2019

Development of an e-textile debugging module to increase computational thinking among graduate education students

Victoria Herbst Kim

Pepperdine University

Graduate School of Education and Psychology

DEVELOPMENT OF AN E-TEXTILE DEBUGGING MODULE TO INCREASE COMPUTATIONAL THINKING AMONG GRADUATE EDUCATION STUDENTS

A dissertation submitted in partial satisfaction
of the requirements for the degree of
Doctor of Education in Learning Technologies

by

Victoria Herbst Kim

April, 2019

Linda Polin, Ph.D. – Dissertation Chairperson

This dissertation, written by

Victoria Herbst Kim

under the guidance of a Faculty Committee and approved by its members, has been submitted to and accepted by the Graduate Faculty in partial fulfillment of the requirements for the degree of

DOCTOR OF EDUCATION

Doctoral Committee:

Linda Polin, Ph.D., Chairperson

Paul Sparks, Ph.D.

Jack McManus, Ph.D.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

DEDICATION

I dedicate this research to my husband, Paul, who has supported my work intellectually, emotionally, and financially. I am thankful, too, for his help with the transcriptions. I also dedicate my work to my parents, Chester and Jocelyn Herbst, who have encouraged my love of learning throughout my whole life. I give thanks for a lifetime of wonderful teachers and fellow students who have also taught me. Thanks, CFKA16, for making this such a fun experience, and thank you to my friends and family who believed I would get this done!

ACKNOWLEDGEMENTS

Dr. Linda Polin, the chair of this dissertation, created course projects that caused all of us to stretch and to learn to think like academicians. Her enthusiasm for computational thinking and learning, in general, has been infectious. Thank you, Dr. Paul Sparks and Dr. Jack McManus, for your support in my dissertation process. Thank you, Kay Davis, for helping us understand the research process.

Thanks also to the members of Cadre 18 who agreed to be participants in this study, and who shared some of their learning experience with me. The students from DeVry University's DBM449 class during the May session of 2013 were also instrumental as participants in the pilot study. Members of the Cadre Formerly Known as 16 (CFKA16) in Pepperdine University's Doctor of Learning Technologies program have been generous in sharing ideas, emotional support, and procedural steps as they advance through the program. One of the joys of this doctoral journey has been our association.

Marji McCaffery helped me with the creation of the numerous debugging sets that were used in the study. Thanks for your concern, friendship, encouragement, and many hours of your time!

Deborah Fields graciously shared the debugg'em's research (Fields, Searle, Kafai, & Min, 2012), which is one of the foundations upon which this dissertation is built. Brennan and Resnick's (2012) research about computational thinking concepts, practices, and perspectives is the other foundational study of this dissertation.

I am grateful to my employer, DeVry University, for the educational assistance they have provided for my doctoral studies.

The researcher and future participants of this study have benefited from SparkFun's generous educational discount, which has made the purchase of the LilyPad components used in this study feasible.

A Pepperdine University IT Grant has also supported the purchase of both LilyPad Arduino and Galaxy tablets for the EDLT740 course. This support, which has allowed research into the development of computational thinking, is greatly appreciated.

Finally, I would also like to acknowledge my appreciation and respect for Seymour Papert and his contributions to learning. Constructionism provides significant insight into how people can construct knowledge and is especially applicable to the current technological state of the world. Leah Buechley's LilyPad technology has provided a tool that allows me to express both my creative and logical sides. I am so glad that I have been able to explore ways that it can do this for other students.

VITA

Victoria Kim

E-mail: vhkim@pepperdine.edu

CAREER OBJECTIVE

To obtain a tenure track faculty position where I can apply my experience and education in Mathematics, Programming, and Learning Technology

EDUCATION

Pepperdine University, Graduate School of Education and Psychology
Doctor of Education in Learning Technologies Anticipated July 2019

Brigham Young University, Provo, UT
Master of Science in Mathematics August 1998

Monterey Institute of International Studies
Master of Arts in Teaching Foreign Language July 1992
Concentration Language, Spanish

Brigham Young University
Bachelor of Arts in Mathematics Education June 1979

CREDENTIALS HELD

California Professional Clear Single Subject Teaching Credential in Mathematics
Bilingual Cross-Cultural Certificate (BCC) Certification

TEACHING EXPERIENCE

DeVry University, Fremont, CA
Current Title: Professor, National Faculty Chair Summer 1999 - Present
Courses Taught: Developmental Mathematics, College Algebra, Pre-Calculus, Critical Thinking, Introduction to Database, Advanced Database, Database Administration, Advanced Spreadsheets

- New Faculty Orientation
- Teaching Excellence Course
- Bay Area Blended Training Champion
- San Francisco Bay Area Metro MyMathLab Coordinator
- DeVry eLearning Platform Champion for San Francisco Bay Area Metro

Seaside High School, Seaside, CA
Teacher August 1985 – June 1999
Courses Taught: Developmental Mathematics, Algebra, Geometry, Algebra2, AP Calculus, Bilingual Algebra, ESL Algebra, ESL Mathematics

- Class Advisor, Class of 1994
- Girls Tennis Coach
- Boys Volleyball Coach (1 year)
- Asian Club Founder and Advisor
- Planned and Executed Activities for Bilingual Students
- Math Tutor

TEACHING EXPERIENCE CONTINUED

Monterey Peninsula College, Monterey, CA
Adjunct Professor Fall 1998 – Spring 1999
Courses Taught: Pre-Algebra, Algebra

Brigham Young University, Provo, CA
Graduate Assistant August 1997 – May 1998
Courses taught: Calculus 1Discussion Sections, Math for Elementary Teachers

- Created and Administered Quizzes
- Presented Concrete Examples of Theoretical Concepts
- Created Projects to Support Learning Objectives

COMMITTEE WORK

DeVry University, Fremont, CA

- **Teaching and Learning Committee**
- **Faculty Training Committee**

PRESENTATIONS

Jensen, Crystal & Kim, Victoria (March 2013). “Native American Female Leadership Model to Address Indigenous Peoples’ Social Justice Issues”
Round Table, International Conference on Technology in College Mathematics

Kim, Victoria (April 2012). “Using Livescribe to Answer Students’ Emailed Math Questions”
DeVry University Faculty Symposium, Chicago, IL

Kim, Victoria (March 2012). “Using Wolfram|Alpha to Build Confidence in Developmental Algebra Students”
International Conference on Technology in College Mathematics

Kim, Victoria (March 2011). “Using Vyew to Answer Students’ Email Questions”
International Conference on Technology in College Mathematics

PROFESSIONAL MEMBERSHIPS

American Education Research Association (AERA)
Association for Computing Machinery (ACM)

HONORS / AWARDS

Pepperdine University Capstone Honors July 2012
Pepperdine University, West Los Angeles, CA

DeVry University Pride Award December 2010
DeVry University, Fremont, CA

Ron Taylor Award July 2008
DeVry University, Fremont, CA

Rated Number 1 by Students December 2004
CIS Department
DeVry University, Fremont, CA

Faculty of the Semester Fall 2002
DeVry University, Fremont, CA

Outstanding Graduate Assistant May 1998
Brigham Young University, Provo, UT

Outstanding Young Educator May 1995
Seaside Jaycees, Seaside, CA

ABSTRACT

The increased presence of technology in all aspects of daily life makes computational thinking a necessary skill. Predictions say that the rising need for computational thinkers will be unmet by computer science graduates. An e-textile learning module, based on principles of constructionism, was designed as a method to develop computational thinking skills and encourage interest and confidence in the computing fields in both male and female graduate education students. The module leveraged the affordances of the LilyPad Arduino, a technology that allows for the creation of projects that integrate textiles and electronics without soldering. The creation of the learning module relied on design-based research methodologies and followed the use-modify-create principle for the included activities. Multiple data sources were analyzed using The Computational Thinking Rubric for Examining Students' Project Work to examine artifacts and interactions for indications of computational thinking concepts, practices, and perspectives. Students participated in debugging activities and created their own projects as part of the learning module. Analysis of the learning module activities showed students using computational thinking concepts, engaged in computational thinking practices, and exhibiting computational thinking perspectives. During the coding process, several new computational thinking concepts, practices, and perspectives emerged. There was evidence of both an increase and decrease in confidence among the student participants. Improvements for the next iteration of the learning module were presented and the implications for the study of computational thinking explored. The study helps contradict the shrinking pipeline metaphor by showing that it is possible to encourage interest in computation in university students, not just middle-school students.

Chapter 1: Study Introduction

Problem Statement

With the increased presence of technology in daily life, the need for computational thinking is growing within many disciplines. This type of computational thinking is becoming a survival skill in today's environment, where most appliances are computerized, and tablet and phone applications abound. While computer science enrollment has finally been increasing, after several years of decline (Lacey & Wright, 2009), the increase is not enough to fill the predicted 22% rise in computer related jobs by 2018 (Zweben, 2012). A related problem is that women are severely underrepresented in the field of computing. This underrepresentation is detrimental, because it means that women are not getting the advantage of a well-compensated career in a growing field (Lacey & Wright, 2009). In addition, the field itself is not able to benefit from the diversity of women's points of view (Cox & Blake, 1991; Johansson, 2006; Page, 2008).

Traditionally, students develop their computational thinking skills in beginning programming classes, while also learning the specifics of a particular programming language. However, many students struggle with learning a new way of viewing problems at the same time they are learning to program in a specific language. The efforts to learn computational thinking and programming skills occur while students are also typically immersed in a very competitive and intimidating culture (Siegel, 1999). Alternative avenues need to be developed to encourage computational thinking so that a more diverse group of people can contribute to creatively solving current problems and designing innovative products.

Debugging is an essential part of computational thinking. While some aspects of computational thinking are still debated, there is agreement that debugging is one of the integral concepts (Polin, 2012). Not only is debugging an important part of computational thinking, it is

also influential in how students view computing and error correction (Lapidot & Hazzan, 2005). Experience with debugging can help students become confident about their ability to overcome any problems that occur in the process of implementing their projects.

In this study, a LilyPad Arduino e-textile debugging module was selected as a vehicle to develop computational thinking skills in graduate education students.

E-textiles were selected since research suggests that their affordances are appealing to a wider audience than the traditional methods of entering electronics (Buechley, 2010). McLuhan (1964) discussed how a technology's affordances might dominate its intended purpose. Some affordances of LilyPad Arduino are that they allow electrical connections without soldering and that they encourage the combination of art and science.

Developing computational skills in graduate students in education not only affects the graduate students themselves, but also the students whose education they will influence once their degree program is finished. Thus, it is an effective place to begin to address the gap of needed computational thinkers.

The debugging module was developed using design-based research principles, since relatively little research exists about developing computational thinking in university-level students using the LilyPad Arduino e-textile computing environment. This study took the design through a pilot test and a single iteration of the module. Further research will continue to refine and improve the design but was outside the scope of this dissertation. The first iteration is considered a proof of concept for the learning module.

Purpose of the Study

The purpose of this design-based research study was to create, test, and improve a series of e-textile activities that support the development of computational thinking in graduate education students.

Key Outcomes

Key outcomes sought were:

1. Evidence of computational thinking concepts, practices, and perspectives
2. Increased student confidence in computing abilities
3. Activities that appealed to non-computer science majors

Conceptual Focus

The design of the e-textile debugging learning module was guided by constructionist learning theory and sought to incorporate prior research about computational thinking. The purpose of the study was to create an e-textile module that allows non-computer science major students to explore aspects of debugging, which is one of the key components of computational thinking (Lapidot & Hazzan, 2005; Papert, 1980; Polin, 2012). An additional purpose was to explore methods that would allow for more diverse contributors to computer science. These conceptual areas are examined in Chapter 2.

Significance of the Study

The theoretical significance of this study is that it has the contributed to the understanding of what elements can be included in a learning module to allow students to experience computational thinking concepts, practices, and perspectives and to affect confidence in computational ability among graduate level university students. As more is known about how to encourage computational thinking, measures can be taken to address the current inequity and

economic issues associated with the underrepresentation of women and other minorities in computer science.

In addition to its theoretical significance, this design-based research study has added to the body of knowledge of how to leverage the affordances of e-textiles to encourage computational thinking. Practically, the study helps contradict the shrinking pipeline metaphor, showing that it is possible to intervene with university-aged students and not just middle-school ones. In addition, the study provided evidence that even a relatively short e-textile module could change attitudes and engage students in computational thinking.

Assumptions

There were several assumptions upon which this study was built. It was assumed that the doctoral students in Pepperdine's Learning Technologies program were representative of a larger graduate education student population. It was also assumed and shown that the students would write about their ongoing debugging experiences in the threaded discussions that were part of their class. Another assumption was that the hands-on debugging activities provided authentic experience with debugging and e-textiles.

