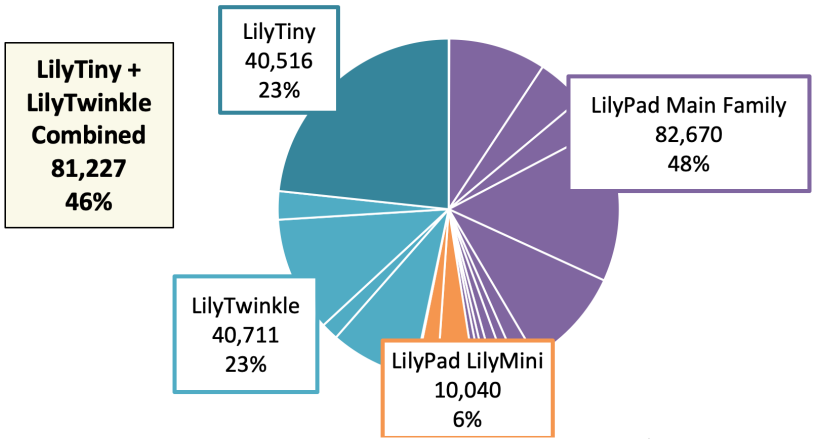

Figure 4.6. SparkFun sewable microcontroller sales, July 2012 through June 2020. Note that the LilyPad LilyMini was not introduced until 2016. Each color represents a different product family. Each pie slice represents a different product release (i.e. LilyPad Arduino 328 Main Board, LilyPad Arduino Simple Board, Firefly Jar kit, etc.). Kits are categorized by which board they include.

The data in Figure 4.6 also allows for comparison of sales between the LilyTiny and the LilyTwinkle. This is important to consider, as I designed the LilyTiny and its supporting curriculum with the intent of reaching educators – while the LilyTwinkle is likely to appeal to a more general audience. I had guessed that the hobbyist focus and additional marketing variations would have made the LilyTwinkle more popular. However, to my surprise, the LilyTiny has sold twice as many standalone boards as the LilyTwinkle – and about the same number of total units when considering all kits containing the LilyTwinkle. I believe this finding affirms that a board released with

appropriate curriculum and pre-programmed code, supporting the introduction of computation, invites broad adoption.

Finally, I wanted to know if the LilyTiny is being used by educators; that is to say, whether it has reached my intended market. The sales data doesn't directly specify who is purchasing boards, but it does tell us the quantity purchased in each order. Individual hobbyists probably buy a few boards at most, while educators typically buy in quantity appropriate for classrooms or workshops. (For this analysis, I excluded distributor orders since I am interested in individual purchasing patterns.) Figure 4.7 reports on order quantities for each product family. Indeed, a much greater percentage of LilyTiny orders include multiples of the product and the average units per order is higher, when compared to the LilyPad Main family, LilyTwinkle, and LilyPad LilyMini. This is true despite the fact that the LilyTwinkle and LilyPad Main boards were explicitly marketed in "lab packs" of ten units. I believe this provides evidence that the LilyTiny, with its choice of assorted programmed light behaviors and supporting curriculum, is likely being used for teaching more frequently than the more complex LilyPad LilyMini and LilyPad Main boards – or even its sister product, the LilyTwinkle.

PRODUCT	AVERAGE UNITS/ ORDER	% OF ORDERS CONTAINING QUANTITY 5+	% OF ORDERS CONTAINING QUANTITY 10+
LilyPad Main Family	4.2	12.1%	7.7%
LilyPad LilyMini	5.1	14.7%	8.9%
LilyTiny	**9.8**	**34.1%**	**24.9%**
LilyTwinkle	6.7	18.0%	11.2%

Figure 4.7. SparkFun sewable microcontroller ordering patterns, after adjusting for lab packs which contain multiple boards. Notice that a much greater percentage of LilyTiny orders include quantities of the board suitable for teaching.

Taken altogether, the sales data seems to support the ongoing impact of the LilyTiny. It is especially notable that the LilyTiny has undergone no major revisions, nor has it been sold as part of a kit or lab pack during its lifetime. While the lack of revisions may be attributable to the simplicity of the hardware and software, it is nonetheless rare to be able to purchase a device maintaining compatibility with any support resources developed in its lifetime. I believe that this stability is crucial for educational adoption.

Customer Reviews and Projects

To complement the analysis of sales data, I wanted to get a sense of customers' actual experiences with the LilyTiny. First, I surveyed all of the LilyTiny product reviews on SparkFun's website, which are submitted by verified customers. I then preliminarily surveyed social media to see what kinds of artifacts individuals are *making* with the LilyTiny. I did this by searching both Twitter and Instagram for public tweets/posts tagged with "#lilytiny".

A first glance reveals that the LilyTiny is being used for a variety of hobbyist projects. A few examples may be seen in Figure 4.8.

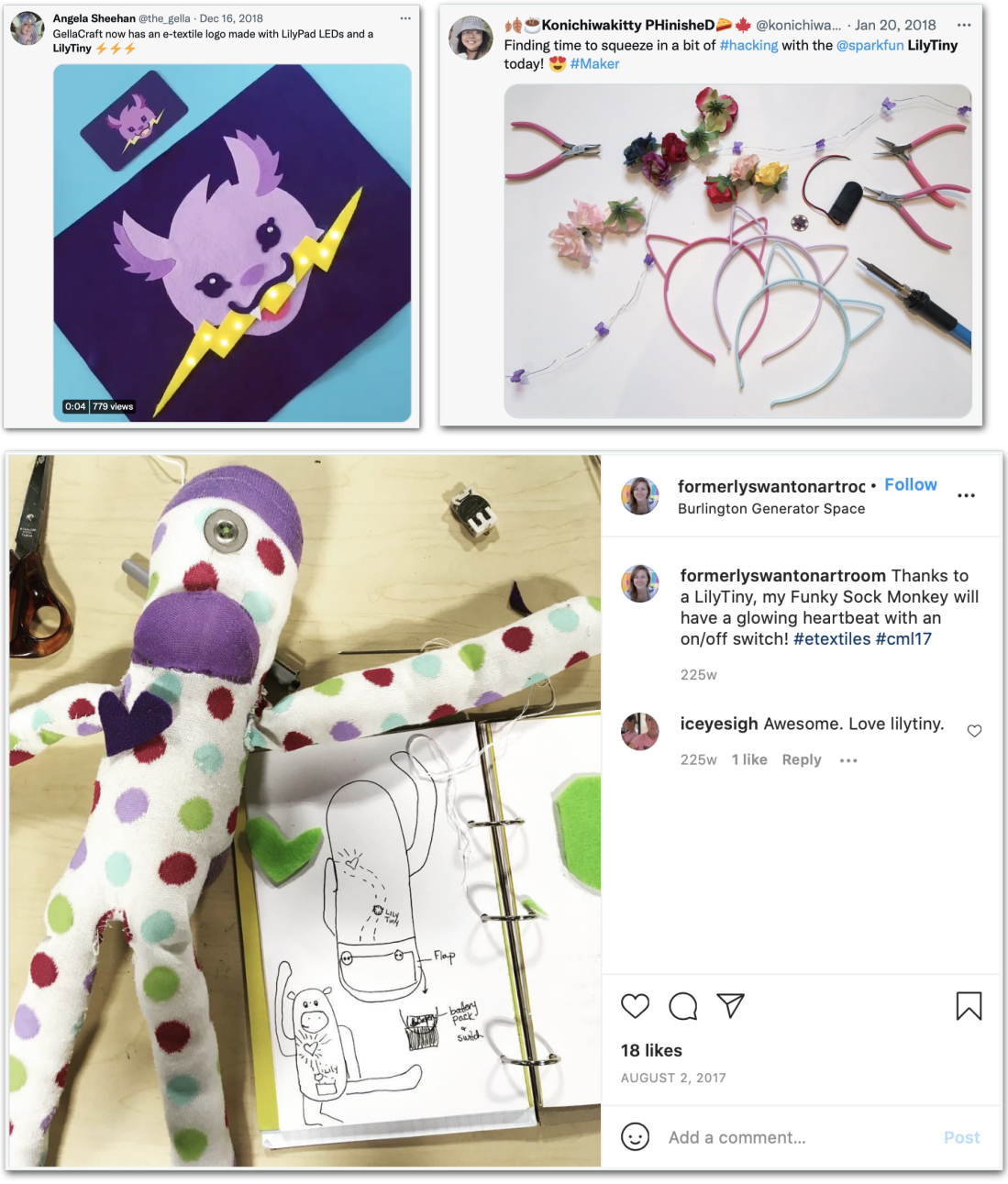

Figure 4.8. Hobbyist projects using the LilyTiny (clockwise from upper left): an e-textile logo, embellished headbands, and a sock monkey with a glowing heart.

These projects are supported by customer reviews which speak to the utility of the LilyTiny for hobbyist projects, both because it is easy-to-use and because it is affordable:

> *"... It is a great board in a small form factor. Very easy to use, works well... I recommend this to anyone - you can't go wrong."*

> *"... Perfect size and power for some of my projects... Highly recommended, especially since they are so inexpensive."*

I also found that the LilyTiny is being used specifically for projects involving handcraft and fine art. A few examples appear in Figure 4.9. This application is also supported by customer reviews like the following:

> *"Just returned from teaching a class for the Southeast Fiber Forum Association... The students were all new to e-textiles... Everyone went away knowing how to finish the stitching at home and a little about circuitry thanks to this great product. All are excited about the possibilities for adding electronics to their fiber art."*

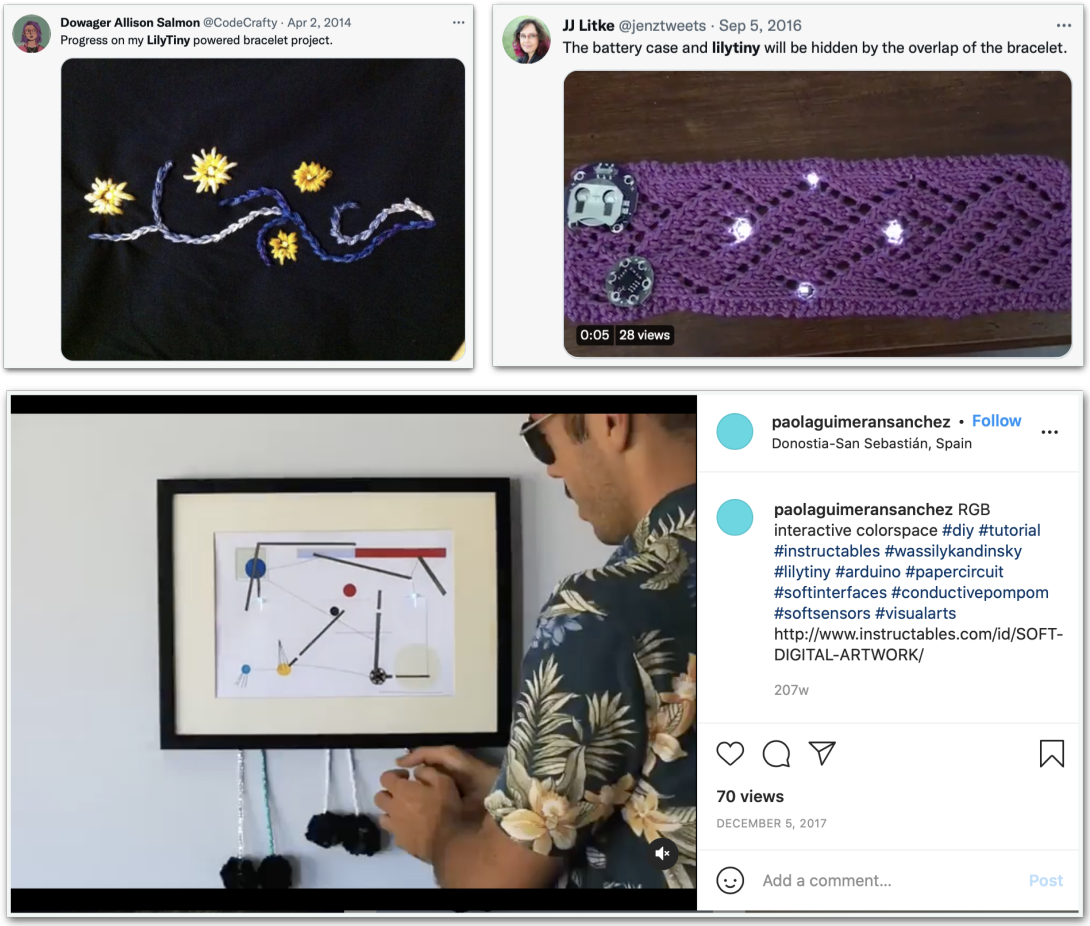

Figure 4.9. Art and craft projects using the LilyTiny (clockwise from upper left): an embroidered bracelet, a knit bracelet, and a mixed media art piece.