Delimitations

This study introduced e-textiles using the particular computational thinking practice of debugging, but assessed a broad range of computational thinking concepts, practices, and perceptions. The participants in the study were students who were committed to learning technologies (they were doctoral students in a learning technologies program), but they had varying exposure to computational thinking and debugging. The learning module was implemented during a semester course, EDLT 740, which was a required part of the learning technologies doctoral program at Pepperdine University.

Variables

The variables of interest for this design-based research were computational thinking concepts, practices, and perspectives, interest in computational activities, confidence in ability to complete projects that depend upon computational thinking, and gender. Data about the variables was gathered in a variety of methods that are described in Chapter 3.

Timeline

A timeline for the major milestones of this research project is shown below. The learning module was part of a semester course that was offered in Pepperdine's Doctor of Education in Learning Technologies program. The course was offered in a blended format, with synchronous meetings online and two five-day face-to-face (F2F) sessions during the semester. The first iteration of the learning module took place in the Fall 2013 semester and is considered a proof-of-concept for the learning module.

- Designed debug projects March 2013
- Created parts list March 2013
- Obtained LilyPad Arduino materials May 2013
- Submitted Proposal June 2013
- Constructed debug samples May – August 2013
- Piloted debug samples July 2013
- Participated in the debugging forum September – December 2013
- First Iteration September – October 2013
- Preliminary Evaluation October 2013
- Data Analysis September 2013- June 2015

Organization of the Dissertation

This dissertation is organized into five chapters, followed by a list of references and appendices. This chapter, Chapter 1, introduced the study. It included the problem statement, purpose of the study, key outcomes, conceptual focus, significance of the study, assumptions, delimitations, variables, and timeline. Chapter 2 examines literature on computer science study, computational thinking, constructionism, LilyPad Arduino research, e-textiles, and the underrepresentation of women, which are background for this study. Chapter 3 covers the methodology of the study, elaborating on the purpose, study design, participants, investigative techniques, instrumentation, data collection, and learning module design. In Chapter 4, the data are examined for computational thinking concepts, practices, perspectives, and indications of an increase or decrease in confidence and appeal. The design of the learning module itself is also evaluated for future iterations. Chapter 5 discusses the suggested design changes based on the findings from the first iteration of the study and looks at wider implications for the study of computational thinking. Finally, future research directions are suggested.

Chapter Summary

This chapter introduced a design-based research study that implemented an e-textile learning module focused on debugging in order to provide graduate education students with a context-rich environment in which to learn and use computational thinking concepts, practices, and perspectives. In addition to the goal of developing computational thinking, the learning module was designed with the purpose of using e-textiles as a way to interest a more diverse group of students in computing.

To determine the effectiveness of the learning module, students were asked to answer questionnaires, post to a debugging forum, create a class Wiki, write a weekly project notebook,

design and create an Arduino project, and respond anonymously to a learning module evaluation questionnaire. Recorded interactions and student posts and projects were evaluated using The Computational Thinking Rubric for Examining Students' Project Work, built upon the Brennan & Resnick (2012) description of computational thinking concepts, practices, and perspectives.

A timeline for the major milestones of the study was given. Before the first iteration of the project was conducted in September 2013, a pilot study tested the instruments and hands-on materials with a small group of university students. This pilot study is described in Chapter 3.

Chapter 2: Review of the Literature

Background

The rate of technological innovation continues to increase from year to year, with most technology imagined, designed, and developed by people with computer backgrounds. In addition, full utilization of today's range of innovative products and software often requires an understanding of computational principles. However, while the need for computer scientists and people prepared in related fields continues to rise, the number of students electing computer science as a major is insufficient to meet the growing need. Intensifying the problem is the fact that the proportion of women in the United States selecting these college majors has actually declined, resulting in an even more marked underrepresentation of women in computer science.

Computational thinking is a term that was introduced by Seymour Papert in 1991 and reinvigorated by Jeannette Wing in 2006. (Papert & Harel, 1991; Wing, 2006). Wing (2006) describes computational thinking as "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (p. 33). Clearly, from Wing's description, computational thinking is more than merely computer programming. While there are still differences of opinion about what constitutes computational thinking, increasingly it is becoming a research focus for those who are trying to understand and address the need for more, and better prepared, computer scientists and a more computer literate general populace.

This dissertation elaborated on the design, implementation, and assessment of a computational thinking learning module for graduate education students utilizing e-textiles to focus on debugging, one of the key practices of computational thinking. The design of the module was guided by the theoretical precepts of constructionism.

The literature review presented in this chapter intertwines the following conceptual areas as background for the e-textile learning module that was designed in the study:

1) The current state of computer science study in the United States
2) The current state of computational thinking research
3) Constructionist learning theory
4) The place of debugging in computational thinking and constructionism
5) E-textiles and computational thinking
6) The underrepresentation of women in computer science

Computer Science Study

Predictions for the continued growth of computer science as a very-high paying employment area are strong (Lacey & Wright, 2009), which should make it an attractive choice for students beginning their post-secondary studies. Statistics show that the number of students selecting computer science as a major declined between 2001 and 2007, but the trend has reversed, and enrollments in the computer science programs in the United States have increased for the last several years (Zweben, 2012).

However, an important aspect to consider is that it appears that the current growth in enrollment in computer science is being constrained by administrative factors rather than student interest (Zweben, 2012). With interest in studying in the field higher than the availability of the programs, existing practices that continue the status quo will make efforts to affect change and increase diversity more difficult to accomplish.

There is speculation that the misconceptions about the narrowness of computer science impact "both the size and character of the body of students" attracted by the field (Lu & Fletcher,

2009, p. 1). Most students' current perceptions are that there isn't room for their individual interests within computer science.

Another significant factor that must be considered is that while overall enrollment in computer science (CS) is increasing, the percentage of CS majors who are women is decreasing. In 1986, 35% of computer science Bachelor's degrees went to women, but by 2009, only 18% of the graduates were women (National Science Board, 2012).

Computational Thinking

Computational thinking can be conceived as a combination of concepts, practices, and perspectives (Brennan & Resnick, 2012). These concepts, practices, and perspectives work together to aid the computational thinker in solving problems, usually with the assistance of some type of computer or computer application. As smart phones and tablet computers have become nearly essential tools, there has been an explosion of applications available to make tasks easier, allow global collaboration, and put diversion at one's fingertips.

Not only are the applications themselves widely available, but so also are software tools that can create the applications. AppInventor, Google Sites, and the Arduino platform are examples of these types of software tools. This availability of low-threshold tools for producing applications has contributed to changing most youth, as well as many others, from consumers to producers of media and applications. Evidence of this change can be seen in the rapid rise of the Maker Culture. This change has resulted in a large group of people who need computational thinking skills, most of whom are outside the traditional pathway for learning computational thinking, the computer science classroom.

Computational thinking is for everyone. Because of the strong presence of computerized solutions to daily tasks, computational thinking is not something that can only be

applied by computer scientists. In fact, computational thinking is not limited to interactions with computers; the same principles can be applied to solving problems and applying solutions that do not involve computers, at all (Wing, 2006). Wing's extremely inclusive view of what constitutes computational thinking is one of several aspects of the field that is still being debated, but computational thinking is widely regarded as a twenty-first century thinking skill (Einhorn, 2011; Grover & Pea, 2013; Phillips, 2008; Wing, 2006). Grover and Pea (2013) states that "those in possession of computational competencies will be better positioned to take advantage of a world with ubiquitous computing" (p. 40). Similarly, Barr and Stephenson (2011) discusses how recent leaps of innovation and imagination "drive the need for educated individuals who can bring the power of computing-supported problem solving to an expanded field of endeavors" (p. 49).

Concepts. Brennan and Resnick's (2012) breakdown of the components of computational thinking identify computational concepts as sequence, loop, parallelism, events, conditionals, operators, and data. These are the building blocks that allow humans to direct how computers act. They are realized differently in distinct programming languages, but the concepts are the same.

Practices. Brennan and Resnick (2012) describe computational practices as incremental and iterative, involving testing, debugging, reusing, remixing, abstracting, and modularizing. These practices, while extremely helpful in computer programming, are not limited to interaction with computers. This is perhaps why many say that computational thinking is generalizable beyond computer science. It is significant that Grover & Pea (2013) state that abstraction is computational thinking's keystone, and that it is what distinguishes it from other types of thinking.

Perspectives. The computational perspectives identified by Brennan and Resnick (2012) are expressing, connecting and questioning. Where the computational practices above have more general applicability than the concepts, the perspectives have even broader application. They are directly related to problem solving, in the most general sense, and the construction of understanding.

Dispositions and pre-dispositions. Barr and Stephenson (2011) describes dispositions and pre-dispositions to computational thinking. These include confidence in dealing with complexity, persistence, ability to handle ambiguity, ability to deal with open-ended problems, setting aside differences to work with others to achieve a common goal, and knowing one's strengths and weaknesses. These dispositions and pre-dispositions are critical to all types of problem solving, so it is evident that they would be influential in computational thinking, which is a specialized subset of generalized problem solving. While generally useful, these dispositions needed to be specifically encouraged and brought to the attention of the learners in the e-textile debugging module that was designed in the present research, so the participants could use self-talk and encouragement to overcome the difficulties that presented themselves. Viewing problems in this way opens learners to creative solutions. This is one of the areas that will be more fully addressed in the next iteration of the learning module as a way to help students avoid personalizing the set-backs they experience in debugging.

Field of computational thinking still evolving. Although computational thinking is related to several other disciplines, computational thinking, as a distinct field, is relatively new. As a result, there is still discussion of what does and does not constitute computational thinking, lack of agreement about how one goes about developing computational thinking, and no clear ways to measure and assess it. The e-textile learning module in this research study is part of a

course for learning technologies doctoral students that explores the development and assessment of computational thinking.

Some of the questions in the field of computational thinking that are still being argued are

- What are the elements of computational thinking?
- How generalizable is it?
- How does one assess computational thinking?

In a review of the current state of discourse on computational thinking, Grover and Pea (2013) discuss the ‘definitional confusion’ that is still an aspect of the field of computational thinking. The definition of computational thinking is still being debated, with leaders in the field proposing varying definitions highlighting various aspects of the subject. (Aho, 2012; Barr & Stephenson, 2011; Einhorn, 2011; Lee et al., 2011; Wing, 2006). Aho (2012) “considers computational thinking to be the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms” (p. 832), so the concept of finding an appropriate model has great significance for him. Lee et al. (2011) describes computational thinking as “a set of thinking skills, habits and approaches that are integral to solving complex problems using a computer and widely applied in the information society” (p. 32). Phillips (2008) summarizes computational thinking as “the integration of the power of human thinking with the capabilities of computers” (p. 2). While definitions of computational thinking vary, most include some aspect of debugging as an essential component (Polin, 2012).

Debugging, a large component of computational thinking. In his highly influential book, Mindstorms, Papert states, “Learning to be a master programmer is learning to become highly skilled at isolating and correcting ‘bugs,’ the parts that keep the program from working” (Papert, 1980, p. 23). Is his introduction to the second edition of Mindstorms, Papert further

stresses this point by asserting “the constant need for course corrections, which I call ‘debugging’ in this book, is the essence of intellectual activity” (Papert, 1993, p. xiii).

Papert is not alone is placing great significance on debugging skills. The CS community treats errors themselves matter-of-factly; bugs are viewed as just a common consequence of the creative process and an opportunity to gain deeper knowledge. Talented debuggers are valued on computer science projects. Lapidot & Hazzan (2005) discusses debugging’s exceptional educational value, highlighting the transferability of the problem-solving skills and attitudes towards errors. “We believe that learning to debug can change student’s negative attitudes towards errors and increase their awareness with respect to the importance of mistakes in the learning process” (Lapidot & Hazzan, 2005, p. 2). This belief aligns well with Papert’s statement that, “Surely ‘debugging’ strategies were developed by successful learners long before computers existed. But thinking about learning by analogy with developing a program is a powerful and accessible way to get started on becoming more articulate about one’s debugging strategies and more deliberate about improving them” (Papert, 1980, p. 23).

Computational thinking tools checklist. Repenning, Webb, and Ioannidou (2010) identifies conditions that should be fulfilled by computational thinking tools. The list consists of the following 6 conditions:

1) Has low threshold
2) Has high ceiling
3) Scaffolds flow
4) Enables transfer
5) Supports equity
6) Systemic and sustainable (Repenning, Webb, & Ioannidou, 2010, section 2)

E-textiles projects are low-threshold, because with an understanding of a simple circuit, projects can be designed and created; they are also high-ceiling because complex systems can be created by combining soft circuits, sensors, and the LilyPad Arduino processor and Arduino programming environment. The checklist item 'scaffolds flow' refers to the fact that the computational thinking tool should support the process of moving from the easy entry point to the sophisticated possibilities. Built-in help functions and abundant examples both within the Arduino environment and outside in the wider Maker community allow a beginner to understand and implement advanced applications of e-textiles. Knowledge of circuits gained through e-textile experience is transferrable to other projects using electricity. The coding experience using the Arduino environment is directly transferable to other C programming and indirectly transferable to programming in multiple other languages. E-textile projects support equity, because they combine the artistic area of textiles with the scientific application of electronics modules. The combination of the two can appeal to a broader range of people. Also, the relative newness of the field of e-textiles means that it has not yet been labeled as the prerogative of one gender or another. While inclusion of e-textile projects is not yet systemic, there are increasing opportunities for involvement in the local and online community, so that further learning is sustainable.

Constructionism

Because one of the goals of this research is to develop computational thinking, it is important to have the learning module's activities guided by learning theory. The learning theory that underlies these activities is constructionism. Constructionism is related to Piaget's psychological theory of constructivism but has several distinguishing tenets. Both constructivism and constructionism hold that knowledge structures must be built, but constructivism places

great emphasis on knowledge being built in a series of hierarchical stages. Constructionism, unlike constructivism, integrates elements of social learning theory (Papert & Harel, 1991).

In describing constructionism, Papert and Harel (1991) talks about two styles of thinking and learning. One is a style of learning where the learner likes to keep close to physical things and the other is a more abstract, distanced way of thinking. Constructionism finds that building and playing with items rooted in contemporary culture provide learning-rich activities (Papert & Harel, 1991). Kafai and Resnick (1996) describes constructionism as intertwining the construction of knowledge with the construction of personally meaningful artifacts (p. 1). Relationships with knowledge as well as representations of knowledge are important in constructionism.

While constructionism has many points in common with constructivism, it contrasts sharply with instructionism. Instructionism is one-directional and aimed at passing knowledge through telling, while constructionism emphasizes learners actively building knowledge structures through interaction with concrete objects.

As might be expected about a theory whose title focuses on building, constructionism isn’t viewed as static; ideas related to constructionism continue to be refined (Kafai & Resnick, 1996). The importance of community in constructing knowledge is an area that has become more prominent in recent years. “Constructionism suggests that learners are particularly likely to make new ideas when they are actively engaged in making some type of external artifact which they can reflect upon and share with others” (Kafai & Resnick, 1996, p. 1).

Concrete and abstract in constructionist terms. Concreteness and abstraction are important concepts in both constructionism and computational thinking. “Concepts that were hopelessly abstract at one time can become concrete for us if we get in the ‘right relationship’

with them" (Wilensky, 1991, p. 198). Having a concrete understanding of an abstract concept allows the learner to extrapolate and apply the idea to other situations. Papert (1980) refers to this when he discusses his experiences with gears and how his deep understanding of how they worked allowed him to understand Algebra. For Papert, gears were 'objects-to-think-with'. Papert explains, "My interest is in the process of invention of 'objects-to-think-with,' objects in which there is an intersection of cultural presence, embedded knowledge, and the possibility for personal identification" (Papert, 1980, p. 11). For an object to be available to think with requires that learners have the opportunity to experiment and make multiple connections with it.

Abstracting is one of the key practices of computational thinking, yet experience with tangible, concrete objects is a goal of constructionism. While at first these ideas may seem to be contradictory, Papert's description of 'objects-to-think-with' is an example of abstracting. With sufficient concrete interaction and experimentation, the object becomes available for mental manipulation, which is a type of abstraction of essential qualities. Abstraction can be seen as reducing information and detail to focus on concepts relevant to understanding and solving problems (Grover & Pea, 2013). Another view of abstraction is that is the process of generalizing the common core or essence (Kramer, 2007). The ability to apply the generalized essence of a solution to other concrete problems is one reason that abstraction is a valued practice in computational thinking.

Blackwell, Church, and Green (2008) gives a contrasting view of abstraction. The article describes the abstract as an enemy. In the process of abstracting to remove the inconvenient complexities, the resulting clean model may not fit the human necessity. The constructionist view that values experience with the concrete can mitigate the tendency to over-abstract when creating a particular solution. Lee et al. (2011) mentions both abstracting at multiple levels and

analyzing the appropriateness of the abstractions as part of the description of computational thinking.

LilyPad Arduino Workshops and Research

Searching for LilyPad Arduino workshops brings up offerings from across the United States and the world. Non-profits, hobby groups, and community organizations offer the workshops. Some of the workshops are specifically aimed at encouraging more female participation in STEM fields, while others are aimed at integrating aesthetics and technology. The workshops target a range of age groups from children to adults. The National Science Foundation (NSF) is currently funding eight projects that use e-textiles (National Science Foundation).

E-Textiles and Computational Thinking

In the review of the state of computational thinking, Grover and Pea (2013) mentions e-textiles as a computational environment that can provide computing in context and that can bridge the gender gap in the computing field. E-textiles have the ‘low floor, high ceiling’ characteristic that makes an ideal environment for introducing computational thinking. As discussed earlier in this chapter, low floor refers to the fact that it is easy for a beginner to become involved with e-textiles; high ceiling refers to the fact that using the environment and tools it is possible to create extremely sophisticated projects and solutions to problems.

Underrepresentation of Women in Computer Science

According to 2009 U. S. Department of Commerce statistics, only 25% of the college-educated workers in computer fields were women (Beede et al., 2011). With such lopsided employment figures, there is obvious inequity; this inequity has an economic impact on the individual women who are capable of, but not working in, these fields, because computing

careers are higher paying and have better job security than most other careers (Hill, Corbett, & St. Rose, 2010). In addition to the economic impact on the individual women and their families, there is an economic impact on businesses and institutions that are missing women's input. With fewer women in computer science, there are fewer creative types of projects undertaken by research institutions because the pool of contributors is less diverse (Cox, 1991; Johansson, 2006; Page, 2008). This affects the success of the institutions themselves and also impacts the overall economy of the country. "Diversity brings better decision-making, better innovation outcomes, and better team dynamics" (Simard, n.d., paragraph 2).

Achieving more equitable female participation in science, technology, engineering and math (STEM) subjects has been an increasing concern over the past 30 years, but inequities still persist.

Progress has been made in raising female participation in some STEM subjects, but large gaps still remain. Great strides have been made in the biologic sciences, for example, but female representation in computer science is still quite low. US Department of Commerce data shows that only 25% of the computer scientists in the U.S. were women in 2009 (Beede et al., 2011).

In the first push to counteract the inequitable representation of females in computer science, differences in preparation were hypothesized to be the cause. Thus, efforts were made to strengthen female students' preparation in math and science at the secondary level so that they would have the option of choosing STEM majors in college. Statistics show that the goal of equal preparation for male and female students in math and science has been reached, with nearly equal numbers of males and females obtaining the prerequisites for STEM studies at the college level. Yet, in most cases, the number of high school young women actually choosing

STEM studies at the university level remains much lower than the number of young men (Hill et al., 2010).

The underrepresentation of women in the STEM fields affects not only the opportunities for individual women, but also the kinds and quality of the materials and outcomes produced by research teams in business and in educational settings. The absence of women's voices mean that the field is forfeiting some of the creative solutions to the current problems that are being investigated, and also that the input on what kinds of problems should be investigated is limited. "If technology and science are going to go forward with wild creativity, we need the brilliance of more than a narrow group of people" (Thom, 2001, p. 108). People that come from diverse races, classes, and genders ask different questions, which can lead to different projects, emphases, and solutions. More women in the field would likely impact the projects that were selected, because women seem to be drawn to projects that they view as benefiting society (Hill et al., 2010; Margolis, Fisher, & Miller, 1999-2000).

The existence of an extreme imbalance of representation within an organization is an instance of bias, whether or not overt prejudice caused the imbalance. It is important to be aware of such imbalances and to increase opportunities for the underrepresented group, for ethical reasons as well as for the benefits that increased representation will bring. The underrepresentation of women in computer science has been called alarming not only because of the possibility it presents of unethical treatment, but also because of the detrimental impact that discouraging potential contributors has on the field (Pearl et al., 2002).

However, there is another ethical issue that is related to addressing the underrepresentation of women in STEM fields. Care needs to be taken that the actions that are initiated to increase the representation of women do not treat men unethically. The backlash

against affirmative action is an example of people feeling that support aimed at rectifying past prejudice was not ethical. Participation in this study's learning module in e-textiles was by students of both genders.

Beyond better preparation. Better math and science preparation for female secondary students satisfies the goal of eliminating missing prerequisites as blocks to female students' participation in STEM fields at the college level. However, the continued low participation rates and higher major-change rates by women in STEM majors, despite equal preparation, point to other problems.

Stereotypes and expectations. Geeky stereotypes of computer scientists are one of the reasons women do not choose the field (Margolis & Fisher, 2006). Also, the existing imbalance of significantly fewer women in the field makes it seem that computer science is not an appropriate choice of career for a young woman. Another factor is that women's lower confidence makes them vulnerable to experiencing discouragement as a result of the setbacks that most new students face (Fisher, Margolis, & Miller, 1997; Hill et al., 2010).

Effects of lower confidence. Self-confidence has profound effects on women's choices of study and career. Research shows that girls who have similar abilities to boys rate themselves as being less-skilled, indicating less confidence in the same abilities. Not only do women assess their abilities more critically, they also hold themselves to higher standards. These higher standards make them feel even less adequate. A belief that men and women are equally capable of success in computer science can lead to increased female selection of these as career choices (Hill et al., 2010).

Confidence not only affects women's selection of computer science as a career, it also affects retention of women in the field. While the following quote specifically mentions

engineering, it can also be applied to women's confidence in computer science. "If women develop less confidence about their abilities to be successful professionals and express more ambiguity about their fit or comfort within the discipline, then women will remain in engineering at lower rates than men" (Cech, Rubineau, Silbey, & Seron, 2011, p. 660).

Culture. Making sure that women had equal preparation to pursue studies in STEM fields was a logical step to increase the number of women in the fields, but equal preparation has not solved the inequity of representation. The underlying culture that continues to view computer science as men's work, even if this is an implicit rather than an overt bias, still asserts a strong influence. Cultural interpretations are one of the most powerful forms of social stability (Bruner, 1990). For a young woman to diverge from the cultural expectation that computer science is not for her will require that she find a personal narrative that can propel her to break free from the gravitational pull of cultural expectations.

Pipeline metaphor. Many of the interventions and attempts to address the underrepresentation of women in the computing fields are aimed at middle-school aged girls. This is partly due to the perception of preparation in computer science as a pipeline (Camp, 1997; Hamner, Lauwers, Bernstein, Nourbakhsh, & DiSalvo, 2008). The pipeline metaphor emphasizes that there is a fixed entry point for people coming into computer science and long, solid walls that keep everyone else out. Using the pipeline metaphor, the only ways to increase the number of women coming out as computer scientists is to increase the number going in at the beginning of the pipeline (grade school and middle school), and to plug any leaks. The problem with this metaphor is that it makes older girls and young women seem as though they are already beyond the reach of any interventions, so, by extension, it also seems wasteful to use resources trying to reach them.

Rather than a pipeline, an alternative metaphor would be to consider preparation in computer science as a river with various tributaries. This would encourage the notion that there are many possible entry points for entering the computing fields and therefore would not exclude efforts toward reaching a wider range of students. Jesse (2006) discusses the poverty of the pipeline metaphor, noting that it is misleading because it makes the preparation process seem smooth and predictable. She also compares and contrasts the pipeline to a river, but rather than pointing at the flow into the river she focuses on a river's winding nature and the presence of rocks and rapids. One of Jesse's main issues with the pipeline metaphor is that it makes the problem seem that it stems from a lack of supply rather than a problem of creating greater demand. While an incorrect metaphor may not seem like a critical matter, Eagly and Carli (2007) emphasizes, when advocating for an alternative to the glass ceiling metaphor, "Metaphors matter because they are part of the storytelling that can compel change" (p. 3).

Experience with tangible projects. Tinkering can be defined as playful experimentation. Female students tend to have less tinkering experience with electronics and computers, which is one of the factors leading to their feelings of being less prepared than male classmates. Educational research shows that tinkering is a strategy adopted more by males than females, and that a playful approach to learning can increase both motivation to learn and ability (Beckwith et al., 2006). A study designed to examine the effects of two treatments on tinkering, debugging and self-efficacy of males and females on a particular task concluded, "although they tinkered less, females' tinkering was very effective: it was significantly tied to understanding and to successfully testing and debugging, regardless of treatment" (Beckwith et al., 2006, p. 239). Encouraging a tinkering attitude towards debugging can provide increased insight and understanding, especially for female students.

Connections

Review of the literature has shown that there are underlying connections between computational thinking, constructionism, and e-textiles. It is not simply fortuitous that there is a natural fit of e-textile technology with constructionist learning theory and the goal of developing computational thinking that is the purpose of the present study.

First there is the connection between constructionism and computational thinking. While Wing (2006) caused a resurgence of interest, it was originally Papert who coined the term computational thinking. Papert, himself, is one of the connections between constructionism and computational thinking. Papert's ties to the MIT Media lab can still be seen in the influence of constructionist style learning projects that the lab pursues.