I had hoped that the LilyTiny might provide an affordable stepping stone between novice projects and the broader world of Arduino programming, and for some users this does seem to be the case. The following customer reviews speak to the LilyTiny's versatility in this regard:

> *"This is a great little board... I figured out how to reprogram it to do what I needed. It's not too hard... Great price too!"*

"Easy entry point - no regrets! I bought a LilyTiny to power my first project using wearables... The pre-programmed functions took away a layer of complexity and let me just focus on learning how to set up a wearable circuit."

"I've learned the LilyTiny is a great little programmable chip, to me it's a mini-Arduino... It is possibly the smallest form-factor for a Blinky LED circuit. Now I program the Tiny myself..."

Figure 4.10 shows two examples of projects for which the creators have managed to reprogram the LilyTiny.

Figure 4.10. Evidence that some LilyTiny users are choosing to reprogram their boards (left) and are successful in doing so (right).

Lastly, and most importantly to my own goals for the project, there is ample evidence that the LilyTiny is being used for teaching. Some examples of customer reviews to support its value in this arena:

"Just using this as is was a simple project with cool results. I am hoping to use these for a new tinkering club at my school. Fun way to get kids excited without being intimidating."

"We ran an event at our makerspace, to introduce folks to wearable electronics... and this item was exactly what we needed. The price is perfect, the simplicity of it is perfect, and it's a sturdy, well functioning little product. Very pleased and will be ordering hundreds more in the future, I'm sure."

"The Lily Tiny is great for teachers: it is not as cost prohibitive as other microcontrollers and is pretty user friendly for beginners but still allows a programming option to add a challenge."

Figure 4.11 shows an assortment of social media posts showcasing the LilyTiny's use in workshops and classes across a variety of venues.

Although more in-depth research is warranted, I believe that these customer reviews and artifacts affirm that the LilyTiny is helping to expand access to computational textiles. This early evidence suggests that the board offers an affordable entry point for hobbyists, is capable of supporting users in the transition from simple to complex projects, and is reaching my target audience of educators.

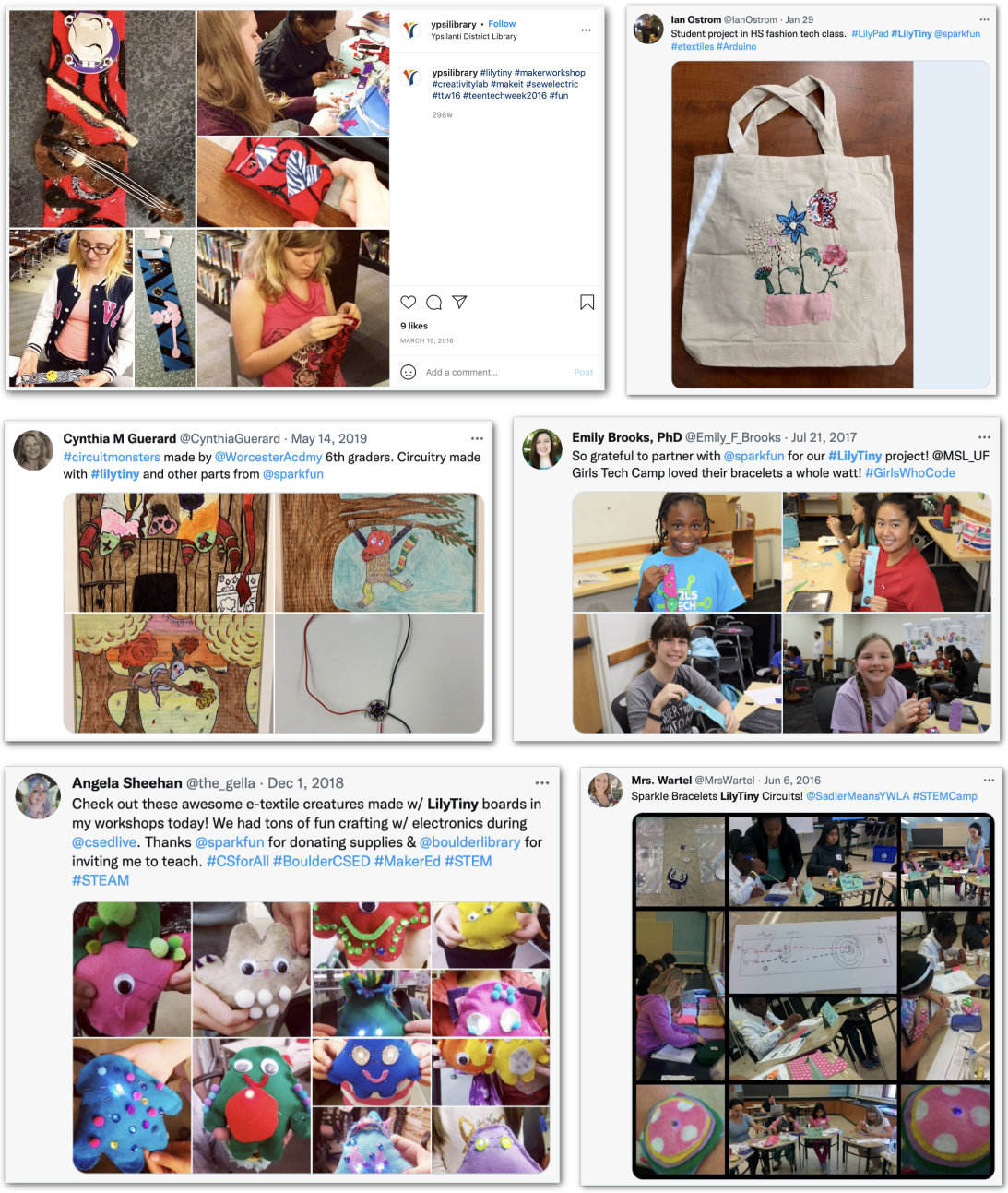

Figure 4.11. Evidence of teaching with the LilyTiny, including offerings at camps, libraries, and K-12 schools.

4.5 Future Work

While this chapter provides an overview of market impact, I plan to continue these investigations to paint a richer picture of how this hardware is being used. A survey of follow-on curriculum and academic research will deepen understanding of educational use at the macro level, complemented by surveys or interviews with educators/facilitators who have used our hardware. I also plan to conduct further analysis of LilyTiny artifacts, to better understand the character of projects enabled by this work.

4.6 Summary

Ten years ago, I set out to develop, pilot, and release a hardware tool and curriculum to support broader educational adoption of e-textile activities. This case study affirms that our hardware has addressed a pressing market need, as evidenced by a variety of follow-on products and several years of sales data. Additionally, exploration of ordering patterns and customer reviews is highly suggestive that the LilyTiny is being used in educational settings.

5 | Conclusion

Looking back now on 15 years of my own research, teaching, and tool/curriculum design, the thread through it all has been a desire to make computing more accessible to those not historically invited into the "clubhouse". To do so requires making computing more *inclusive*; to experiment with new approaches and materials, to celebrate different ways of learning, knowing, and making, and to prod the ever-shifting boundaries between computer science and adjacent fields. As a researcher-practitioner, my work has sought to broaden participation in computing through extensive fieldwork in education, the highlights of which constitute this dissertation.

I have detailed two complementary courses I designed at the margins of collegiate offerings: *Craft of Computing*, which aims to attract a diversity of first- and second-year students to computing, and *Open Source Software Engineering*, which seeks to retain a diversity of upperclassmen through graduation and into computing careers beyond. While more targeted analysis is required to better understand students' pathways beyond these courses, evidence so far suggests that they piqued students' interest in new domains, while positively influencing their confidence, identity, and belonging.

I have also revisited my own prior work in tool/curriculum design for informal learning, conducting follow-on analysis for the LilyTiny sewable microcontroller and accompanying workshop guide. This analysis showed that an inexpensive and stable tool, coupled with freely available instructional resources, can indeed achieve widespread adoption in a market suggestive of novice and educational use – even when challenged by the release of similar and competitor products.

All of these efforts have been driven and shaped by endless conversations with students and educators; I believe the success of my work is a direct testament to the importance of these voices in the design process, along with an iterative approach where continuous feedback is welcomed. I hope that this dissertation helps to affirm the value of interdisciplinary research and teaching towards broadening participation in computing, as the need for this very much persists.

Appendix A | Definitions & Acronyms

C&C: The ACM Conference on Creativity & Cognition.

CS0: Computer Science 0. Common way of referring to a topical computer science course open to non-majors – and often used to attract a diversity of students to computing. Such a course may or may not count towards computer science degree requirements. CS0 courses are often structured either as a survey of the field, combining very introductory programming with an overview of topics like security, ethics, and data science – or as an applied introduction to computing within a specific domain (e.g. robotics, game design, design, etc.).

CS1: Computer Science 1. Common way of referring to the first required course in any computer science department (toward a computer science degree). Typically this is an introductory programming course in a language such as Python or Java.

CS2: Computer Science 2. Common way of referring to the second required course in any computer science department (toward a computer science degree). Typically this is a data structures course.

E-sewing: electronic sewing; the process of sewing with electrically conductive materials (usually to create a *soft circuit*, see below).

E-textiles: electronic textiles; fabric artifacts that contain soft, embedded circuitry.

Educators: not only classroom teachers, but also workshop facilitators and leaders of summer camps or outreach programs.

FIE: The IEEE Frontiers in Education Conference.

FOSS: free and open source software.

GHTC: The IEEE Global Humanitarian Technology Conference.

HFOSS: humanitarian free and open source software.

ICER: The ACM Conference on International Computing Education Research, a single track research conference held annually (held in locations both domestic and abroad).

IDC: The ACM Interaction Design and Children Conference.

LMS: Learning Management System.

MIT: Massachusetts Institute of Technology.

Multimedia: The ACM Annual Conference on Multimedia.

POSSE: Professors' Open Source Software Experience.

SIGCSE: The ACM Special Interest Group on Computer Science Education, also shorthand for this group's annual conference/symposium which gathers computing

education researchers and practitioners from around the world (held in the United States).

Soft circuit: a flexible electrical circuit constructed on the surface of (or embedded in) textiles. Such a circuit may be created using a variety of soft conductive materials (such as conductive threads and fabrics) in conjunction with discrete electronics components (such as lights, batteries, switches, and sensors).

UCSC: The University of California at Santa Cruz.

UIST: The ACM Symposium on User Interface Software and Technology.