In describing how e-textiles can be used to investigate engagement and aesthetics in computer science education, Buechley, Eisenberg, Catchen, & Crocket (2008) states, "We have been strongly influenced by the tradition of constructionism, which postulates that people are more likely to become engaged in an activity and learn things from it when they are active and creative participants" (p. 428). Buechley has been one of the driving forces in creating and bringing the LilyPad technology to the public and directs MIT Media Lab's High-Low Tech research group.

Mitch Resnick, head of MIT Media Lab's Lifelong Kindergarten group, has been honored as one of the most influential people affecting the advancement of technology for his work with Lego MindStorms and Scratch. Resnick is the LEGO Papert Professor of Learning Research and is one of the editors of *Constructionism in Practice* (1996). In the following quote Resnick (2007) explains the creative thinking spiral.

> In this process, people imagine what they want to do, create a project based on their ideas, play with their creations, share their ideas and creations with others, reflect on their

> experiences – all of which leads them to imagine new ideas and new projects. As students go through this process, over and over, they learn to develop their own ideas, try them out, test the boundaries, experiment with alternatives, get input from others, and generate new ideas based on their experiences. (p. 18)

Strong ties to Papert's constuctionist ideas can be seen in Resnick's creative thinking spiral, as well as overlapping ideas with Brennan & Resnick's (2012) description of computational thinking concepts, practices, and perspectives.

Chapter Summary

This literature review has examined computer science (CS) study, computational thinking concepts, practices, and perspectives, constructionism, and the underrepresentation of women in CS as the background for the design-research study. Chapter 3 focuses on the methodology that will be used to conduct the study.

Chapter 3: Methodology

Purpose

The purpose of this design-based research study was to create, test, and improve a series of e-textile activities that support the development of computational thinking in graduate education students.

Key Outcomes

Key outcomes sought were:

1. Evidence of computational thinking concepts, practices, and perspectives
2. Increased student confidence in computing abilities
3. Activities that appealed to non-computer science majors

Study Design

Design-based research has been described as a series of approaches that examine learning in naturalistic contexts that are designed and systematically changed by the researcher (Barab & Squire, 2004). Since the field of computational thinking is still evolving and e-textiles are a relatively new technology, design-based research methodologies were used to add to the understanding of what contributes to the development of computational thinking using e-textiles and how to assess its presence. Design-based research involves multiple iterations, as lessons learned from prior instances are incorporated to strengthen the design. The multiple iterations can be a lengthy process, especially if the design is tied to an event that does not occur frequently. Since the EDLT740 course where the learning module was introduced occurs only once per year, this study took the design through a pilot test and a single iteration of the learning module. Further research will continue to refine and improve the design but will be outside the

scope of this dissertation. The first iteration is considered a proof of concept for the learning module.

Learning module overview. The learning module began with hands-on exposure to the electronic principles of polarity, circuits, and conductivity. These concepts were introduced during the EDLT740 course's weekly online synchronous sessions. Originally, the foundational introductory activities were planned to be conducted in-person, but the course's face-to-face (F2F) meetings were scheduled too far into the course to wait until that time. Therefore, the introduction was done online and the debugging activities that occurred at the course's F2F meeting in October were conducted without further introduction to the topics. By the time they experienced the debugging activities, students had already begun creating their own projects. Appendix A has a copy of the learning module guide, which gives a more complete description of the activities as ideally planned.

Following the F2F debugging exercises, the students continued to work on a larger Arduino project of their own design over the next four weeks. During this time, while they were working individually on their projects, the students posted about their bugs and debugging experiences in a threaded discussion with their classmates and professors. They also contributed to a class Wiki where they shared principles and techniques that could help others with their Arduino experiences. In addition, the students maintained a project notebook or blog, where they logged their progress on their projects. At the end of the learning module, at the second F2F meeting, the students completed a final debugging design scenario, and completed a questionnaire about their experiences in the learning module.

Design-based research principles. Wang & Hannafin (2005) identifies nine principles that should be used in design-based research to "generate practical, credible, and contextual

design theories" (p. 18). These nine principles guided this study design. How this study design maps to the principles will be elaborated in the following sections.

Principle 1: Support design with research from the outset. The design for this research was based upon constructionist learning theory, which finds that there is a close connection between constructing knowledge and constructing personally meaningful articles (Kafai & Resnick, 1996). The computational thinking concepts, practices, and perspectives that were encouraged and assessed are identified by Brennan & Resnick (2012). The design of the debugging projects was guided by the identification of the six engineering and coding challenges identified in Fields, Kafai, Searle, & Min (2012).

Principle 2: Set practical goals and develop an initial plan. A proposal presenting the initial plan for the research was informed by input from the dissertation committee. The scope of the design was manageable, since it dealt with one university course to which the researcher had access. The research objectives represent the setting of the goals.

Principle 3: Conduct research in representative real-world settings. The research was conducted in a graduate education course, which is the setting for which the design is being created.

Principle 4: Collaborate closely with participants. There was ongoing collaboration with the course participants through the debugging forum as well as at the two F2F meetings. The researcher participated in all the synchronized online sessions of the course and was added to the cadre's facebook group where conversation about the projects also continued. Participant suggestions were sought through debriefings and questionnaires and were incorporated into the evolving design.

Principle 5: Implement research methods systematically, and purposefully. The data collection and analysis methods described in following sections were the purposeful and systematic methods that captured data to inform the design.

Principle 6: Analyze data immediately, continuously, and retrospectively. Questionnaire data obtained at the first F2F meeting was analyzed for insight into the experience and attitudes of the participants. This insight guided the direction of further interaction with the participants. Recordings of the participant pairs' interaction with the first set of debug materials was reviewed, so that areas for potential improvement could be implemented before the final design scenario at the second F2F meeting. At the study's end, all data was re-examined for further insight.

Principle 7: Refine designs continually. As described in the preceding section, not only was data reviewed immediately after it was collected, design changes that were informed by the collected data were implemented into the design during the research study. No substantive changes to the research methods occurred, however.

Principle 8: Document contextual influences with design principles. Wang and Hannafin (2005) states, "In the design principles section, principles that transcend the local setting are presented with relevant contextual information; warnings and guidance for appropriate application of these principles may also be provided" (p. 19). In the dissertation, these are addressed in the discussion in Chapter 5.

Principle 9: Validate the generalizability of the design. This iteration of the design was created as a 'Proof-of-Concept'. The design is described in sufficient detail that it could be replicated at other locations. Those implementations are outside the scope of this dissertation,

however. Unusual factors particular to this implementation have been highlighted in Chapter 5 so that others implementing the design can address them in an appropriate way.

Participants

A cohort group of students in Pepperdine University's Doctor of Education in Learning Technologies program were part of an EDLT740 course that introduced students to computational thinking using the low-threshold activities AppInventor and LilyPad Arduino. The first half of the course focused on the LilyPad Arduino activities, and the second half of the course focused on AppInventor activities. The course was taught for the first time, beginning September 2013. The students had the opportunity to be part of the research study but could also choose not to participate. All 18 of the students chose to participate in the study. Study participants responded to additional questionnaires and created project notebooks or blogs, in addition to the normal course requirements. As graduate education students, the participants in this study will have future opportunities to affect the learning and opportunities of other students with regard to computational thinking. This means that any gains the participants made in computational thinking could have exponential effects.

Descriptive data about the graduate students who participated in the learning module is presented below. The data was collected through student responses to the Computer Experience Survey.

Appendix B contains a copy of the Computer Experience Survey to which the EDLT740 students responded. The survey was administered once, at the beginning of the learning module. In addition to asking about prior computer experience, the survey had questions regarding prior sewing and e-textile experiences. Seventeen of the 18 students responded to the survey.

Table 1

Reported Prior Experience, Non-Agreement (1) to Agreement (6)

Gender	Never Programmed	Never Sewn	Never Completed E-Textile Project	Never Completed Electronics Project
F (10)	4.1	2.5	5.7	3.9
M (7)	2.4	3.7	5.9	2.3
Comb	3.4	3.0	5.8	3.3

The Reported Prior Experience table, Table 1, shows that the average agreement score for the female students was higher than the average agreement score for the male students with regard to the statement that said they had never programmed; in other words, the average female score showed less programming experience. The average agreement score of the male students to the statement that they had never sewn was higher than the females (average male score showed less sewing experience), but the gap between the gender averages was lower than the gap about programming. There was a high agreement score for both male and female students about never completing an e-textile project. The agreement scores for both male and female students show that there has been some experience with electronics projects, although the male students' average score shows less agreement, meaning that they had completed prior electronics projects.

The Computer Experience Survey also asked about how comfortable students felt about some factors likely to be related to e-textiles projects and computational thinking. The following chart, Table 2, displays the results for several of these questions. The chart shows averages, by gender, and then for the group.

Table 2

Computer Experience, Non-Agreement (1) to Agreement (6)

Gender	Have Artistic Side	Comfortable Programming in Several Languages	Learn New Programming Languages Easily	Good at Debugging
F (10)	3.7	1.7	2.2	3.0
M (7)	5.0	2.4	3.1	4.7
Comb	4.1	2	2.6	3.7

As shown in the Computer Experience table, the male students' level of agreement with these four statements is higher than the female students. Male and female students have average agreement scores above the midpoint with regard to the statement that they have an artistic side, and average agreement scores below the midpoint with regard to the statements that they are comfortable programming in several languages and that they learn new programming languages easily. The male students in the group have an average agreement score above the midpoint about whether they are good at debugging, while the female students have an average agreement score below the midpoint regarding being good at debugging.

Site Description

The site for the F2F parts of this design study was Pepperdine University's West LA campus in Southern California, during the students' normal F2F meetings. These meetings occurred for five days, twice a semester. These two meetings were the only in-person meetings for the course participants. The students met in synchronous online sessions weekly and posted to course forums throughout the week.

Prior to arriving at the F2F meeting students installed the Arduino programming platform on their personal laptops, which they were required to bring to all their F2F meetings. Following their learning module experiences at the F2F meeting, the students carried out further project

investigations and continued posting to the forum, Wiki, and project notebook from their homes across the United States and in other parts of the world.

Investigative Techniques

The main investigative technique for this design-based research study was to build an e-textile learning module, based upon the theoretical principles of constructionism, that incorporated current research into computational thinking, and then use multiple measures to see if computational thinking concepts, practices, and perspectives were evident. Student feedback was gathered to determine impact and plan future improvements for the learning module. Module elements are described in this section. A learning module guidebook is available for review in Appendix A. In addition to the data gathered for from the debugging exercises, the students' work on their project designs and implementations and final reflection papers were also examined for evidence of computational thinking.

Hands-on Protosnap experience. The learning module introduced debugging activities using LilyPad Arduino electronics components and the Arduino programming environment. The LilyPad Protosnap board allowed students to explore electronics and programming principles quickly before putting the principles into practice with Arduino projects. The Protosnap board has prewired circuits that allow for easy experimentation with the electronic sensors and components, but the board can be broken apart for access to the individual components. A picture of the LilyPad Protosnap board is shown in Figure 1.

One of the students found an example Arduino program online that demonstrated all of Protosnap board sensors. He posted this in the threaded discussions, and it became the basis for exploration of the Protosnap board and discussions about programming.

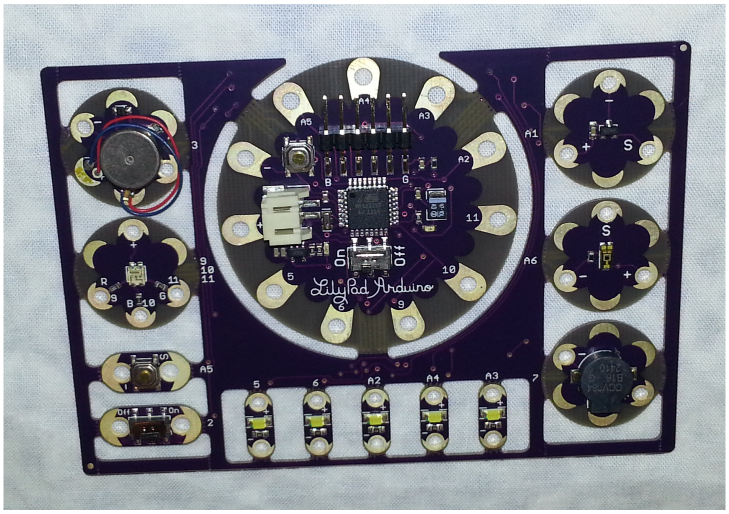


Figure 1. LilyPad Protosnap board.

Demonstrate bug-solving tools. After the online introduction to general and soft circuits, the students were introduced to using the serial port display to show which steps in the program had been performed and in what order. At the F2F meetings, using alligator clips to check the behavior of a circuit was discussed with various students.

Initial debugging design scenarios in pairs. Because the students had been working on their own Arduino projects for several weeks, the researcher assumed that the students would feel comfortable with being able to explore and manipulate e-textile objects, so no further introduction occurred before pairs of students were given six prepared e-textile objects each with a specific bug that they were asked to locate. Feedback showing that several students felt they needed further instruction is discussed in Chapter 4. EDLT740 students were part of a cohort that had already formed strong bonds, so they were allowed to form their own pairs. There were an even number of students, otherwise, one group would have been formed with three members.

The six e-textile objects were planned to contain a single bug of the six types identified in Fields et al. (2012). Fields et al. (2012) identified their bug types through interviews and analysis of student projects from several workshops. In this study's implementation of the

debugging exercises, the fifth e-textile object actually turned out to have several unanticipated bugs in addition to the planned bug. This will be further discussed in the Chapter 4.

The original design of the learning module was to spread the soft circuit introduction and debugging activities over three days. Due to the needs of other courses, the F2F schedule allowed only one day for officially meeting the EDLT740 students, however, so all debug activities were done in one day. As mentioned previously, the soft circuit introduction was carried out several weeks before in the online environment. The debug tasks were presented from a puzzle perspective and the pairs worked together in an informal group atmosphere. The researcher observed the debugging process that the pairs engaged in and circulated and answered students' questions. Students gave permission for their conversations to be recorded as they worked on the bugs. Livescribe pens were used to record the conversations for further analysis. The students themselves started the recordings and paused them, at times.

Schematics showing each type of bug, and a possible correction, follow. The actual solutions that the students created took a variety of forms.

Engineering challenge: Short circuit. The first of the following schematics, in the left side of Figure 2, shows a short circuit at point A, where a positive and a negative path cross. Figure 2, right side, shows one way to correct the short circuit. Note that each of the ends of the LED's has polarity. The polarity is marked on the LED, but it is very small. The end with the two dots is negative. Traditionally, engineers use red to represent positive and black to represent negative, also known as ground.

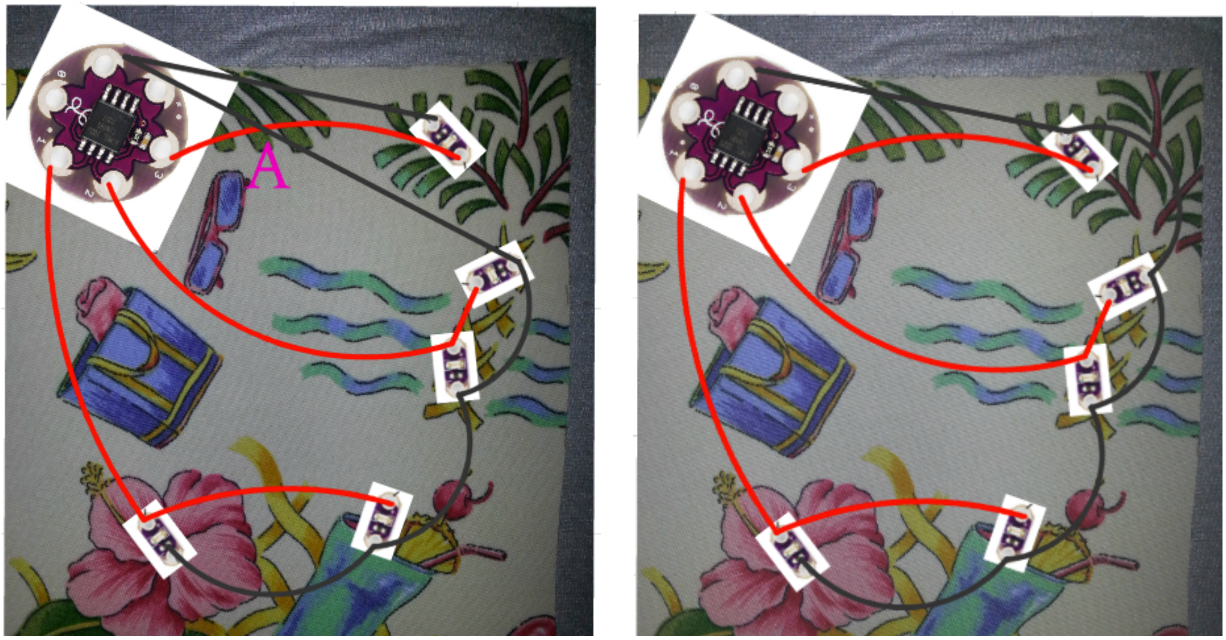


Figure 2. Short circuit error. (Left) Short circuit bug corrected. (Right)

Engineering challenge: Electronic topology. An example of an electronic topology bug would be when a circuit is in parallel, rather than independent, so behavior cannot be controlled independently. This type of bug is shown in the first schematic (Figure 3, left side), with LED's at point B.

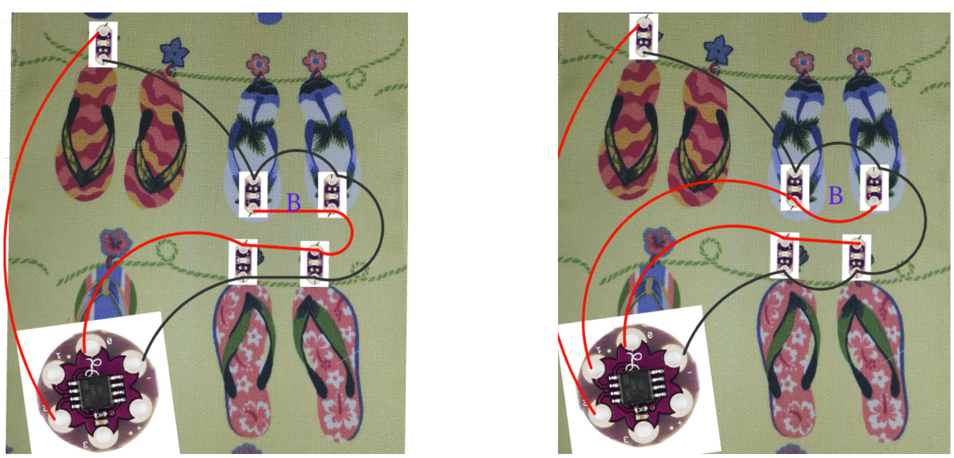


Figure 3. Electronic topology bug. (Left) Electronic topology bug corrected. (Right)

Figure 3, right side, shows how this topological error could be corrected to make the LED at point B have an independent circuit, and therefore allow an independent behavior.

Engineering challenge: Polarization. When positives and negatives are connected incorrectly, polarization problems occur. The majority of the circuits in e-textile projects use parallel circuits. In a parallel circuit, positives should be connected to positives and negatives to negatives. In the following schematic, Figure 4, left, a path connects a negative to a positive at point B. Figure 4, right, shows a possible correction by flipping the direction of the LED at point B.

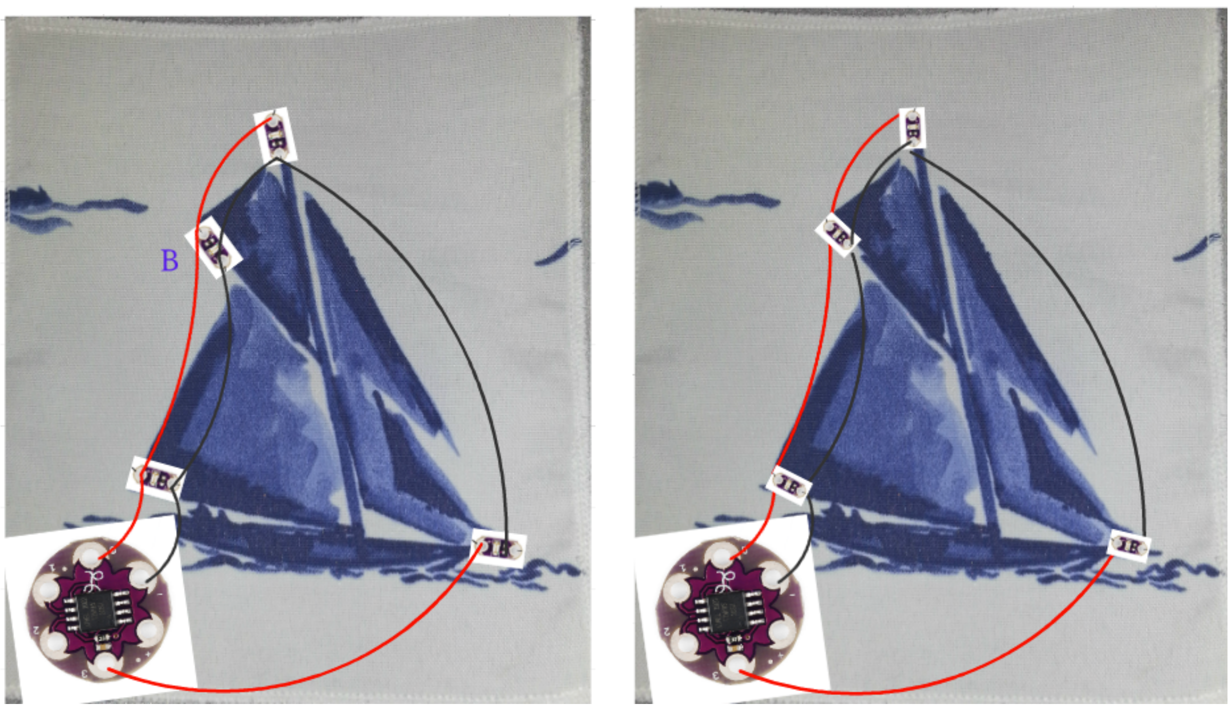


Figure 4. Polarization bug. (Left) Polarization bug corrected. (Right).

In contrast to the engineering challenges shown in the prior examples, the following bug types have errors in the program code. It is this connection between physical object and computer code that makes e-textiles exciting, but it can also make e-textiles more challenging to debug. Note that the code in the following examples used Arduino Open Source Example Programs as a starting point.

Coding challenge: Constant versus variable pins. The schematic for the e-textile project is shown for context. LilyPad controllers have some pins that are constant, and others that can be varied within the code. For example, in the LilyPad controller below, the + and – pins are constant, but pins 2, 3, 4, 5, and 6 are variable. If a variable pin needs to be positive or negative, it should be set to the proper value in the set-up code. The bug in the code is that pin 10 needed to be set to positive, but was set to negative in line 25 (See Figure 6).

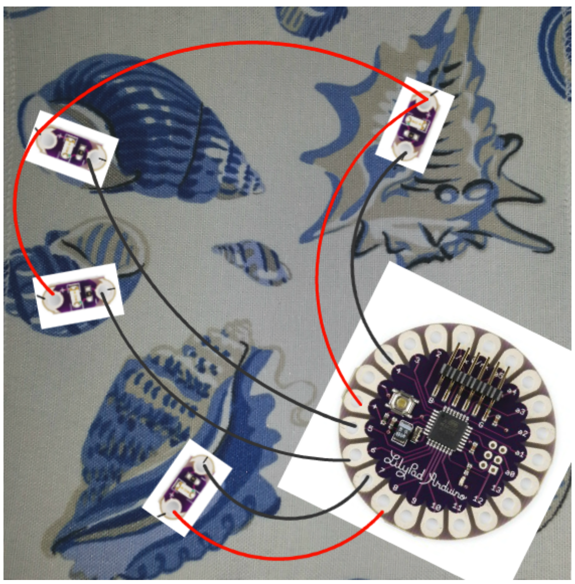


Figure 5. Schematic to accompany code in Figure *6*.

```
1  /*
2  New Bug 4, Constant versus Variable Pins   BUGGY
3  Victoria Kim Dissertation
4
5  Desired Action: Starting with spiked shell, turn shell lights on and off,
6  individually, in a counter clockwise direction.
7  */
8
9  // Name the led pins on the Arduino.
10     int led19 = 19 ;  //  spikey shell
11     int led5 = 5 ;    //  smooth shell
12     int led6 = 6 ;    //  little shell
13     int led9 = 9 ;    //  rough shell
14     int led10 = 10 ;  //  rough shell 2nd conn
15     int waitTime = 1000;
16
```

```
// setup runs once
void setup() {
  // initialize the digital pins as an output.
  pinMode(led19, OUTPUT);
  pinMode(led5, OUTPUT);
  pinMode(led6, OUTPUT);
  pinMode(led9, OUTPUT);
  pinMode(led10, OUTPUT);
  digitalWrite(led10, LOW);    //set pin to negative

  digitalWrite(led19, HIGH);   //start with pin off
  digitalWrite(led5, HIGH);    //start with pin off
  digitalWrite(led6, HIGH);    //start with pin off
  digitalWrite(led9, HIGH);    //start with pin off
}

// Loop routine (continually repeats)
void loop() {
  digitalWrite(led19, LOW);   // turn spikey on
  delay(2*waitTime);          // wait 2 second
  digitalWrite(led19, HIGH);  // turn spikey off
  delay(2*waitTime);          // wait 2 second

  digitalWrite(led5, LOW);    // turn smoothy on
  delay(2*waitTime);          // wait 2 second
  digitalWrite(led5, HIGH);   // turn smoothy off
  delay(2*waitTime);          // wait 2 second

  digitalWrite(led6, LOW);    // turn baby on
  delay(2*waitTime);          // wait 2 second
  digitalWrite(led6, HIGH);   // turn baby off
  delay(2*waitTime);          // wait 2 second

  digitalWrite(led9, LOW);    // turn roughy on
  delay(2*waitTime);          // wait 2 second
  digitalWrite(led9, HIGH);   // turn roughy off
  delay(2*waitTime);          // wait 2 second
}
```

Figure 6. Code demonstrating error in using variable pin.