Appendix B | Computing Education Seminar Resources

Useful Resources in Computing Education Research

- Mark Guzdial's blog: https://computinged.wordpress.com
- The University of Auckland's *Intro to Computing Education Research* course: https://www.cs.auckland.ac.nz/courses/compsci747s2c/lectures/
- relevant conferences
    - SIGCSE (flagship conference, practice & research) - student volunteer opportunity
    - ITiCSE (working groups)
    - ICER (resarch focused) - doctoral consortium opportunity
    - Koli Calling (intimate discussion)
    - ICREE International Conference on Research on Engineering Education
- relevant journals/places to publish in *print*
    - *ACM Inroads* (2010 to present) quarterly magazine (http://inroads.acm.org/)
    - SIGCSE link for places to publish (http://sigcse.org/resources/publish)
    - ACM Transactions on Computing Education (TOCE) (http://toce.acm.org/)
    - Journal of Engineering Research JEE
    - Journal of Research on Technology in Education
    - International Journal of Research & Methods in Education
    - Empirical Software Engineering
    - Journal of Educational Data Minin
    - Software Engineering Education & Training
    - Computers & Education
    - Journal of Women and Minorities in Science and Engineering
    - American Educational Research Association Conference
    - Journal of Education Computing Research
    - Communications of the ACM
    - Journal of Computer Science Education Online
    - Computing Research News
    - CSTA Voice
- methods texts
    - Creswell, J. W. (2013). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications.
    - Luker, K. (2008). *Salsa dancing into the social sciences: Research in an age of info-glut*. Harvard University Press.
    - Patton, M. Q. (1990). *Qualitative evaluation and research methods* . SAGE Publications, inc.
    - Pring, R. (2004). *The Philosophy of Educational Research*. Bloomsbury Publishing.
    - Weiss, R. S. (1995). *Learning from strangers: The art and method of qualitative interview studies*. Simon and Schuster.
- relevant classes at UCSC
    - EDUC 235: Introduction to Educational Inquiry
    - EDUC 236: Quantitative Methods in Educational Research
    - EDUC 237: Qualitative Research Methods
    - PSYCH 204: Quantitative Data Analysis
    - PSYCH 214A: Multivariate Techniques
    - PSYCH 214B: Advanced Multivariate Techniques
- SOE grad student mailing list (for computing education research)
    - Emily can add you!
    - usually, we meet informally once a month for lunch/coffee to check in - although on hiatus this quarter due to the seminar
- Center for Statistical Analysis in the Social Sciences (CSASS)
    - http://csass.ucsc.edu/
    - Free experiment design and psychometric consulting available to faculty and students

- Basic Experiment Research Paper Rubric
    - https://drive.google.com/file/d/0Bzp0dKE9hnvBLWx2MVFTX28yUFk/view?usp=sharing

Duties for Weekly Discussion Leader

- you do *not* need to be an expert on the topics or readings
- if you'd like to focus on readings other than the ones that Emily prioritized for your week, touch base with her and update the TitanPad at least 1 week in advance of your discussion date
- come to class with some questions for group discussion
- take a lead in faciltating the group discussion
- feel free to be creative! you can ask the group to do a hands-on activity (either to prepare - with enough notice - or in class), you can share background info on the author(s), etc.

Week 1 (9/25): Research Methods, Conferences, Context, etc.

- Borrego, M., Douglas, E. P., & Amelink, C. T. (2009). Quantitative, qualitative, and mixed research methods in engineering education. *Journal of Engineering Education*, *98*(1), 53-66.
    - https://www.cs.auckland.ac.nz/courses/compsci747s2c/lectures/borrego.ea-quantitative-jee-09.pdf
- Case, J. M., & Light, G. (2011). Emerging research methodologies in engineering education research. *Journal of Engineering Education*, *100*(1), 186-210.
    - https://www.cs.auckland.ac.nz/courses/compsci747s2c/lectures/case.light-emerging-jee-11.pdf

Week 2 (10/2): Theoretical Background - Motivation and Development

- **Required Reading**
    - Csikszentmihalyi, M. (1991). *Flow: The psychology of optimal experience* (Vol. 41). New York: HarperPerennial.
        - Chapter 4: The Conditions of Flow (p. 71-93): https://llk.media.mit.edu/courses/readings/Csikszentmihalyi-Flow-Ch4.pdf
    - Ryan, R. M., & Deci, E. L. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American psychologist*, *55*(1), 68.
        - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.335.6945&rep=rep1&type=pdf
    - Wikipedia page for *Zone of Proximal Development*:
        - https://en.wikipedia.org/wiki/Zone_of_proximal_development
    - Wikipedia page for *Maslow's Hierarchy of Needs*:
        - https://en.wikipedia.org/wiki/Maslow%27s_hierarchy_of_needs
- **Stretch Goals!**
    - Maslow, A. H. (1943). A theory of human motivation. *Psychological review*, *50*(4), 370.
        - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.318.2317&rep=rep1&type=pdf
    - Vygotsky, L. (1978). Interaction between learning and development. *Mind and society: The development of higher psychological processes*.
        - http://www.psy.cmu.edu/~siegler/vygotsky78.pdf
- **In Class Resources**
    - Mark Guzdial's blog post on integration of research into teaching practice
        - https://computinged.wordpress.com/2015/09/21/a-terrific-and-dismal-view-of-what-influences-cs-faculty-to-adopt-teaching-practices/
    - paper on belonging (Emily will add)
    - paper on peer leaders in the classroom (Emily will add)

- The Pause Procedure (Charlie)
    - http://www.tc.umn.edu/~bunte002/resources/Ruhl_1987_.pdf
- student critics of teaching (Dylan)
    - http://crookedtimber.org/2013/05/28/employing-a-student-to-criticize-my-teaching/
- learning styles debunked (Charlie)
    - http://www.psychologicalscience.org/index.php/news/releases/learning-styles-debunked-there-is-no-evidence-supporting-auditory-and-visual-learning-psychologists-say.html

Week 3 (10/9): Theoretical Background Continued - Self Efficacy, Growth Mindset, and Grit

- **Required Reading**
    - Wikipedia page for "Self-Efficacy" (brief initial summary only)
        - brief initial summary only: https://en.wikipedia.org/wiki/Self-efficacy
    - Dweck, C. S. (2000). *Self-theories: Their role in motivation, personality, and development*. Psychology Press.
        - Chapter 1 (p. 1-4): https://llk.media.mit.edu/courses/readings/Dweck.pdf
    - Duckworth, A. L., Peterson, C., Matthews, M. D., & Kelly, D. R. (2007). Grit: perseverance and passion for long-term goals. *Journal of personality and social psychology*, *92*(6), 1087.
        - http://rrhs.schoolwires.net/cms/lib7/WI01001304/Centricity/Domain/187/Grit%20JPSP.pdf
- **Stretch Goals!**
    - Bandura, A. (1977). Self-efficacy: toward a unifying theory of behavioral change. *Psychological review*, *84*(2), 191.
        - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.315.4567&rep=rep1&type=pdf
    - Pajares, F. (1996). Self-efficacy beliefs in academic settings. *Review of educational research*, *66*(4), 543-578.
        - http://files.eric.ed.gov/fulltext/ED384608.pdf
    - Dweck, C. S. (2000). *Self-theories: Their role in motivation, personality, and development*. Psychology Press.
        - Chapters 2, 3 (p. 5-19): https://llk.media.mit.edu/courses/readings/Dweck.pdf

Week 4 (10/16): Constructivism, Constructionism, and Project-Based Learning

- **Required Reading**
    - Ackermann, E. (2001). Piaget's constructivism, Papert's constructionism: What's the difference. *Future of learning group publication*, *5*(3), 438.
        - http://www.sylviastipich.com/wp-content/uploads/2015/04/Coursera-Piaget-_-Papert.pdf
    - Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc..
        - Chapter 1: Computers and Computer Culture
        - Full book here: http://eclass.uoa.gr/modules/document/file.php/PPP240/Βιβλία/MNDSTORMS%20Children,%20Computers,%20and%20Powerful%20Ideas.%20Papert.pdf
- **Stretch Goals!**
    - Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc..
        - Preface: The Gears of My Childhood
        - Introduction: Computers for Children

        - Full book here: http://eclass.uoa.gr/modules/document/file.php/PPP240/Βιβλία/MNDSTORMS%20Children,%20Computers,%20and%20Powerful%20Ideas.%20Papert.pdf
    - Eisenberg, M. (2003). Mindstuff educational technology beyond the computer. *Convergence: The International Journal of Research into New Media Technologies*, *9*(2), 29-53.
        - http://l3d.cs.colorado.edu/~ctg/pubs/Mindstuff.pdf
    - Turkle, S., & Papert, S. (1992). Epistemological pluralism and the revaluation of the concrete. *Journal of Mathematical Behavior*, *11*(1), 3-33.
        - http://kvantti.kapsi.fi/Documents/Turkle%20Papert%20-%20Epistemological%20Pluralism%20and%20the%20Revaluation%20of%20the%20Concrete%20-%201992.pdf
    - Krajcik, J. S., & Blumenfeld, P. C. (2006). *Project-based learning* (pp. 317-334). na.
        - http://pisga.lms.education.gov.il/pluginfile.php/121596/mod_resource/content/1/CHAPTER%2019%20PBL%20Kraichik.docx
- **In Class Resources**
    - Authenticity - Martin Heidegger's vision of becoming more authentic
        - http://www.tc.umn.edu/~parkx032/XP226.html
    - Coursera: Learning How to Learn
        - https://www.coursera.org/learn/learning-how-to-learn
    - Oakley, Barbara. "A Mind for Numbers"
        - http://www.amazon.com/Mind-Numbers-Science-Flunked-Algebra/dp/039916524X/ref=asap_bc?ie=UTF8
    - CodeSpells (programming by writing "spells")
        - http://codespells.org
    - Glitch (programming through debugging games)
        - http://betsydisalvo.com/projects/glitch-game-testers/
    - Blog (Norris/Soloway) on Sept. 2015 OECD (Organisation for Economic Cooperation and Development) report on how computers in the classroom HURT learning
        - https://thejournal.com/articles/2015/09/21/oecd-report.aspx

Week 5 (10/23): Programming Languages for Teaching

- **Required Reading**
    - McIver, L., & Conway, D. (1996, January). Seven deadly sins of introductory programming language design. In *Software Engineering: Education and Practice, 1996. Proceedings. International Conference* (pp. 309-316). IEEE.
        - http://www.csse.monash.edu.au/~damian/papers/PDF/SevenDeadlySins.pdf
    - Mannila, L., & de Raadt, M. (2006, February). An objective comparison of languages for teaching introductory programming. In *Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006* (pp. 32-37). ACM.
        - http://eprints.usq.edu.au/1701/2/research2.pdf
    - Guzdial, M. (2013, August). Exploring hypotheses about media computation. In *Proceedings of the ninth annual international ACM conference on International computing education research* (pp. 19-26). ACM.
        - https://users.soe.ucsc.edu/~emme/other/p19-guzdial.pdf
        - http://dl.acm.org/citation.cfm?id=2493397 try this if Emily's link doesn't work for you (sorry, there was a typo! first link should be working now)
- **Stretch Goals!**
    - Henriksen, P., & Kölling, M. (2004, October). Greenfoot: combining object visualisation with interaction. In *Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications* (pp. 73-82). ACM.
        - http://www.researchgate.net/profile/Michael_Koelling/publication/221321474_greenfoot_combining_object_visualisation_with_interaction/links/0c9605244913f84d8

8000000.pdf

- Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, *10*(4), 16.
  - http://web.media.mit.edu/~jmaloney/papers/ScratchLangAndEnvironment.pdf
- Cooper, S., Dann, W., & Pausch, R. (2000, April). Alice: a 3-D tool for introductory programming concepts. In *Journal of Computing Sciences in Colleges* (Vol. 15, No. 5, pp. 107-116). Consortium for Computing Sciences in Colleges.
  - http://web.stanford.edu/~coopers/alice/ccscne00.PDF
- Reas, C., & Fry, B. (2006). Processing: programming for the media arts. *AI & SOCIETY*, *20*(4), 526-538.
  - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.207.7452&rep=rep1&type=pdf
- Hadjerrouit, S. (1998). Java as first programming language: a critical evaluation. *ACM SIGCSE Bulletin*, *30*(2), 43-47.
- maybe include a hands-on exercise where we all try Python & Scratch (Snap? Processing.org? Guzdial "Media Computation" Language vs Tool/Environment)?