The code in Figure 7 shows a possible correction of the coding error. The correction in the code is marked with a ** Correction comment.

```
/*
 New Bug 4 CORRECTED, Constant versus Variable Pins
 Victoria Kim Dissertation

 Desired Action: Starting with spiked shell, turn shell lights on and off,
 individually, in a counter clockwise direction.
 */

// Name the led pins on the Arduino.
int led19 = 19 ;   //   spikey shell
int led5 = 5 ;     //   smooth shell
int led6 = 6 ;     //   little shell
int led9 = 9 ;     //   rough shell
int led10 = 10;     // create a negative pin
int waitTime = 1000;

// setup runs once
void setup() {
  // initialize the digital pins as an output.
  pinMode(led19, OUTPUT);
  pinMode(led5, OUTPUT);
  pinMode(led6, OUTPUT);
  pinMode(led9, OUTPUT);
  pinMode(led10, OUTPUT);

  digitalWrite(led10, HIGH);   //set pin to positive   ** Correction

  digitalWrite(led19, HIGH);   //start with pin off
  digitalWrite(led5, HIGH);    //start with pin off
  digitalWrite(led6, HIGH);    //start with pin off
  digitalWrite(led9, HIGH);    //start with pin off

}

// Loop routine (continually repeats)
void loop() {
  digitalWrite(led19, LOW);   // turn spikey on
  delay(2*waitTime);         // wait 2 second
  digitalWrite(led19, HIGH);  // turn spikey off
  delay(2*waitTime);         // wait 2 second

  digitalWrite(led5, LOW);    // turn smoothy on
  delay(2*waitTime);         // wait 2 second
  digitalWrite(led5, HIGH);   // turn smoothy off
  delay(2*waitTime);         // wait 2 second

  digitalWrite(led6, LOW);    // turn baby on
  delay(2*waitTime);         // wait 2 second
```

```
digitalWrite(led6, HIGH);   // turn baby off
delay(2*waitTime);          // wait 2 second
digitalWrite(led9, LOW);    // turn roughy on
delay(2*waitTime);          // wait 2 second
digitalWrite(led9, HIGH);   // turn roughy off
delay(2*waitTime);          // wait 2 second
}
```

Figure 7. Bug 4, corrected code.

Coding challenge: Control flow. Fields et al. (2012) describes the control flow challenge as coding one end of an LED to a constant value and the other end to be variable. The schematic (Figure 8) for this coding challenge has one LED that has both ends tied to variable pins; because of this, its coding needs to be different than the coding for the other LEDs. In the following program code (Figure 9) the error is that pin 16 needs to be set to HIGH in the initial setup for the LED to operate properly.

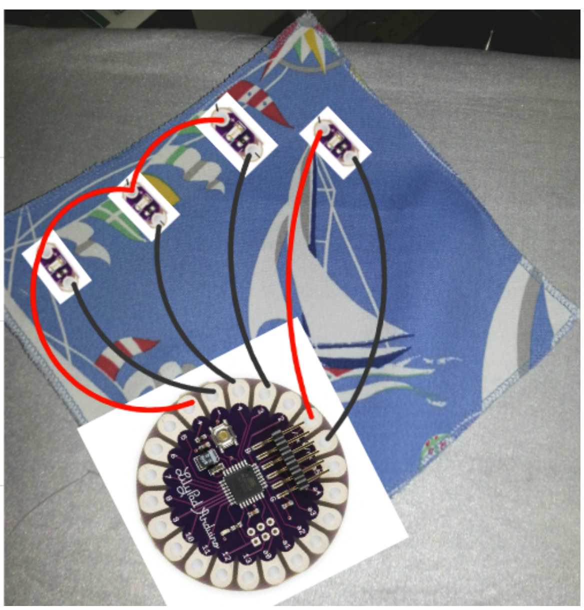


Figure 8. Schematic to accompany code in Figure 9.

```
/*
 New Bug 5, Control Flow     BUGGY
```

```
 Victoria Kim Dissertation

 Desired Actions: Bottom flag (not on boat) stays on,
                  Middle flag (not on boat) blinks quickly,
                  Top flag (not on boat) blinks slowly,
                  Light on mast blinks twice quickly, then pauses, then repeats.
 */

// Name the led pins on the Arduino.
int led11 = 11 ;
int led16 = 16 ;
int led17 = 17 ;
int led19 = 19 ;
int waitTime = 1000;
int cycleCnt = 0;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pins as output.
  pinMode(led11, OUTPUT);  //mast
  pinMode(led16, OUTPUT);  //mast
  pinMode(led17, OUTPUT);  //top
  pinMode(led19, OUTPUT);  //middle

  digitalWrite(led17, LOW);   // turn the LED on
  digitalWrite(led19, LOW);   // turn the LED on
  digitalWrite(led11, LOW);   // turn the LED on

  delay(waitTime/2);          // wait 1/2 second

}

// loop routine
void loop() {

  digitalWrite(led19, HIGH);  // turn the LED off
  digitalWrite(led11, HIGH);  // turn the LED off

  // led17 is on slower cycle
  // fmod divides by the second number and looks at the remainder
  if (fmod(cycleCnt,2) == 0)
  {
    digitalWrite(led17, HIGH);   // turn the LED off
  }

  delay(waitTime/2);

  digitalWrite(led19, LOW);  // turn the LED on
```

```
// slower blink
if (fmod(cycleCnt,2) == 0)
{
digitalWrite(led17, LOW);   // turn the LED on
}

//Every third time, the mast LED will not come on
if (fmod(cycleCnt,3) != 0)
{
  digitalWrite(led11, LOW);
}

delay(waitTime/2);

cycleCnt = cycleCnt + 1;

}
```

Figure 9. Code illustrating control flow error.

The code in Figure 10 is a possible correction of the error.

```
/*
 New Bug 5, Control Flow     CORRECTED
 Victoria Kim Dissertation

 Desired Actions: Bottom flag (not on boat) stays on,
                  Middle flag (not on boat) blinks quickly,
                  Top flag (not on boat) blinks slowly,
                  Light on mast blinks twice quickly, then pauses, then repeats.
 */

// Name the led pins on the Arduino.
int led11 = 11 ;
int led16 = 16 ;
int led17 = 17 ;
int led19 = 19 ;
int waitTime = 1000;
int cycleCnt = 0;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pins as output.
  pinMode(led11, OUTPUT);  //mast
  pinMode(led16, OUTPUT);  //mast
```

```
  pinMode(led17, OUTPUT);  //top
  pinMode(led19, OUTPUT);  //middle
  digitalWrite(led16, HIGH);        //        **  Corrected
  digitalWrite(led17, LOW);   // turn the LED on
  digitalWrite(led19, LOW);   // turn the LED on
  digitalWrite(led11, LOW);   // turn the LED on
  delay(waitTime/2);         // wait 1/2 second
}
// loop routine
void loop() {
digitalWrite(led19, HIGH);  // turn the LED off
  digitalWrite(led11, HIGH);  // turn the LED off
// led17 is on slower cycle
  // fmod divides by the second number and looks at the remainder
  if (fmod(cycleCnt,2) == 0)
  {
    digitalWrite(led17, HIGH);   // turn the LED off
  }
delay(waitTime/2);
digitalWrite(led19, LOW);  // turn the LED on
// slower blink
  if (fmod(cycleCnt,2) == 0)
  {
  digitalWrite(led17, LOW);   // turn the LED on
  }
//Every third time, the mast LED will not come on
  if (fmod(cycleCnt,3) != 0)
  {
    digitalWrite(led11, LOW);
  }
  delay(waitTime/2);
  cycleCnt = cycleCnt + 1;
 }
```

Figure 10. Corrected code.

Coding challenge: End-state definition. Because the program controlling the LilyPad behavior is a loop, it is important to realize that the final instruction at the end of the loop will flow immediately into the beginning instruction at the beginning of the loop. The desired behavior of the program in Figure 12 was to light all four corners together, flashing on and off.

However, since there is no delay after the LED turned on, the light is imperceptible as the LED immediately turns off again. To correct the bug, a delay should be inserted after line 21. Figure 11 is a schematic of the e-textile that was used with the coding challenge.

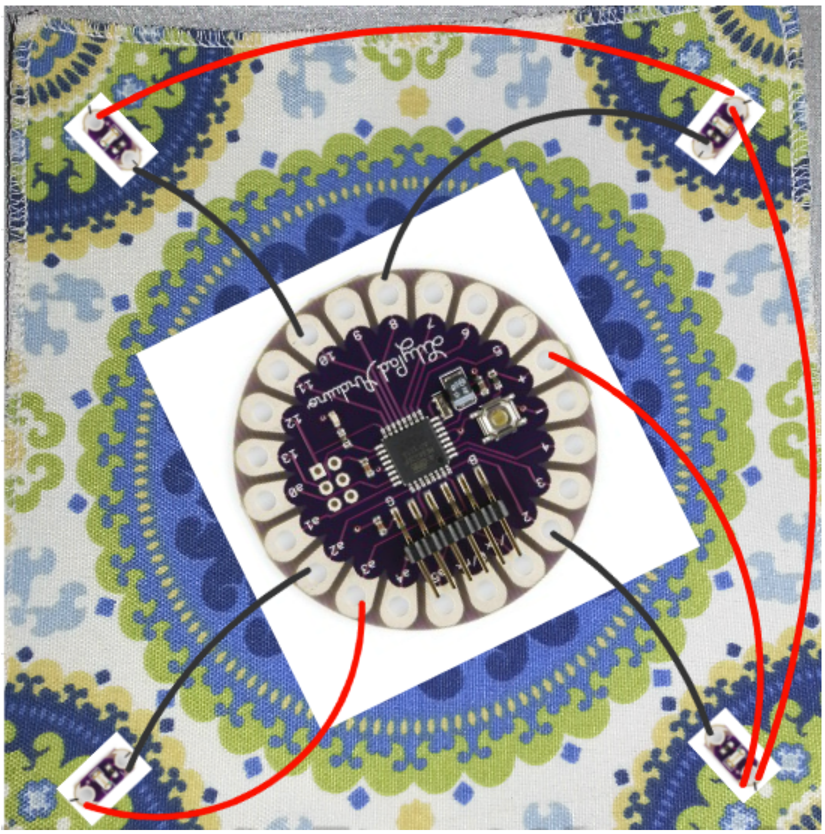


Figure 11. Schematic to accompany code in Figure 12.

```
1   /*
2    Bug 6, End State Definition, Buggy
3    Victoria Kim Dissertation
4
5    Desired behavior:  All corners light together, flash on and off
6    */
7
8   // Name the led pins on the Arduino.
9   int led6 = 6 ;
10  int led10 = 10 ;
11  int led17 = 17 ;
12  int led16 = 16 ;
13  int led19 = 19 ;
14  int waitTime = 1000;
```

```

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pins as output.
  pinMode(led6, OUTPUT);
  pinMode(led10, OUTPUT);
  pinMode(led17, OUTPUT);
  pinMode(led16, OUTPUT);
  pinMode(led19, OUTPUT);

  digitalWrite(led16, HIGH);   // Set as positive, Keep pin positive
}

// loop routine
void loop() {
  digitalWrite(led10, LOW);   // turn the LED on
  digitalWrite(led19, LOW);   // turn the LED on
  digitalWrite(led16, LOW);   // turn the LED on
  digitalWrite(led6,  LOW);   // turn the LED on

  delay(waitTime);            // wait 1 second

  digitalWrite(led10, HIGH);  // turn the LED off
  digitalWrite(led19, HIGH);  // turn the LED off
  digitalWrite(led16, HIGH);   // turn the LED on
  digitalWrite(led6,  HIGH);  // turn the LED on

}
```

Figure 12. Code illustrating end state error.

The code in Figure 13 is a possible correction of the error. It adds a delay at the end of the loop. In the code it is marked by ‘** Correction.’

```
/*
Bug 6, End State Definition, Corrected
Victoria Kim Dissertation
Desired behavior:  All corners light together, flash on and off
*/
// Name the led pins on the Arduino.
int led6  = 6 ;
int led10 = 10 ;
int led17 = 17 ;
int led16 = 16 ;
int led19 = 19 ;
int waitTime = 1000;
```

```
// the setup routine runs once when you press reset:
void setup() {
// initialize the digital pins as output.
pinMode(led6, OUTPUT);
pinMode(led10, OUTPUT);
pinMode(led17, OUTPUT);
pinMode(led16, OUTPUT);
pinMode(led19, OUTPUT);
digitalWrite(led17, HIGH);   // Set as positive, Keep pin positive
}
// loop routine
void loop() {
digitalWrite(led10, LOW);   // turn the LED on
digitalWrite(led19, LOW);   // turn the LED on
digitalWrite(led16, LOW);   // turn the LED on
digitalWrite(led6,  LOW);   // turn the LED on


delay(waitTime);           // wait 1 second


digitalWrite(led10, HIGH);  // turn the LED off
digitalWrite(led19, HIGH);  // turn the LED off
digitalWrite(led16, HIGH);   // turn the LED on
digitalWrite(led6,  HIGH);   // turn the LED on


delay(waitTime);           // wait 1 second                    ** Correction
}
```

Figure 13. Bug 6, corrected code.

Semi-independent Arduino project. As part of the graded course requirements, the students had an additional Arduino project which they designed and worked on semi-independently, over the next half-semester. The projects were semi-independent, because as the students work on their projects, they were asked to participate in a debugging forum, where they described problems they had encountered and how they resolved them. They were also encouraged to contribute to a class Wiki about e-textiles and Arduinos and to maintain a project notebook. Students were allowed to design and create their own Arduino project; originally it was anticipated that the students would create e-textile projects, but they were allowed to choose

more traditional Arduino projects, if they desired. The course professors gave the students a rubric that stated that their projects would be graded on interactivity, complexity, elegance, functionality, and creativity. Student projects were a mix of e-textile projects and other types of Arduino projects. Students brought their in-progress projects to the October F2F meeting, and then created project videos that were shared at the November F2F meeting. The level of complexity for the student projects was beyond that which the professors originally anticipated.

This progression of moving from pre-made bugged items to personally designed projects follows the use-modify-create pattern of engagement advocated for involving learners with rich computational environments (Lee et al., 2011). Selecting and designing their own projects gave students the opportunity to use the computational thinking perspective of expressing. Student projects are briefly described in Chapter 5.

Questionnaires. At the beginning of the course, students were asked to complete two questionnaires about their previous experience with, and confidence and attitudes toward computer and e-textile activities. At the completion of the learning module they were again asked to respond to the original attitudinal questionnaire. In addition, the students responded to a questionnaire about e-textile self-efficacy at the end of the first F2F meeting, and again at the end of the learning module. Finally, the students also completed an additional anonymous questionnaire about their perceptions of the usefulness of the debugging learning module at the completion of the module.

Final design-scenario with multiple-bugged item. When the students met for their second F2F meeting in November they participated in a last debug exercise. Originally it was planned that the final design scenario would be done individually, but a review of the recordings from the first design scenarios revealed that many students would be unduly stressed if they were

asked to do the debugging alone. Therefore, the original pairs were asked to work together again to solve the final debugging design scenario. Again, the pairs were recorded using Livescribe pens.

The final design-scenario contained all six of the previously described physical and coding bugs. A schematic for the e-textile is shown in Figure 14. The code in Figure 15 was the bugged code that accompanied the final debug scenario.

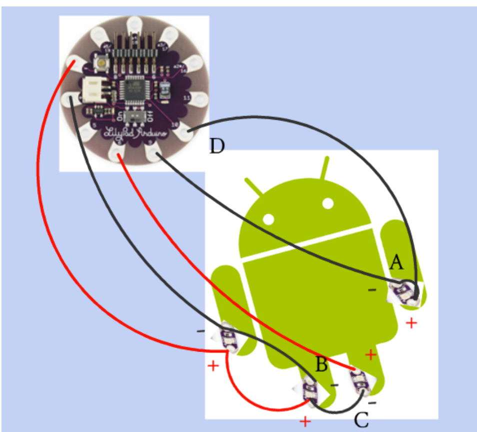


Figure 14. Schematic for the final debug scenario, multiple bugs.

```
/*
 New Bug 7     BUGGY,       Multibug
 Victoria Kim Dissertation

 Desired Action: The Robot's arms and legs alternate, in pairs.
 */

// Name the led pins on the Arduino.
int led5 = 5 ;     //   Robot's right arm
int led6 = 6 ;     //   Robot's right leg
int led9 = 9 ;     //   Robot's left leg
int led10 = 10;     //  Robot's left arm

int waitTime = 1000;

// setup runs once
void setup() {
  // initialize the digital pins as an output.
  pinMode(led5, OUTPUT);
```

```
  pinMode(led6, OUTPUT);
  pinMode(led9, OUTPUT);
  pinMode(led10, OUTPUT);

  digitalWrite(led5, HIGH);   //start with pin off
  digitalWrite(led6, HIGH);    //start with pin off
  digitalWrite(led9, HIGH);    //start with pin off
  digitalWrite(led10, HIGH);    //start with pin off

}

// Loop routine (continually repeats)
void loop() {
  digitalWrite(led5, LOW);    // turn arms on
  digitalWrite(led10, LOW);   // turn arms on

  digitalWrite(led6, HIGH);   // turn legs off
  digitalWrite(led9, HIGH);   // turn legs off

  delay(waitTime);          // wait 1 seconds

  digitalWrite(led5, HIGH);   // turn arms off
  digitalWrite(led10, HIGH);  // turn arms off

  digitalWrite(led6, LOW);    // turn legs on
  digitalWrite(led9, LOW);    // turn legs on

}
```

Figure 15. Code for the final debug scenario.

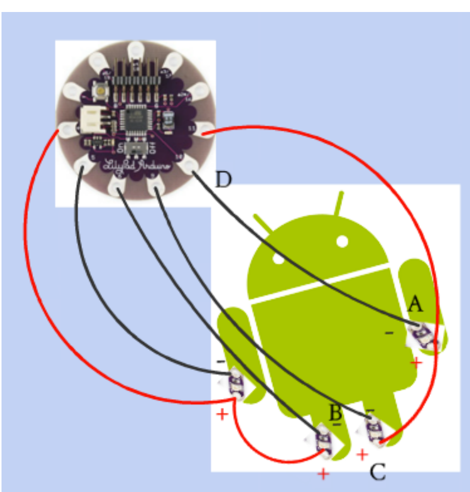


Figure 16. Schematic for final debug scenario, physical bugs corrected.

Figure 16 shows one way of correcting the physical bugs in the final debug scenario. The code in Figure 17 is a possible correction of the coding errors in the final design scenario when used with the corrected schematic in Figure 16.

```
/*
 New Bug 7 CORRECTED, Multibug
 Victoria Kim Dissertation

 Desired Action: The Robot's arms and legs alternate, in pairs.
 */

// Name the led pins on the Arduino.
int led5 = 5 ;     //   Robot's right arm
int led6 = 6 ;     //   Robot's right leg
int led9 = 9 ;     //   Robot's left leg
int led10 = 10;     //  Robot's left arm  (neg)
int led11 = 11;     //  Robot's left arm  (pos)              ***  bug fix

int waitTime = 1000;

// setup runs once
void setup() {
  // initialize the digital pins as an output.
  pinMode(led5, OUTPUT);
  pinMode(led6, OUTPUT);
  pinMode(led9, OUTPUT);
  pinMode(led10, OUTPUT);
  pinMode(led11, OUTPUT);              //                   ***  bug fix

  digitalWrite(led11, HIGH);   //set pin to positive        ***  bug fix

  digitalWrite(led5, HIGH);   //start with pin off
  digitalWrite(led6, HIGH);    //start with pin off         *** bug fix (sewing)
  digitalWrite(led9, HIGH);    //start with pin off
  digitalWrite(led10, HIGH);    //start with pin off

}

// Loop routine (continually repeats)
void loop() {
  digitalWrite(led5, LOW);    // turn arms on
  digitalWrite(led10, LOW);   // turn arms on

  digitalWrite(led6, HIGH);   // turn legs off
  digitalWrite(led9, HIGH);   // turn legs off

```

```
delay(waitTime);          // wait 1 seconds

digitalWrite(led5, HIGH);   // turn arms off
digitalWrite(led10, HIGH);  // turn arms off

digitalWrite(led6, LOW);    // turn legs on
digitalWrite(led9, LOW);    // turn legs on

delay(waitTime);          // wait 1 seconds               *** bug fix

}
```

Figure 17. Possible code corrections for Bug 7.

Instrumentation

The following instruments were used to gather data to inform the design:

- E-Textile Self-Efficacy Questionnaire. An 11-question e-textile self-efficacy questionnaire was given at the end of the first F2F meeting and again at end of the learning module. The questions are modifications of those found in the computer self-efficacy measure—revised (Kwahk & Ahn, 2010). The phrase ‘a job using the information system’ was replaced by ‘an e-textile project using the LilyPad Arduino environment’. The complete questionnaire is included in Appendix C. A sample question is shown here to illustrate the style of the questionnaire.

1	I could complete an e-textile project using the LilyPad Arduino environment if there were **no one around** to tell me what to do.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree

Figure 18. Sample question from the e-textile self-efficacy instrument.

- Computer Experience Questionnaire. A 12-question questionnaire about past experience with programming, electronics, and e-textiles was given at the

beginning of the learning module, since prior experience with these topics was anticipated to influence the students' projects and views. Sample questions from the questionnaire are shown. The complete survey is shown in Appendix B.

4	I have never made an electronics project.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
5	I have tinkered with HTML.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
6	I am comfortable programming in several languages.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree

Figure 19. Sample questions from the computational experience questionnaire.

- Computational Attitude Questionnaire. A 20-question questionnaire about attitudes was given both at the beginning and end of the learning module. Participants were asked to respond to semantic pairs related to computing, electronics, e-textiles and technology. The semantic pairs were based on those used in Tyler-Wood, Knezek, & Christensen (2010). Sample questions from the questionnaire are shown in Figure 20. The complete survey is shown in Appendix D.

To me, TECHNOLOGY is

1.	mundane	1	2	3	4	5	6	fascinating
2.	unappealing	1	2	3	4	5	6	appealing
3.	hard	1	2	3	4	5	6	easy
4.	unimportant	1	2	3	4	5	6	important
5.	boring	1	2	3	4	5	6	exciting

Figure 20. Sample questions from the computational attitude questionnaire.

- The Computational Thinking Rubric for Examining Students' Project Work. A rubric was used to document evidence of the presence or lack of computational thinking. The rubric was applied to the pair recordings of the design scenarios, forum posts, and reflection papers. In addition to computational thinking

elements, the rubric has a place to note evidence of confidence, interest, and gender focus. The Computational Thinking Rubric for Examining Students' Project Work is a new instrument created for this research, but it is closely aligned with Brennan and Resnick's (2012) article. While it was used to analyze projects and interactions for computational thinking in this research, it is also one of the design elements that were examined for opportunities for improvement. The original rubric contained Brennan and Resnick's (2012) breakdown of computational thinking concepts, practices, and perspectives. As new concepts, practices, and perspectives emerged they were added to the rubric. In addition to the computational thinking aspects, there is a place to note evidence of error correction, since debugging is of particular interest to this study. The rubric was initially validated by eliciting feedback about its design from those who are familiar with computational thinking and its assessment. A few rows of the rubric are shown in Figure 21. See Appendix E for the complete rubric.

Computational Thinking Concept, Practice, or Perspective (Brennan & Resnick)	Description Location in project, photo, etc., distinguishing characteristics	Error Free (EF), Error with Attempt to Correct (AC), Error Corrected (EC)	Notes Debugging Technique Used, Etc.
Practice: Abstracting and Modularizing			
Perspective: Expressing			
Perspective: Connecting			

Figure 21. Sample rows from the rubric for examining students' project work.

- Group Debrief. Following the completion of the first six debug exercises, study participants participated in a group debriefing conducted by the course professor. Students answered questions about their debugging processes, the experience working with partners, and connections between the exercises and their own projects.
- Feedback about Final Debugging Exercise. A short interview was planned to follow the multi-bug design scenario, however other course needs superseded this, and the study participants were asked to respond to several questions posted in the course forums, instead. Questions asked about the participants' debugging methods in the design scenario, most challenging aspects of the exercise, and prior debugging experience.
- Student Evaluation of the Learning Module Questionnaire. Because it was anticipated that the student evaluations of the learning module might contain criticism, to obtain the participants' true feelings, the evaluation was done anonymously. The evaluation contains 16 close-ended questions and six open-ended questions. A copy of the evaluation can be found in Appendix F. Sample questions are shown here in Figure 22.