- **In Class Resources**
  - lots of websites where you can test drive a language quickly
    - tryruby.org
    - trypython.org
  - Philip Guo's survey showing Python at the top
    - http://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-us-universities/fulltext
  - Logical Journey (computer game from the 1990s)
    - http://www.amazon.com/Zoombinis-Logical-Journey-PC-Mac/dp/B00005LBVU
    - https://itunes.apple.com/us/app/zoombinis/id961739806?mt=8
  - LaPlaya - UCSB's more constrained take on Scratch
    - https://octopi.herokuapp.com
  - Snap! - Berkeley's extended take on Scratch
    - http://snap.berkeley.edu
  - programming Arduino with a Scratch-like environment
    - S4A: http://s4a.cat
    - Modkit: http://www.modkit.com
  - Karel the Robot - turtle-like programming
    - https://www.cs.mtsu.edu/~untch/karel/
  - Bee-Bot - physical turtle robot
    - https://www.bee-bot.us
  - NAND to Tetris course/book
    - http://www.nand2tetris.org

Week 6 (10/30) : Hot Topics - Flipped Classrooms, Peer Instruction, and Active Learning

- **Required Reading**
  - NYT Article: Are College Lectures Unfair?
    - http://www.nytimes.com/2015/09/13/opinion/sunday/are-college-lectures-unfair.html?_r=0
  - Bishop, J. L., & Verleger, M. A. (2013, June). The flipped classroom: A survey of the research. In *ASEE National Conference Proceedings, Atlanta, GA*.
    - http://www.studiesuccesho.nl/wp-content/uploads/2014/04/flipped-classroom-artikel.pdf
  - The attention curve: http://cain.blogspot.com/2012/10/podcasting-and-attention-curve.html

        - Ruhl et al, Using the Pause Procedure to Enhance Lecture Recall, http://www.tc.umn.edu/~bunte002/resources/Ruhl_1987_.pdf
        - Porter, L., Bailey Lee, C., & Simon, B. (2013, March). Halving fail rates using peer instruction: a study of four computer science courses. In *Proceeding of the 44th ACM technical symposium on Computer science education* (pp. 177-182). ACM.
            - https://users.soe.ucsc.edu/~emme/other/p177-porter.pdf
- **Stretch Goals!**
    - Peer Instruction/Collaborative Learning
        - Crouch and Mazur, Peer Instruction: Ten years of experience and results, http://web.mit.edu/jbelcher/www/TEALref/Crouch_Mazur.pdf
        - Buffum, Philip Sheridan, et al. "Leveraging collaboration to improve gender equity in a game-based learning environment for middle school computer science." *Research in Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT), 2015*. IEEE, 2015. http://ieeexplore.ieee.org.oca.ucsc.edu/xpls/abs_all.jsp?arnumber=7296496
        - Linda Werner, Jill Denner, Shannon Campe, Eloy Ortiz, Dawn DeLay, Amy C. Hartl, and Brett Laursen. 2013. Pair programming for middle school students: does friendship influence academic outcomes?. In *Proceeding of the 44th ACM technical symposium on Computer science education*(SIGCSE '13). ACM, New York, NY, USA, 421-426. DOI=http://dx.doi.org/10.1145/2445196.2445322
        - Porter, L., Bailey Lee, C., Simon, B., & Zingaro, D. (2011, August). Peer instruction: do students really learn from peer discussion in computing?. In *Proceedings of the seventh international workshop on Computing education research* (pp. 45-52). ACM.
        - Simon, B., Esper, S., Porter, L., & Cutts, Q. (2013, August). Student experience in a student-centered peer instruction classroom. In *Proceedings of the ninth annual international ACM conference on International computing education research* (pp. 129-136). ACM.
- **In Class Resources**
    - recent news story on meditation in San Francisco schools
        - http://www.inhabitots.com/schools-in-san-francisco-implement-meditation-time-students-happiness-and-academic-success-soars/

Week 7 (11/6): Challenges & Broadening Participation

- Speaker: Jill Denner, Ph.D. is a senior research scientist at ETR. She does applied research with a focus on increasing the number of women, girls and Latino/a students in computing. Dr. Denner also has led the development of several after-school programs designed to increase children's opportunities to become producers, not just users, of technology. She is nationally recognized as an expert in strategies to engage girls/women and Latino/a students in computer science, in both K–12 and community college, and regularly does peer review of journal articles as well as grant proposals for the National Science Foundation.
- **Required Reading**
    - Denner, J., Bean, S., & Martinez, J., The Girl Game Company: Engaging Latina Girls in Information Technology. *National Institute on Out-of-School Time, "Afterschool Matters Spring 2009" (2009). Afterschool Matters. Book 21, pp 26-35.*
        - http://repository.wellesley.edu/cgi/viewcontent.cgi?article=1016&context=afterschoolmatters
    - Margolis, Goode, Binning (2015). Expanding the Pipeline -Exploring Co mputer Science: Active Learning for Broadening Participation in Computing. *Computing Research News*. Oct. 2015, Vol. 27/No. 9.
        - http://cra.org/crn/2015/10/expanding-the-pipeline-exploring-computer-science-active-learning-for-broadening-participation-in-computing/
    - Margolis, J., Estrella, R., Goode, J., Holme, J.J., & Nao, K., (2008). Stuck in the Shallow End: Education, Race, and Computing. MIT Press. Read the Conclusion - "The Best and

the Brightest"? The book is a quick read.
    - http://ieeexplore.ieee.org.oca.ucsc.edu/xpl/ebooks/bookPdfWithBanner.jsp?fileName=6283349.pdf&bkn=6267411&pdfType=chapter conclusion
    - http://ieeexplore.ieee.org.oca.ucsc.edu/xpl/bkabstractplus.jsp?bkn=6267411 table of contents
    - http://ieeexplore.ieee.org.oca.ucsc.edu/xpl/ebooks/bookPdfWithBanner.jsp?fileName=6283348.pdf&bkn=6267411&pdfType=chapter introduction
- **Stretch Goals!**
    - Zweben, S., & Bizot, B. (2015). 2014 Taulbee Survey. *COMPUTING*, *27*(5).
        - http://archive2.cra.org/uploads/documents/resources/crndocs/2014-Taulbee-Survey.pdf
    - Pinkard, N., Barron, B., & Martin, C. (2008, June). Digital youth network: fusing school and after-school contexts to develop youth's new media literacies. In *Proceedings of the 8th international conference on International conference for the learning sciences-Volume 3* (pp. 113-114). International Society of the Learning Sciences.
    - Rusk, N., Resnick, M., & Cooke, S. (2009). Origins and guiding principles of the computer clubhouse. *The computer clubhouse: Constructionism and creativity in youth communities*, 17-25.
        - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.141.5830&rep=rep1&type=pdf
    - Margolis, J., & Fisher, A. (2003). *Unlocking the clubhouse: Women in computing*. MIT press.
    - ***READING ON BELONGING? MENTORSHIP?***
    - ***ASK BRAD FOR A PAPER SUGGESTION FOR COGNITIVE DISABILITIES***
- **In Class Resources**
    - Max/MSP (multimedia programming with visual modules and "wires")
        - https://en.wikipedia.org/wiki/Max_(software)
        - https://cycling74.com/products/max/
        - http://www.instructables.com/id/Intro-to-MaxMSP/
    - Pure Data - similar to Max/MSP but open source!
        - https://puredata.info
    - music-meets-computer science/programming class by someone in Boston? (mentioned by Jill)

Week 8 (11/13): Growth Mindset & Self-Efficacy in the Classroom

- **Required Reading**
    - Hutchison, M. A., Follman, D. K., Sumpter, M., & Bodner, G. M. (2006). Factors influencing the self-efficacy beliefs of first-year engineering students. *JOURNAL OF ENGINEERING EDUCATION-WASHINGTON-*, *95*(1), 39.
        - https://users.soe.ucsc.edu/~emme/other/p39-hutchison.pdf
    - McDowell, C; Werner, L; Bullock, H E; & Fernald, J. (2006). Pair programming improves student retention, confidence, and program quality. *Communications of the ACM*, 49(8), 90 - 95. doi: 10.1145/1145287.1145293.
        - http://escholarship.org/uc/item/1s49s13f
    - Cutts, Q., Cutts, E., Draper, S., O'Donnell, P., & Saffrey, P. (2010, March). Manipulating mindset to positively influence introductory programming performance. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 431-435). ACM.
        - http://ims.mii.lt/ims/konferenciju_medziaga/SIGCSE'10/docs/p431.pdf
- **Stretch Goals!**
    - Murphy, L., & Thomas, L. (2008). Dangers of a fixed mindset: implications of self-theories research for computer science education. *ACM SIGCSE Bulletin*, *40*(3), 271-275.
        - http://www.researchgate.net/profile/Lynda_Thomas4/publication/220808023_Dangers_of_a_fixed_mindset_implications_of_self-

theories_research_for_computer_science_education/links/02bfe5137acd1ccea0000000.pdf

- Dunlap, J. C. (2005). Problem-based learning and self-efficacy: How a capstone course prepares students for a profession. *Educational Technology Research and Development*, *53*(1), 65-83.
- Zeldin, A. L., & Pajares, F. (2000). Against the odds: Self-efficacy beliefs of women in mathematical, scientific, and technological careers. *American Educational Research Journal*, *37*(1), 215-246.
- Zeldin, A. L., Britner, S. L., & Pajares, F. (2008). A comparative study of the self-efficacy beliefs of successful men and women in mathematics, science, and technology careers. *Journal of Research in Science Teaching*, *45*(9), 1036-1058.
    - https://users.soe.ucsc.edu/~emme/other/p1036-zeldin.pdf
- Quintin Cutt's crib sheet - http://www.dcs.gla.ac.uk/~quintin/cribsheet.doc
- paper that discussed pair programming pairings:
    - Bevan, J., Werner, L., & McDowell, C. (2002). Guidelines for the use of pair programming in a freshman programming class. In *Software Engineering Education and Training, 2002.(CSEE&T 2002). Proceedings. 15th Conference on* (pp. 100-107). IEEE.
    - http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=995202

Week 9 (11/20) : Computing *for* Education - MOOCs, Intelligent Tutoring, Games, Crowdgrader

- **Required Reading**
    - Koedinger, K. R., Anderson, J. R., Hadley, W. H., & Mark, M. A. (1997). Intelligent tutoring goes to school in the big city.
        - http://repository.cmu.edu/cgi/viewcontent.cgi?article=1000&context=hcii
    - DiSalvo, B., & Bruckman, A. (2011). From interests to values. *Communications of the ACM*, *54*(8), 27-29.
        - http://www.researchgate.net/profile/Betsy_Disalvo/publication/220423452_From_Interests_to_Values/links/0046353a98acf9159b000000.pdf
    - de Alfaro, L., & Shavlovsky, M. (2014, March). CrowdGrader: A tool for crowdsourcing the evaluation of homework assignments. In *Proceedings of the 45th ACM technical symposium on Computer science education* (pp. 415-420). ACM.
        - http://arxiv.org/pdf/1308.5273.pdf
    - Compeau, P., & Pevzner, P. A. (2015). Life after MOOCs. *Communications of the ACM*, *58*(10), 41-44.
        - http://cacm.acm.org/magazines/2015/10/192385-life-after-moocs/fulltext
- **Stretch Goals!**
    - Koedinger, K. R., & Corbett, A. (2006). *Cognitive tutors: Technology bringing learning sciences to the classroom*. na.
        - http://isites.harvard.edu/fs/docs/icb.topic603902.files/KoedingerCorbett05.pdf
    - Lee, M. J., & Ko, A. J. (2011, August). Personifying programming tool feedback improves novice programmers' learning. In *Proceedings of the seventh international workshop on Computing education research* (pp. 109-116). ACM.
        - http://www.pixel42.com/cv/publications/Lee2011_GidgetPersonification.pdf

Week 10 (12/4): Open Source Hardware & Software in the Classroom (Fabrication, too!)

- **Required Reading**
    - Eisenberg, M. (2007, March). Pervasive fabrication: Making construction ubiquitous in education. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07. Fifth Annual IEEE International Conference on* (pp. 193-198). IEEE.

        - http://ojs.academypublisher.com/index.php/jsw/article/download/03046268/975
    - Buechley, L., & Hill, B. M. (2010, August). LilyPad in the wild: how hardware's long tail is supporting new engineering and design communities. In *Proceedings of the 8th ACM Conference on Designing Interactive Systems* (pp. 199-207). ACM.
        - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.207.8075&rep=rep1&type=pdf
    - Hislop, G. W., Ellis, H. J., Pulimood, S. M., Morgan, B., Mello-Stark, S., Coleman, B., & Macdonell, C. (2015, July). A Multi-Institutional Study of Learning via Student Involvement in Humanitarian Free and Open Source Software Projects. In *Proceedings of the eleventh annual International Conference on International Computing Education Research* (pp. 199-206). ACM.
        - https://users.soe.ucsc.edu/~emme/other/p199-hislop.pdf
- **Stretch Goals!**
    - Lakhani, K., & Wolf, R. G. (2003). Why hackers do what they do: Understanding motivation and effort in free/open source software projects.
        - http://flosshub.org/system/files/lakhaniwolf.pdf
    - Qiu, K., Buechley, L., Baafi, E., & Dubow, W. (2013, June). A curriculum for teaching computer science through computational textiles. In *Proceedings of the 12th International Conference on Interaction Design and Children* (pp. 20-27). ACM.
        - http://www.researchgate.net/profile/Wendy_Dubow/publication/262206765_A_curriculum_for_teaching_computer_science_through_computational_textiles/links/550b49a60cf2855640970616.pdf

Miscellaneous readings of interest that don't fit anywhere in particular...

- Winograd, T., & Flores, F. (1986). *Understanding computers and cognition: A new foundation for design*. Intellect Books.
- Karat, J., & Dayton, T. (1995, May). Practical education for improving software usability. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 162-169). ACM Press/Addison-Wesley Publishing Co..
- Strike, K. A., & Posner, G. J. (1985). A conceptual change view of learning and understanding. *Cognitive structure and conceptual change*, *211*, 231.
- Raymond, E. S. (2001). *The Cathedral & the Bazaar: Musings on linux and open source by an accidental revolutionary*. " O'Reilly Media, Inc.".
- ***check ICER 2015? (something about blocks-based languages...)***
- Pair Programming at UCSC
    - McDowell, C., Werner, L., Bullock, H., & Fernald, J. (2002, February). The effects of pair-programming on performance in an introductory programming course. In *ACM SIGCSE Bulletin* (Vol. 34, No. 1, pp. 38-42). ACM. http://p8888-ucelinks.cdlib.org.oca.ucsc.edu/sfx_local?sid=google&auinit=C&aulast=McDowell&atitle=The+effects+of+pair-programming+on+performance+in+an+introductory+programming+course&id=doi:10.1145/563340.563353&title=SIGCSE+bulletin&volume=34&issue=1&date=2002&spage=38&issn=1096-3936
    - Werner, L. L., Hanks, B., & McDowell, C. (2004). Pair-programming helps female computer science students. *Journal on Educational Resources in Computing (JERIC)*, *4*(1), 4. http://p8888-ucelinks.cdlib.org.oca.ucsc.edu/sfx_local?sid=google&auinit=LL&aulast=Werner&atitle=Pair-programming+helps+female+computer+science+students&id=doi:10.1145/1060071.1060075&title=Journal+on+educational+resources+in+computing&volume=4&issue=1&date=2004&spage=4&issn=1531-4278
    - McDowell, C; Werner, L; Bullock, H E; & Fernald, J. (2006). Pair programming improves student retention, confidence, and program quality. *Communications of the ACM*, 49(8), 90 - 95. doi: 10.1145/1145287.1145293. UC Santa Cruz: Retrieved from:

http://escholarship.org/uc/item/1s49s13f
    - Werner, L. L., Hanks, B., McDowell, C., Bullock, H., & Fernald, J. (2005). Want to increase retention of your female students. *Computing Research News*,*17*(2), 2. https://users.soe.ucsc.edu/~charlie/projects/pairprogramming/Pipeline.pdf
- CSTA K-12 Computer Science Standards http://www.csta.acm.org/Curriculum/sub/K12Standards.html
- Grover, S., & Pea, R. (2013). Computational Thinking in K–12 A Review of the State of the Field. *Educational Researcher*, *42*(1), 38-43.
    - http://edr.sagepub.com/content/42/1/38.full.pdf+html

```
count = 0
def incCount():
  counts = count + 1 #whoops should be count
incCount()
print count #expected 1 got 0
count = "cow"
print count # wait I thought count was an int
```

Appendix C | Berea College Course Syllabi

Craft of Computing Syllabus
Fall 2018

Instructor: Emily Lovell
Instructor e-mail: emily_lovell@berea.edu
Office hours: Mon 2:30pm-4:00pm
Wed 3:30pm-5:00pm
Office: Danforth Tech 102B
Phone: (859) 985-3930

Course Number: CSC 110
Class time: Tue/Thu 1:00pm-2:50pm
Classroom: Danforth Tech 104
Lab times: Sun-Thu 7:00pm-9:00pm

Note: This syllabus is subject to changes and updates throughout the term.

Catalog Description

An introduction to the foundations of computer science within a craft context. Students will gain experience with a text-based programming language while exploring themes of computational design. Throughout the semester, we will also experiment with artisanal craft techniques and realize our designs using tools such as a laser cutters or vinyl cutter. No previous crafting or computer programming experience is required.

In Other Words…

We'll start by getting our feet wet with the Processing programming language, which was originally developed as a tool for artists and designers. While learning computer science fundamentals, we'll also experiment with traditional craft techniques such as felting, sewing, and embroidery.

To this end, we will spend roughly half of our class time in the technical world. For example, we'll spend time learning how to think like a computer scientist and to program in Processing.

We'll spend the rest of our time in the realm of craft/design. This may entail learning a new craft technique or design tool, having a class discussion based on a reading or video, and/or critiquing one another's work.

As the term progresses, we'll combine these two approaches to create designs/patterns that we can realize in physical craft form.

Some examples of artifacts you might make in this course are:

- Lasercut stationery/greeting cards, iron-on patches, wooden coasters, or felt trivets
- Vinylcut stickers

- Pen-plotted drawings or paintings

You will be actively encouraged to propose your own ideas as well!

Practical Reasoning Requirement

This course satisfies Berea College's Practical Reasoning requirement. At completion of the course, a successful student will be:

1. more sensitive to nuances of language use;
2. able to identify logical relationships between and among propositions;
3. able to use and to recognize different patterns of reasoning;
4. able to recognize improper patterns of reasoning;
5. able to use appropriate criteria to evaluate reasoning;
6. able to deliberate about various courses of action by weighing evidence;
7. able to think clearly about values and their place in a reflective and active life.

Required Text & Reading

We will be using Slack as our primary means to communicate outside of class. This is where I will post announcements, additional information regarding assignments, etc. It is also a place where you can ask one another for help or advice. For this reason, I consider our course's Slack channel to be required reading. Once you've joined our team and channel, you can set up notifications that go to your phone and/or e-mail.

The textbook that we'll using is more of a workbook than a traditional textbook. It also has a really great companion website, which you can find at: http://learningprocessing.com

Learning Processing:
A Beginner's Guide to Programming Images, Animation, and Interaction (Second Edition)
by Daniel Shiffman

You should bring your copy of the textbook with you to every class meeting. You should also bring your computer with you to every class.

Grade Distribution

Assignments: 20%

Assignments may include (but are not limited to): written reading responses, programming exercises, and outside-of-class research.

Quizzes: 25%

A handful of quizzes will occur throughout the term, with time to prepare and information on what they will cover. These will mostly cover technical content and will allow me to better understand which topics would benefit from more coverage.

Mini-projects: 30%

Mini-projects will offer the opportunity to bring your code to life. Through these special assignments, you will learn to interface between the digital and physical worlds — for example, by making something on a laser cutter which you first designed on your computer.

Final project: 25%

The final project will allow you to blend craft and computation in a more open-ended context than the mini-projects. Once again, you will bring a design of your own into the physical world — but you will do so through a medium and technique of your choice.

Late Work Policy

Because we will be building upon prior coursework throughout the term, it's extremely important that you complete assignments on time. However, I will accept **one** assignment late without penalty. All other late assignments will be reduced by a flat penalty of 20%. **No submissions will be accepted beyond three days late.** No extensions will be granted on quizzes, in-class presentations, or the final project. Exceptions will be considered only in case of severe illness, hospitalization, etc. which will be handled on a case-by-case basis.

Grading Scale

100%	≥	A	≥	93%	>	A–	≥	90%			
90%	>	B+	≥	87%	>	B	≥	83%	>	B–	≥	80%
80%	>	C+	≥	77%	>	C	≥	73%	>	C–	≥	70%
70%	>	D+	≥	67%	>	D	≥	63%	>	F	≥	0

Attendance Policy

We will be actively engaging during class time in small group exercises, pair programming, presentations, and projects. Your attendance is critical in supporting these activities and for this reason my attendance policy is as follows:

- I'll take attendance at each class, once at the beginning.
- If you arrive late, it's your responsibility to make sure I've included your name on the attendance list. It's my prerogative to count late arrivals as ½ absence. (The same applies if you leave early or do not participate in class.)
- After 3 absences, your course grade will be reduced by ⅓ of a letter grade per absence. There is no distinction between excused and unexcused absences (except in very unusual cases of emergency such as family death or severe illness, which you must discuss with me and which must be documented).
- More than 6 absences (or roughly 20% of the course) automatically results in a failing grade for the course.

Quizzes will be held during class periods. Make-up quizzes must be arranged in advance and will be granted at my discretion. Any quiz that is missed without appropriate communication will earn a grade of zero.

If you miss class for any reason, you are responsible for getting yourself caught up. This means taking it upon yourself to talk to classmates about missed material — after which you should follow up with me if you have outstanding questions.

Delays

In rare instances, I may be delayed arriving to class. If I have not arrived by the time class is scheduled to start, you must wait a minimum of 20 minutes for my arrival. In the event that I will miss class entirely, this will be communicated on Slack along with any additional information regarding assignments.

Academic Honesty

This course adheres fully to the college's policies regarding academic honesty/dishonesty.

In the preparation and presentation of any assigned work, all students shall conform to a strict standard of academic honesty. Any attempt to pass-off the ideas or work of others as your original work or to help another student to do so will be considered a violation of this standard. Thus, you must formally acknowledge in any work you submit the words and/or ideas of others taken from other students or from any print source or any electronic media, whether a direct quotation such as a cut-and paste or a paraphrase. Any omission of this standard, however minor, is dishonest and is called plagiarism. In the real-word, plagiarism is considered theft. In the workplace, such theft can lead to lawsuits which cost the company time, money, and prestige. In this course, you must clearly document in everything you submit what is your own

original work and what is the work of others. Academic dishonesty also includes presenting fabricated data as authentic.

At the first instance of plagiarism or academic dishonesty, assuming I see it as a minor one, the student will receive an "F" for that assignment. At the second minor offense, or any such offense I see as major, the student will receive an "F" in the course. In addition, ALL offenses of plagiarism, including the any minor ones, will be reported to the Associate Provost for Academic Services as detailed in the Berea College Student Handbook.

Technology Policies

Much of the work in this course will require use of a computer, so these policies are designed to help you better understand how to be effective in a technology-rich environment.

Laptop and Software: We will regularly make use of laptops during class, and you are expected to have them unless explicitly stated otherwise. However, I will sometimes ask that you close your laptops in order to fully participate in discussion or critique.

Cell Phones: You may not use a cell phone during class - neither to talk or to text - except in case of emergency. (If you are expecting an emergency call, let me know before class and step out quietly when you receive it.)

Communication: We will use a few different tools to aid in communication throughout the semester.

- **Trello**: Our Trello “board” will serve as our course website. Assignments, due dates, and support resources will all be listed here.
- **Slack**: Slack is a tool for realtime collaboration - and we’ll use it for announcements and communication in between classes. This is the best way to ask myself, our TA, and your classmates for help or clarification. You should either plan to check Slack on a daily basis or configure notifications to go to your e-mail and/or phone.
- **Moodle**: Moodle is Berea College’s course content system. We’ll use Moodle for assignment submission, grades, and feedback on assignments (when applicable).

These are all mechanisms you would likely use in a professional position in the real world. You are, likewise, expected to use them in a responsible and professional manner in this course.

File Submission: It is your responsibility to verify that you have uploaded the correct file(s) for submission to Moodle. You should download a copy of each file after submission to check that it is the file you intended and that it can be opened.

Backups: All students are expected to back-up their work on a daily basis. The best way to do this is to store a copy of all work in a cloud service such as Dropbox, BereaBox, Google Drive, or to use a flash drive. Storing multiple copies of something on your laptop is not a backup. It is your responsibility to ensure that you do not lose any of your work throughout the term.

Statements Regarding Accommodations

Berea College will provide reasonable accommodations for students with disabilities to make all learning experiences accessible. If you feel you may need accommodations based on the impact of a disability of health condition, please contact Lisa Ladanyi (DAS - Disability & Accessibility Services, 111 Lincoln Hall, 859-985-3237, lisa.ladanyi@berea.edu) to initiate a conversation about your options. Students must provide their instructor(s) with an accommodation letter before any accommodations can be provided. Accommodations cannot be provided retroactively. Please meet with your instructor(s) in a confidential environment to discuss arrangements for these accommodations.

Under Title IX of the Education Amendments of 1972, pregnant and parenting students may be afforded certain accommodations regarding their educational experience. If you believe that pregnancy or pregnancy-related conditions are likely to impact your participation in this course, please contact Berea's Title IX Coordinator, Katie Basham, to discuss appropriate accommodations. She may be reached at katherine_basham@berea.edu or 859-985-3606.

Open Source Software Engineering Syllabus

Spring 2018

Instructor: Emily Lovell
Instructor e-mail: emily_lovell@berea.edu
Office hours: Mon 3pm-5pm, Wed 1pm-3pm
Office: Danforth Tech 102B
Phone: (859) 985-3930
Course Number: CSC 426

Class time: Tues/Thurs 1:00pm-2:50pm
Classroom: Danforth Tech 104

Note: This syllabus is subject to changes and updates throughout the term.

Catalog Description

An introduction to open source software engineering, this course covers the philosophy and practice of developing software in large, distributed teams. Students will explore social and technological aspects of this process by contributing directly to an existing open source project. In doing so, students will learn the tools, the techniques, and the strategies — technical and social — that are common to all developers working in teams through this semester-long exercise.

Course Goals

These goals provide the "big picture" of our work developing and contributing to open source communities with collaborators from around the world.

Open Community and Culture

Open software happens in a rich cultural context. It is important not only that we understand who we are working with, but the cultural values they embrace and exemplify through their work within a community.

Tools and Technologies

All open projects involve the use of tools…. lots and lots of tools. Whether they are wikis, or chat systems, or mailing lists, or version control systems, or bug trackers, or… the list goes on and on. This course will introduce you to some of these tools and help you establish the mindset that you will be learning new tools for the rest of your life as you continue to work in computing.

Development and Process

Developing software is a collaborative process. It involves communication with others, agreement on design, agreement on style, and agreement on when to agree and when to disagree. These dialogues all have a process and the technical act of designing, implementing, committing, testing, and publishing software all have processes as well. Understanding these layers of human- and technical-oriented process are critical to modern software development.

Collaboration and Communication

Working with people is hard. It involves patience, kindness, humility, and an awareness of self that many people working in technology do not have or even realize they should focus on developing. We will read about, practice, and engage in readings and discussions about who we are, how we work with others, and how to reflect on those interactions productively in the context of collaborative, distributed software development.

Texts & Required Reading

We will be using Slack as our primary means to communicate outside of class. This is where I will post announcements, additional information regarding assignments, etc. It is also a place where you can ask one another for help or advice. For this reason, I consider our course's Slack channel to be required reading. Once you've joined our team and channel, you can set up notifications that go to your phone and/or e-mail.

We will pull readings primarily from two books, both of which are available for free online.

The Cathedral and the Bazaar:
Musings on Linux and Open Source by an Accidental Revolutionary
By Eric S. Raymond
http://www.catb.org/esr/writings/cathedral-bazaar/

The Art of Community:
Building the New Age of Participation (Second Edition)
By Jono Bacon
http://artofcommunityonline.org/Art_of_Community_Second_Edition.pdf

Grade Distribution

Blog: 40%

You will be blogging regularly throughout this class, which will allow me to assess your learning process as well as progress towards your own goals. Prompts will include reading responses,

in-class activity reports, and reflections on contributing to an open source community (including any obstacles which you may encounter).

In-class participation and activities: 25%

We will frequently spend class time learning about new tools through hands-on activities. Later in the semester, you will be asked to share periodic updates on your own involvement with an open source community.

Peer evaluation: 10%

Because you will be working in small teams (or pairs) for much of the semester, a portion of your grade will come from peer evaluation. This will be a combination of you making a sincere effort to assess your teammates and also the grades which they assign to you.

Final portfolio: 25%

Your final portfolio will be a combination of a written reflection, any deliverables that you have worked on, and an in-class presentation. It will offer the opportunity to share what you've learned throughout the semester and to reflect on progress towards your own learning goals.

Late Work Policy

Because we will be building upon prior coursework throughout the term, it's extremely important that you complete assignments on time. However, I will accept *one* assignment late without penalty. All other late assignments will be reduced by a flat penalty of 20%. **No submissions will be accepted beyond three days late.** No extensions will be granted on in-class presentations or your final portfolio. Exceptions will be considered only in case of severe illness, hospitalization, etc. which will be handled on a case-by-case basis.

Grading Scale

100%	≥	A	≥	93%	>	A–	≥	90%			
90%	>	B+	≥	87%	>	B	≥	83%	>	B–	≥	80%
80%	>	C+	≥	77%	>	C	≥	73%	>	C–	≥	70%
70%	>	D+	≥	67%	>	D	≥	63%	>	F	≥	0

Attendance Policy

We will be actively engaging during class time in small group exercises, pair programming, presentations, and projects. Your attendance is critical in supporting these activities and for this reason my attendance policy is as follows:

- I'll take attendance at each class, once at the beginning.
- If you arrive late, it's your responsibility to make sure I've included your name on the attendance list. It's my prerogative to count late arrivals as ½ absence. (The same applies if you leave early or do not participate in class.)
- After 3 absences, your course grade will be reduced by ⅓ of a letter grade per absence. There is no distinction between excused and unexcused absences (except in very unusual cases of emergency such as family death or severe illness, which you must discuss with me and which must be documented).
- More than 6 absences (or roughly 20% of the course) automatically results in a failing grade for the course.

If you miss class for any reason, you are responsible for getting yourself caught up. This means taking it upon yourself to talk to classmates about missed material — after which you should follow up with me if you have outstanding questions.

Delays

In rare instances, I may be delayed arriving to class. If I have not arrived by the time class is scheduled to start, you must wait a minimum of 20 minutes for my arrival. In the event that I will miss class entirely, this will be communicated on Slack along with any additional information regarding assignments.

Academic Honesty

This course adheres fully to the college's policies regarding academic honesty/dishonesty.

In the preparation and presentation of any assigned work, all students shall conform to a strict standard of academic honesty. Any attempt to pass-off the ideas or work of others as your original work or to help another student to do so will be considered a violation of this standard. Thus, you must formally acknowledge in any work you submit the words and/or ideas of others taken from other students or from any print source or any electronic media, whether a direct quotation such as a cut-and paste or a paraphrase. Any omission of this standard, however minor, is dishonest and is called plagiarism. In the real-word, plagiarism is considered theft. In

the workplace, such theft can lead to lawsuits which cost the company time, money, and prestige. In this course, you must clearly document in everything you submit what is your own original work and what is the work of others. Academic dishonesty also includes presenting fabricated data as authentic.

At the first instance of plagiarism or academic dishonesty, assuming I see it as a minor one, the student will receive an "F" for that assignment. At the second minor offense, or any such offense I see as major, the student will receive an "F" in the course. In addition, ALL offenses of plagiarism, including the any minor ones, will be reported to the Associate Provost for Academic Services as detailed in the Berea College Student Handbook.

Technology Policies

Much of the work in this course will require use of a computer, so these policies are designed to help you better understand how to be effective in a technology-rich environment.

Laptop and Software: We will regularly make use of laptops during class, and you are expected to have them unless explicitly stated otherwise. However, I will sometimes ask that you close your laptops in order to fully participate in discussion or presentations.

Cell Phones: You may not use a cell phone during class - neither to talk or to text - except in case of emergency. (If you are expecting an emergency call, let me know before class and step out quietly when you receive it.)

Communication: We will use a few different tools to aid in communication throughout the semester.

- *Trello*: Our Trello “board” will serve as our course website. Assignments, due dates, and support resources will all be listed here.
- *Slack*: Slack is a tool for realtime collaboration - and we’ll use it for announcements and communication in between classes. This is the best way to ask myself, our TA, and your classmates for help or clarification. You should either plan to check Slack on a daily basis or configure notifications to go to your e-mail and/or phone.
- *Moodle*: Moodle is Berea College’s course content system. We’ll use Moodle for assignment submission, grades, and feedback on assignments (when applicable).

These are all mechanisms you would likely use in a professional position in the real world. You are, likewise, expected to use them in a responsible and professional manner in this course.

File Submission: It is your responsibility to verify that you have uploaded the correct file(s) for submission to Moodle. You should download a copy of each file after submission to check that it is the file you intended and that it can be opened. Similarly, it is your responsibility to verify that your blog entries are published and publically accessible.

Backups: All students are expected to back-up their work on a daily basis. The best way to do this is to store a copy of all work in a cloud service such as Dropbox, BereaBox, Google Drive, or to use a flash drive. Storing multiple copies of something on your laptop is not a backup. It is your responsibility to ensure that you do not lose any of your work throughout the term.

Statements Regarding Accommodations

Berea College will provide reasonable accommodations for students with disabilities to make all learning experiences accessible. If you feel you may need accommodations based on the impact of a disability of health condition, please contact Lisa Ladanyi (DAS - Disability & Accessibility Services, 111 Lincoln Hall, 859-985-3237, lisa.ladanyi@berea.edu) to initiate a conversation about your options. Students must provide their instructor(s) with an accommodation letter before any accommodations can be provided. Accommodations cannot be provided retroactively. Please meet with your instructor(s) in a confidential environment to discuss arrangements for these accommodations.

Under Title IX of the Education Amendments of 1972, pregnant and parenting students may be afforded certain accommodations regarding their educational experience. If you believe that pregnancy or pregnancy-related conditions are likely to impact your participation in this course, please contact Berea's Title IX Coordinator, Katie Basham, to discuss appropriate accommodations. She may be reached at katherine_basham@berea.edu or 859-985-3606.

References

[1] Agrawal, R., Springer, A. and Lovell, E. 2015. QuickResponseHost: Enabling crowdsourced disaster response stations. *2015 IEEE Global Humanitarian Technology Conference (GHTC)* (Oct. 2015), 233–239.

[2] Alfieri, L., Brooks, P.J., Aldrich, N.J. and Tenenbaum, H.R. 2011. Does discovery-based instruction enhance learning? *Journal of educational psychology*. 103, 1 (2011), 1.

[3] Arduino: *https://www.arduino.cc/*. Accessed: 2020-06-15.

[4] Astrachan, O., Barnes, T., Garcia, D.D., Paul, J., Simon, B. and Snyder, L. 2011. CS principles: piloting a new course at national scale. *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (2011), 397–398.

[5] Bacon, J. 2012. *The art of community: Building the new age of participation*. O'Reilly Media, Inc.

[6] Bandura, A. 1982. Self-efficacy mechanism in human agency. *American psychologist*. 37, 2 (1982), 122.

[7] Bandura, A. 1977. Self-efficacy: toward a unifying theory of behavioral change. *Psychological review*. 84, 2 (1977), 191.

[8] Bandura, A. and Wessels, S. 1994. Self-efficacy. na.

[9] Beyer, K. 2012. *Grace Hopper and the invention of the information age*. Lemelson Center Studies in Inv.

[10] Braught, G. 2021. Support for Broadening Participation though Humanitarian Free and Open Source Software. *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (2021), 1306–1306.

[11] Braught, G., Maccormick, J., Bowring, J., Burke, Q., Cutler, B., Goldschmidt, D., Krishnamoorthy, M., Turner, W., Huss-Lederman, S., Mackellar, B. and Tucker, A. 2018. A Multi-Institutional Perspective on H/FOSS Projects in the Computing Curriculum. *ACM Transactions on Computing Education*. 18, 2 (Jul. 2018), 7:1-7:31. DOI:https://doi.org/10.1145/3145476.

[12] Bridging the Computer Science Access Gap (Infographics) (August 2016): *https://ecs.secure.force.com/studies/rstempg?id=a0r0g000009TLeI*. Accessed:

2021-12-07.

[13] Brown, M. 2013. CS0 as an indicator of student risk for failure to complete a degree in computing. *Journal of Computing Sciences in Colleges*. 28, 5 (2013), 9–16.

[14] Buechley, L. and Eisenberg, M. 2008. The LilyPad Arduino: Toward wearable engineering for everyone. *IEEE Pervasive Computing*. 7, 2 (2008), 12–15.

[15] Buechley, L., Eisenberg, M., Catchen, J. and Crockett, A. 2008. The LilyPad Arduino: Using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. *Proceeding of the Twenty-Sixth Annual CHI Conference on Human Factors in Computing Systems - CHI '08* (Florence, Italy, 2008), 423.

[16] Buechley, L., Eisenberg, M. and Elumeze, N. 2007. Towards a curriculum for electronic textiles in the high school classroom. *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (2007), 28–32.

[17] Buechley, L. and Hill, B.M. 2010. LilyPad in the wild: how hardware's long tail is supporting new engineering and design communities. *Proceedings of the 8th ACM Conference on Designing Interactive Systems - DIS '10* (Aarhus, Denmark, 2010), 199.

[18] Buechley, L., Mellis, D., Perner-Wilson, H., Lovell, E. and Kaufmann, B. 2010. Living wall: programmable wallpaper for interactive spaces. *Proceedings of the international conference on Multimedia - MM '10* (Firenze, Italy, 2010), 1401.

[19] Buechley, L., Peppler, K., Eisenberg, M. and Yasmin, K. 2013. *Textile Messages: Dispatches from the World of E-Textiles and Education. New Literacies and Digital Epistemologies. Volume 62.* ERIC.

[20] Buechley, L. and Qiu, K. 2014. *Sew electric*. H.

[21] Calendly: *https://calendly.com/*. Accessed: 2021-05-15.

[22] Cheryan, S., Master, A. and Meltzoff, A.N. 2015. Cultural stereotypes as gatekeepers: Increasing girls' interest in computer science and engineering by diversifying stereotypes. *Frontiers in Psychology*. 6, (2015), 49.

[23] Cheryan, S., Plaut, V.C., Davies, P.G. and Steele, C.M. 2009. Ambient belonging: how stereotypical cues impact gender participation in computer

science. *Journal of Personality and Social Psychology*. 97, 6 (2009), 1045.

[24] Craft of Computing (Course Website): *https://trello.com/b/WGeaxo5n/craft-of-computing*. Accessed: 2021-05-15.

[25] Cutts, Q., Cutts, E., Draper, S., O'Donnell, P. and Saffrey, P. 2010. Manipulating mindset to positively influence introductory programming performance. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (2010), 431–435.

[26] Diekman, A.B., Brown, E.R., Johnston, A.M. and Clark, E.K. 2010. Seeking congruity between goals and roles: A new look at why women opt out of science, technology, engineering, and mathematics careers. *Psychological Science*. 21, 8 (2010), 1051–1057.

[27] Dweck, C.S. 2013. *Self-theories: Their role in motivation, personality, and development*. Psychology press.

[28] Ellis, H.J., Chua, M., Hislop, G.W., Purcell, M. and Dziallas, S. 2013. Towards a model of faculty development for FOSS in education. *2013 26th International Conference on Software Engineering Education and Training (CSEE&T)* (2013), 269–273.

[29] Ellis, H.J., Chua, M., Jadud, M.C. and Hislop, G.W. 2011. Learning through open source participation. *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (2011), 83–84.

[30] Ellis, H.J.C., Hislop, G.W., Jackson, S. and Postner, L. 2015. Team Project Experiences in Humanitarian Free and Open Source Software (HFOSS). *ACM Transactions on Computing Education*. 15, 4 (Dec. 2015), 18:1-18:23. DOI:https://doi.org/10.1145/2684812.

[31] Ellis, H.J.C., Jackson, S., Burdge, D., Postner, L., Hislop, G.W. and Diggs, J. 2014. Learning within a professional environment: shared ownership of an HFOSS project. *Proceedings of the 15th Annual Conference on Information Technology Education* (New York, NY, USA, Oct. 2014), 95–100.

[32] Eng, D. 2009. *Fashion Geek: Clothes Accessories Tech*. North Light Books.

[33] e-Textiles-in-a-Box: *https://www.ncwit.org/resources/e-textiles-box*. Accessed: 2020-12-08.

[34] Fields, D.A., Kafai, Y.B. and Searle, K. 2012. Functional aesthetics for learning: Creative tensions in youth e-textile designs. (2012).

[35] Firefox Developer Tools | MDN: *https://developer.mozilla.org/en-US/docs/Tools*. Accessed: 2021-05-15.

[36] Follmer, S., Carr, D., Lovell, E. and Ishii, H. 2010. CopyCAD: remixing physical objects with copy and paste from the real world. *Adjunct proceedings of the 23nd annual ACM symposium on User interface software and technology - UIST '10* (New York, New York, USA, 2010), 381.

[37] Foss2Serve: *http://foss2serve.org/index.php/Main_Page*. Accessed: 2021-05-15.

[38] Freeman, S., Eddy, S.L., McDonough, M., Smith, M.K., Okoroafor, N., Jordt, H. and Wenderoth, M.P. 2014. Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the national academy of sciences*. 111, 23 (2014), 8410–8415.

[39] Getting Started with E-Sewing: *http://alumni.media.mit.edu/~emme/e-sewing/index.html*. Accessed: 2020-06-08.

[40] Greenberg, I. 2007. *Processing: creative coding and computational art*. Apress.

[41] Hartman, K., Jepson, B., Dvorak, E. and Demarest, R. 2014. *Make: wearable electronics*. Maker Media.

[42] Haungs, M., Clark, C., Clements, J. and Janzen, D. 2012. Improving first-year success and retention through interest-based CS0 courses. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education* (2012), 589–594.

[43] Hébert, C. and Jenson, J. 2020. Making in schools: student learning through an e-textiles curriculum. *Discourse: Studies in the Cultural Politics of Education*. 41, 5 (Sep. 2020), 740–761. DOI:https://doi.org/10.1080/01596306.2020.1769937.

[44] High-Low Tech: *http://highlowtech.org/*. Accessed: 2020-06-15.

[45] Hill, C., Schneider, M., Eisenberg, A. and Gross, M.D. 2021. The ThreadBoard: Designing an E-Textile Rapid Prototyping Board. *Proceedings of the Fifteenth International Conference on Tangible, Embedded, and Embodied Interaction* (2021), 1–7.

[46] Hislop, G.W., Ellis, H.J., Tucker, A.B. and Dexter, S. 2009. Using open source software to engage students in computer science education. *Proceedings of the*

40th ACM Technical Symposium on Computer Science Education (2009), 134–135.

[47] Hislop, G.W., Ellis, H.J.C. and Morelli, R.A. 2009. Evaluating student experiences in developing software for humanity. *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education* (New York, NY, USA, Jul. 2009), 263–267.

[48] Hislop, G.W., Ellis, H.J.C., Pulimood, S.M., Morgan, B., Mello-Stark, S., Coleman, B. and Macdonell, C. 2015. A Multi-Institutional Study of Learning via Student Involvement in Humanitarian Free and Open Source Software Projects. *Proceedings of the Eleventh Annual International Conference on International Computing Education Research* (New York, NY, USA, Aug. 2015), 199–206.

[49] Höhne, E. and Zander, L. 2019. Belonging uncertainty as predictor of dropout intentions among first-semester students of the computer sciences. *Zeitschrift für Erziehungswissenschaft*. 22, 5 (2019), 1099–1119.

[50] How To Get What You Want: *https://www.kobakant.at/DIY/*. Accessed: 2021-05-15.

[51] Instructables | bekathwia (Becky Stern): *https://www.instructables.com/member/bekathwia/*. Accessed: 2020-12-10.

[52] Instructables | Plusea (Hannah Perner-Wilson): *https://www.instructables.com/member/Plusea/*. Accessed: 2020-12-10.

[53] Introducing debugger.html – Mozilla Hacks - the Web developer blog: *https://hacks.mozilla.org/2016/09/introducing-debugger-html*. Accessed: 2021-05-15.

[54] Introducing Gemma: Introducing Adafruit's mini wearable microcontroller: 2013. *https://learn.adafruit.com/introducing-gemma/introduction*. Accessed: 2021-08-10.

[55] Jackson, S. and Ellis, H. 2015. Supporting HFOSS using scrum in a capstone course. *Acm Sigcas Computers and Society*. 45, 2 (2015), 36–37.

[56] Jayathirtha, G. and Kafai, Y.B. 2019. Electronic textiles in computer science education: a synthesis of efforts to broaden participation, increase interest, and deepen learning. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (2019), 713–719.

[57] Kafai, Y.B., Fields, D.A. and Searle, K.A. 2011. Everyday creativity in novice

e-textile designs. *Proceedings of the 8th ACM conference on Creativity and cognition* (2011), 353–354.

[58] Kafai, Y.B., Lee, E., Searle, K., Fields, D., Kaplan, E. and Lui, D. 2014. A crafts-oriented approach to computing in high school: Introducing computational concepts, practices, and perspectives with electronic textiles. *ACM Transactions on Computing Education (TOCE)*. 14, 1 (2014), 1–20.

[59] Kafai, Y.B., Searle, K., Kaplan, E., Fields, D., Lee, E. and Lui, D. 2013. Cupcake cushions, scooby doo shirts, and soft boomboxes: e-textiles in high school to promote computational concepts, practices, and perceptions. *Proceeding of the 44th ACM technical symposium on Computer science education* (2013), 311–316.

[60] Kim, V.H. 2019. *Development of an e-Textile Debugging Module to Increase Computational Thinking among Graduate Education Students*. Pepperdine University.

[61] Lee, J.S. 2008. Tech DIY for moms and kids: the DIY technology project for women. *ACM SIGGRAPH 2008 posters* (2008), 1–1.

[62] Lee, J.S. 2008. Technology education for women by D.I.Y. technology in closing gender gap. *CHI'08 Extended Abstracts on Human Factors in Computing Systems* (2008), 3447–3452.

[63] Lewis, A., Lin, F.-Y., Weston, H. and Sugie, H. 2008. *Switch craft: battery-powered crafts to make and sew*. Potter Craft.

[64] Lewis, C., Bruno, P., Raygoza, J. and Wang, J. 2019. Alignment of goals and perceptions of computing predicts students' sense of belonging in computing. *Proceedings of the 2019 ACM Conference on International Computing Education Research* (2019), 11–19.

[65] Lewis, C.M., Yasuhara, K. and Anderson, R.E. 2011. Deciding to major in computer science: a grounded theory of students' self-assessment of ability. *Proceedings of the seventh international workshop on Computing education research* (2011), 3–10.

[66] Light, J.S. 1999. When computers were women. *Technology and culture*. 40, 3 (1999), 455–483.

[67] LilyTiny - DEV-10899 - SparkFun Electronics: *https://www.sparkfun.com/products/10899*. Accessed: 2021-11-15.

[68] Lodi, M. 2018. Can creative computing foster growth mindset? *Joint*

Proceedings of the 1st Co-Creation in the Design, Development and Implementation of Technology-Enhanced Learning workshop (CC-TEL 2018) and Systems of Assessments for Computational Thinking Learning workshop (TACKLE 2018) co-located with 13th European Conference on Technology Enhanced Learning (ECTEL 2018) (2018).

[69] Lovell, E. 2011. Getting Hands-On with Soft Circuits: A Workshop Facilitator's Guide.

[70] Lovell, E. 2014. Promoting constructive mindsets for overcoming failure in computer science education. *Proceedings of the tenth annual conference on International computing education research - ICER '14* (Glasgow, Scotland, United Kingdom, 2014), 159–160.

[71] Lovell, E. and Buechley, L. 2010. An e-sewing tutorial for DIY learning. *Proceedings of the 9th International Conference on Interaction Design and Children - IDC '10* (Barcelona, Spain, 2010), 230.

[72] Lovell, E. and Buechley, L. 2011. LilyPond: an online community for sharing e-textile projects. *Proceedings of the 8th ACM conference on Creativity and cognition - C&C '11* (Atlanta, Georgia, USA, 2011), 365.

[73] Lovell, E., Buechley, L. and Davis, J. 2022. The LilyTiny: A Case Study in Expanding Access to Electronic Textiles. *CHI'22 Extended Abstracts on Human Factors in Computing Systems*.

[74] Lovell, E. and Davis, J. 2021. Craft of Computing: Using a Novel Domain to Broaden Undergraduate Participation and Perceptions of Computing at the CS0 Level. *2021 IEEE Frontiers in Education Conference (FIE)* (2021).

[75] Lovell, E. and Davis, J. 2021. Scaffolding Student Success in the Wilds of Open Source Contribution. *2021 IEEE Frontiers in Education Conference (FIE)* (2021).

[76] Lovell, E., Qi, J. and Freed, N. 2011. Plush Monsters: Creatures with Character.

[77] Lovell, E.M. 2011. *A Soft Circuit Curriculum to Promote Technological Self-Efficacy*. Massachusetts Institute of Technology.

[78] Maker Ed's Maker Corps Program: *https://makered.org/makercorps/*. Accessed: 2021-11-13.

[79] Margolis, J. 2010. *Stuck in the Shallow End: Education, Race, and Computing*.

MIT Press.

[80] Margolis, J. and Fisher, A. 2003. *Unlocking the Clubhouse: Women in Computing*. MIT Press.

[81] Master, A., Cheryan, S. and Meltzoff, A.N. 2016. Computing whether she belongs: Stereotypes undermine girls' interest and sense of belonging in computer science. *Journal of Educational Psychology*. 108, 3 (2016), 424.

[82] McConnell, J.J. 1996. Active learning and its use in computer science. *Proceedings of the 1st Conference on integrating Technology into Computer Science Education* (1996), 52–54.

[83] McDowell, C., Werner, L., Bullock, H.E. and Fernald, J. 2006. Pair programming improves student retention, confidence, and program quality. *Communications of the ACM*. 49, 8 (2006), 90–95.

[84] McDowell, C., Werner, L., Bullock, H.E. and Fernald, J. 2003. The impact of pair programming on student performance, perception and persistence. *25th International Conference on Software Engineering, 2003. Proceedings.* (2003), 602–607.

[85] Mellis, D.A., Banzi, M., Cuartielles, D. and Igoe, T. 2007. Arduino: An Open Electronics Prototyping Platform. *Proceedings of CHI* (2007), 1–11.

[86] Menabrea, L.F. and Lovelace, A. 1842. Sketch of the analytical engine invented by Charles Babbage. (1842).

[87] Mills, J.E. and Treagust, D.F. 2003. Engineering education—Is problem-based or project-based learning the answer. *Australasian journal of engineering education*. 3, 2 (2003), 2–16.

[88] Miura, I.T. 1987. The relationship of computer self-efficacy expectations to computer interest and course enrollment in college. *Sex roles*. 16, 5–6 (1987), 303–311.

[89] Morales-Chicas, J., Castillo, M., Bernal, I., Ramos, P. and Guzman, B.L. 2019. Computing with relevance and purpose: A review of culturally relevant education in computing. *International Journal of Multicultural Education*. 21, 1 (2019), 125–155.

[90] Morelli, R., de Lanerolle, T. and Tucker, A. 2012. The Humanitarian Free and Open-Source Software Project: Engaging Students in Service-Learning through Building Software. *Service-Learning in the Computer and Information*

Sciences: Practical Applications in Engineering Education. (2012), 117–136.

[91] Morgan, B., Hislop, G.W. and Ellis, H.J. 2019. Faculty Development for FLOSS Education. *IFIP International Conference on Open Source Systems* (2019), 165–171.

[92] Morgan, R. and Klaric, J. 2007. *AP® Students in College: An Analysis of Five-Year Academic Careers. Research Report No. 2007-4*. College Board.

[93] Moxie 2010. *I Felt Awesome: tips and tricks for 35+ needle-poked projects*. North Light Books.

[94] Murphy, L. and Thomas, L. 2008. Dangers of a fixed mindset: implications of self-theories research for computer science education. *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education* (2008), 271–275.

[95] Narayanan, S., Cunningham, K., Arteaga, S., Welch, W.J., Maxwell, L., Chawinga, Z. and Su, B. 2018. Upward mobility for underrepresented students: A model for a cohort-based bachelor's degree in computer science. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (2018), 705–710.

[96] National Center for Science and Engineering Statistics 2021. *Women, Minorities, and Persons with Disabilities in Science and Engineering: 2021*. National Science Foundation.

[97] NCWIT Programs-in-a-Box: *https://www.ncwit.org/resources/type/programs-box*. Accessed: 2020-12-07.

[98] New Textiles 2012: *https://newtextiles.media.mit.edu/*. Accessed: 2020-06-17.

[99] Ngai, G., Chan, S.C., Leong, H.V. and Ng, V.T. 2013. Designing i* CATch: A multipurpose, education-friendly construction kit for physical and wearable computing. *ACM Transactions on Computing Education (TOCE)*. 13, 2 (2013), 1–30.

[100] Ngai, G., Chan, S.C.F., Cheung, J.C.Y. and Lau, W.W.Y. 2009. The TeeBoard: an education-friendly construction platform for e-textiles and wearable computing. *Proceedings of the 27th international conference on Human factors in computing systems - CHI 09* (Boston, MA, USA, 2009), 249.

[101] Ngai, G., Chan, S.C.F., Ng, V.T.Y., Cheung, J.C.Y., Choy, S.S.S., Lau, W.W.Y. and Tse, J.T.P. 2010. i*CATch: a scalable plug-n-play wearable computing framework for novices and children. *Proceedings of the 28th international*

conference on Human factors in computing systems - CHI '10 (Atlanta, Georgia, USA, 2010), 443.

[102] Olsson, T. 2011. *Open Softwear: fashionable prototyping and wearable computing using the Arduino*. Blushing Boy.

[103] Open Source Comes to Campus: *https://campus.openhatch.org/*. Accessed: 2021-05-15.

[104] Page, S. 2019. *The diversity bonus*. Princeton University Press.

[105] Pakhchyan, S. 2008. *Fashioning Technology: A DIY Intro to Smart Crafting*. O'Reilly Media, Inc.

[106] Papert, S. 1993. *The children's machine: Rethinking school in the age of the computer.* ERIC.

[107] Paulsen, C.A., Green, S. and Carroll, S. 2011. *Design Squad Nation: Evaluation report*. Concord Evaluation Group, LLC.

[108] Pearce, J. and Nakazawa, M. 2008. The funnel that grew our CIS major in the CS desert. *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education* (2008), 503–507.

[109] Perner-Wilson, H. and Buechley, L. 2010. Making textile sensors from scratch. *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction* (2010), 349–352.

[110] Picard, R.W., Papert, S., Bender, W., Blumberg, B., Breazeal, C., Cavallo, D., Machover, T., Resnick, M., Roy, D. and Strohecker, C. 2004. Affective learning—a manifesto. *BT technology journal*. 22, 4 (2004), 253–269.

[111] Pirker, J., Riffnaller-Schiefer, M. and Gütl, C. 2014. Motivational active learning: engaging university students in computer science education. *Proceedings of the 2014 conference on Innovation & technology in computer science education* (2014), 297–302.

[112] Postner, L., Burdge, D., Jackson, S., Ellis, H., Hislop, G. and Goggins, S. 2015. Using humanitarian free and open source software (HFOSS) to introduce computing for the social good. *ACM SIGCAS Computers and Society*. 45, 2 (2015), 35–35.

[113] Professors' Open Source Software Experience: *http://foss2serve.org/index.php/POSSE*. Accessed: 2021-11-13.

[114] Qi, J. 2012. *The fine art of electronics: paper-based circuits for creative*

expression. Massachusetts Institute of Technology.

[115] Qi, J. and Buechley, L. 2014. Sketching in circuits: designing and building electronics on paper. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2014), 1713–1722.

[116] Qiu, K., Buechley, L., Baafi, E. and Dubow, W. 2013. A curriculum for teaching computer science through computational textiles. *Proceedings of the 12th international conference on interaction design and children* (2013), 20–27.

[117] Raymond, E.S. 2001. *The cathedral and the bazaar: musings on Linux and Open Source by an accidental revolutionary*. O'Reilly.

[118] React – A JavaScript library for building user interfaces: *https://reactjs.org/*. Accessed: 2021-05-15.

[119] Reas, C. and Fry, B. 2014. *Processing: a programming handbook for visual designers and artists*. The MIT Press.

[120] Reas, C. and Fry, B. 2006. Processing: Programming for the media arts. *AI & SOCIETY*. 20, 4 (Sep. 2006), 526–538. DOI:https://doi.org/10.1007/s00146-006-0050-9.

[121] Sahana Eden: 2011. *https://sahanafoundation.org/products/eden/*. Accessed: 2021-11-13.

[122] Searle, K.A. and Kafai, Y.B. 2015. Boys' Needlework: Understanding Gendered and Indigenous Perspectives on Computing and Crafting with Electronic Textiles. *ICER* (2015), 31–39.

[123] Shiffman, D. 2009. *Learning Processing: a beginner's guide to programming images, animation, and interaction*. Morgan Kaufmann.

[124] Simon, B., Hanks, B., Murphy, L., Fitzgerald, S., McCauley, R., Thomas, L. and Zander, C. 2008. Saying isn't necessarily believing: influencing self-theories in computing. *Proceedings of the Fourth International Workshop on Computing Education Research* (2008), 173–184.

[125] Slack: *https://slack.com/*. Accessed: 2021-05-15.

[126] Soft Circuit Saturdays: *https://www.gellacraft.com/softcircuitsaturdays*. Accessed: 2020-12-05.

[127] SparkFun Education - Maker Education:

https://sparkfuneducation.com/index.html. Accessed: 2021-08-12.

[128] Stern, B. and Cooper, T. 2015. *Getting started with Adafruit FLORA: making wearables with an Arduino-compatible electronics platform*. Maker Media, Inc.

[129] The Tinkering Studio Home | Exploratorium: *https://www.exploratorium.edu/tinkering*. Accessed: 2021-11-13.

[130] Trello: *https://trello.com/*. Accessed: 2021-05-15.

[131] Turkle, S. and Papert, S. 1992. Epistemological pluralism and the revaluation of the concrete. *Journal of Mathematical Behavior*. 11, 1 (1992), 3–33.

[132] Veilleux, N., Bates, R., Allendoerfer, C., Jones, D., Crawford, J. and Floyd Smith, T. 2013. The relationship between belonging and ability in computer science. *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (2013), 65–70.

[133] Weng, J. and Murphy, C. 2018. Bridging the Diversity Gap in Computer Science with a Course on Open Source Software. *2018 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)* (Baltimore, MD, Feb. 2018), 1–4.

[134] Werner, L.L., Hanks, B. and McDowell, C. 2004. Pair-programming helps female computer science students. *Journal on Educational Resources in Computing (JERIC)*. 4, 1 (2004), 4-es.

[135] Wolsky, M. 2014. Design Squad: Inspiring a New Generation of Engineers. *The Go-To Guide for Engineering Curricula, Grades 6-8: Choosing and Using the Best Instructional Materials for Your Students*. Corwin Press. 19.

[136] Wood, Z.J., Clements, J., Peterson, Z., Janzen, D., Smith, H., Haungs, M., Workman, J., Bellardo, J. and DeBruhl, B. 2018. Mixed approaches to CS0: Exploring topic and pedagogy variance after six years of CS0. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (2018), 20–25.

[137] Xu, D., Wolz, U., Kumar, D. and Greenburg, I. 2018. Updating introductory computer science with creative computation. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (2018), 167–172.

[138] Yardi, S. and Bruckman, A. 2007. What is computing? Bridging the gap between teenagers' perceptions and graduate students' experiences. *Proceedings of the Third International Workshop on Computing Education*

Research (2007), 39–50.

[139] 2021. *Sahana Eden on GitHub*. Sahana Software Foundation.

[140] TeachingOpenSource – Instructors and open source communities supporting teaching open source.