8	I really liked being able to experiment with e-textiles.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
9	I would have been more interested in Arduino that use soldering instead of sewing.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
10	The debug forum was useful and informative.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree

17. What suggestions do you have to improve this e-textile learning module for future students?

18. What part of the e-textile learning module did you find most enjoyable?

Figure 22. Sample questions from the student evaluation of the learning module.

Pilot

Prior to beginning data collection with the EDLT740 participants, sample materials and the study instruments were piloted. The pilot is described below.

Pilot participants. The participants in the pilot were the four undergraduate students enrolled in DeVry University's DBM449 database course taught by the researcher during the May 2013 session. The students were nearing the end of their CIS studies. All four students in the course were male. The participants were put into two groups for the debugging exercises and worked on the e-textile project as one group. Figure 23 and Figure 24 are images of the pilot participants working on the sample debugging exercises and the small e-textile projects that were provided.

Pilot materials. The materials that were piloted were the preliminary circuit exercises, one engineering bug, one programming bug, and a sample e-textile project. Additionally, the use of the Livescribe pens for recording and the tools in the e-textile kit were also piloted. Since the pilot participants were available for only two days, rather than have them create their own project, a stenciled textile of a monkey was given to them, with LED's, battery, thread, and other e-textile tools. The students created their own design of LED's on the textile.

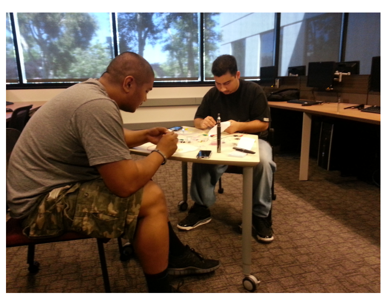


Figure 23. Pilot participants working on e-textile project.

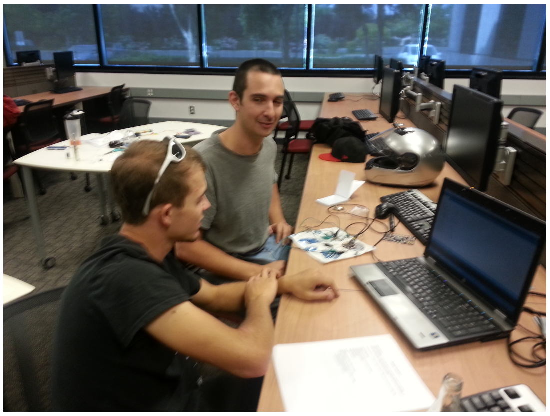


Figure 24. Pilot participants working on debug exercises.

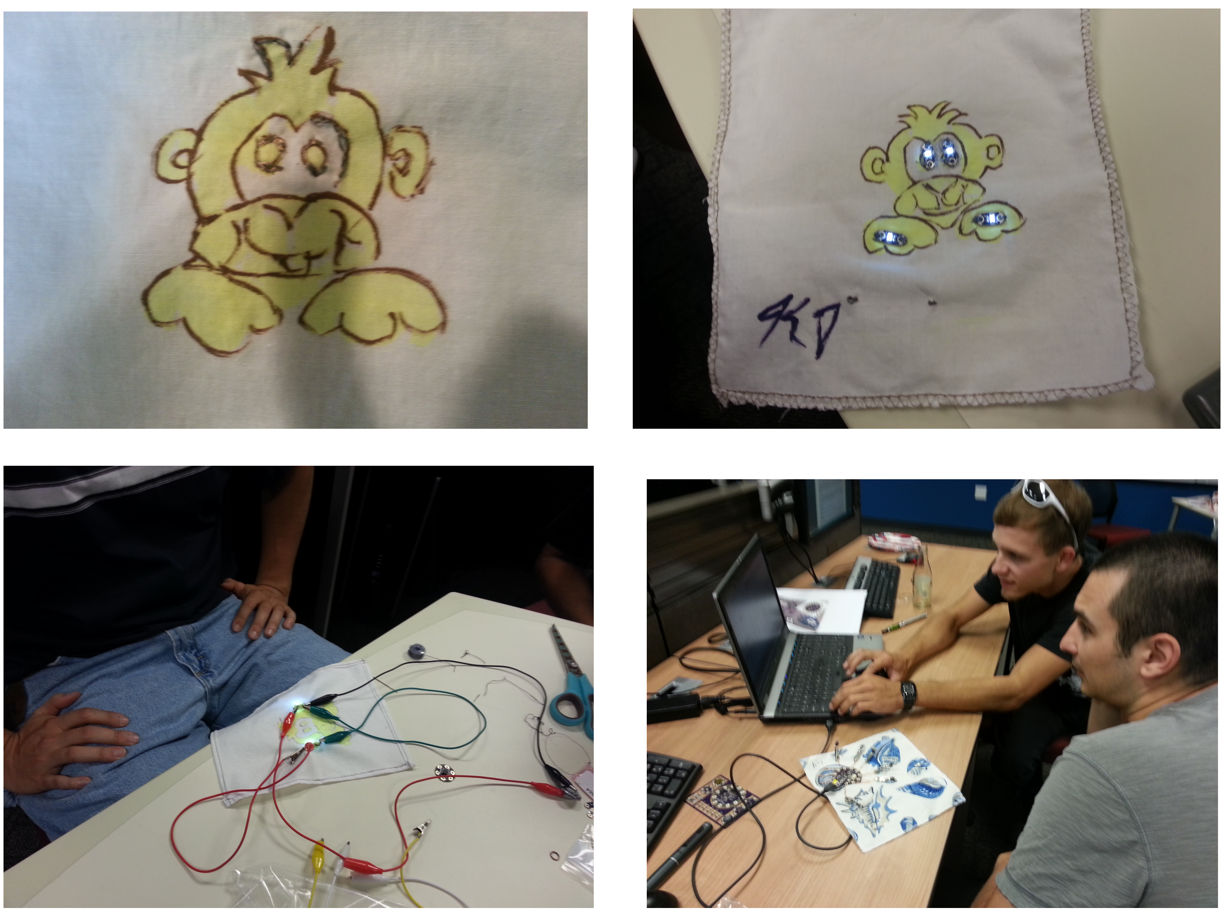


Figure 25. E-textile project and debug exercise.

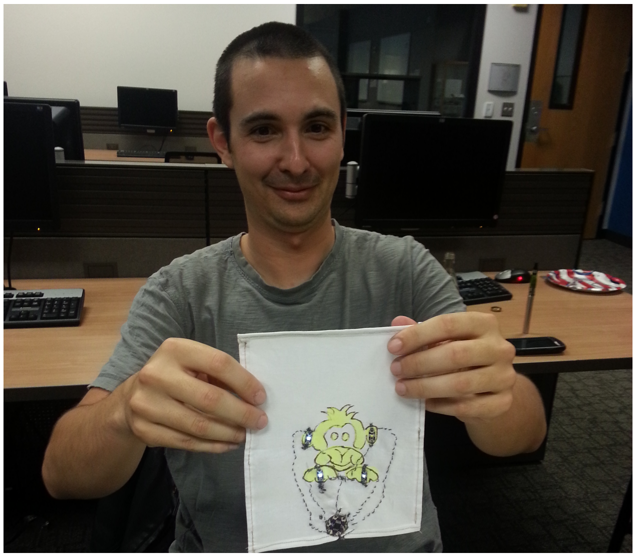


Figure 26. Showing finished project.

Piloted instruments. The instruments that were piloted were paper versions of the E-Textile Self-Efficacy Questionnaire, the Computer Experience Questionnaire, the Computational Attitude Questionnaire, and the Student Evaluation of the Learning Module. All instruments were given the second day of the pilot. A summary of the collected data is given in the following section.

Pilot data collected. A summary of the data collected from the pilot participants is given in this section.

Reported prior experience. Table 3 summarizes the pilot participants reported prior experience with e-textiles and electronics. The low average reported score for never programmed and never completed an electronics project indicate that the group did have experience programming and using electronics. The 6 for never completed an e-textile project means that e-textiles were very new to the pilot participants.

Table 3

Reported Prior Experience, Pilot Participants, Non-Agreement (1) to Agreement (6)

Gender	Never Programmed	Never Sewn	Never Complete E-Textile Project	Never Completed Electronics Project
M (4)	1.25	3.5	6.0	2.0

Computer experience. Self-efficacy. The overall e-textile self-efficacy for the pilot participants was 4.8 after completing the two-day experience with the debugging exercises and the e-textile project. The overall self-efficacy for the study participants after their e-textile experiences at the first F2F are shown in Table 16.

Table 4 shows a summary of the study participants' computer experience. The scores of 5.5, 5, and 5.5 on the questions about being comfortable programming in several languages, learning new programming languages easily, and being good at debugging are much higher than the scores of 2, 2.6, and 3.7 reported for the study participants (see Table 2), due to the pilot participants' major being CIS.

Self-efficacy. The overall e-textile self-efficacy for the pilot participants was 4.8 after completing the two-day experience with the debugging exercises and the e-textile project. The overall self-efficacy for the study participants after their e-textile experiences at the first F2F are shown in Table 16.

Table 4

Computer Experience, Pilot Participants, Non-Agreement (1) to Agreement (6)

Gender	Have Artistic Side	Comfortable Programming in Several Languages	Learn New Programming Languages Easily	Good at Debugging
M (4)	4.0	5.5	5	5.5

Learning module evaluation. The pilot participants completed the same Learning Module Evaluation as the study participants but were told to ignore questions referring to elements that didn't apply, such as class Wiki and project notebook.

The four students had an average agreement score of 4.75 on the question about whether they enjoyed working with e-textiles and an average score of 5 on the question about whether the bugs were too easy and on the question about whether e-textiles make good projects for both males and females (scale from 1 to 6, disagreement to agreement).

The answers to some of the pilot participants' open-ended questions of the Learning Module Evaluation are presented below.

What was the most enjoyable part of the learning module?

- I found that the programming was enjoyable once you know that all connections work
- All of it
- When we were sewing the LED's
- Class interaction

What are your suggestions to improve the learning module?

- I suggest use thicker thread with smaller needles
- Concise lessons on programming
- Could use more variety in the template to sew, more time on the programming

Other to share?

- Take into account the greater applications beyond clothing, running equipment, advertising.

Observations and feedback from pilot. The pilot of the materials and instruments showed several things:

- The wire stems off the LilyPad were not a reliable way of making the LilyPad portable. (One of the researchers' first efforts to make the LilyPad portable.)
- The materials in the kits were sufficient to complete the debug exercises.
- The design scenarios for the debug exercises were able to elicit debugging concepts, practices, and perspectives.
- The Livescribe pens captured the debugging between the group partners.
- It took only minutes to fill out all the instruments.
- E-textile projects could keep the interest of male students.
- Prior experience with automotive batteries transferred over to the e-textile circuits.
- Students were unwilling to leave an e-textile project unfinished.
- Students were engaged enough to ask for more time with the projects.
- Students with prior programming experience could quickly pick up the C language used in the Arduino environment.
- Students enjoyed the camaraderie of working together on the project and chatting about events in their lives.

The researcher showed some example e-textile projects so the pilot participants could understand some of the possibilities. One example that seemed to catch their interest was a turn-signal jacket, modeled off of Leah Buechley's project. Three of the four pilot participants happened to ride motorcycles, which probably accounted for their heightened interest in the jacket.

VKim: And something that I actually thought might be useful for you guys.

VKim: They call it a turn signal jacket, and I'm still debugging mine.
MP201: Is there a switch or a button you can press?
VKim: Yeah.
MP201: For left and right?
VKim: Yeah.
MP201: No way. I want one.
. . .
MP101: Can you do both at once, kind of like a hazard?
VKim: You can. That's part of what I am debugging.
(Pilot Group 1, Introduction to Project)

Students enjoyed the change of pace of working together on the e-textile projects. As the students worked, their conversations ranged over several topics including crime at the local mall, their current classes, moving to America as a child, and friendships. The following conversational snippet captures the feelings that the students experienced as they worked together.

MP101: Sewing is bringing out some interesting conversations.
VKim: Yeah, see?
MP201: I feel like an old lady right now.
MP102: (laughing) Sitting here gossiping and sewing.
MP101: What is wrong with us? Oh my God.
(more laughter)
MP201: It brings out the inner old lady in all of us.
VKim: Are we still recording?
VKim: I am going to have to put that in my paper.
MP201: This is fun. We should do this more.
VKim: That's right!
MP201: We should have a sewing club. A bunch of programmers for computers.
MP201: Hey, where you guys going tonight? Well we are having a sewing club party.
(Laughter)
(Pilot Group, Project Time)

This sentiment was later echoed in the main study when students discussed the non-structured project-time they experienced one evening after their regular class meetings.

The student in Figure 27 brought the embroidery hoop from his mother's home on the second day because he wanted the fabric to be easier to work with.

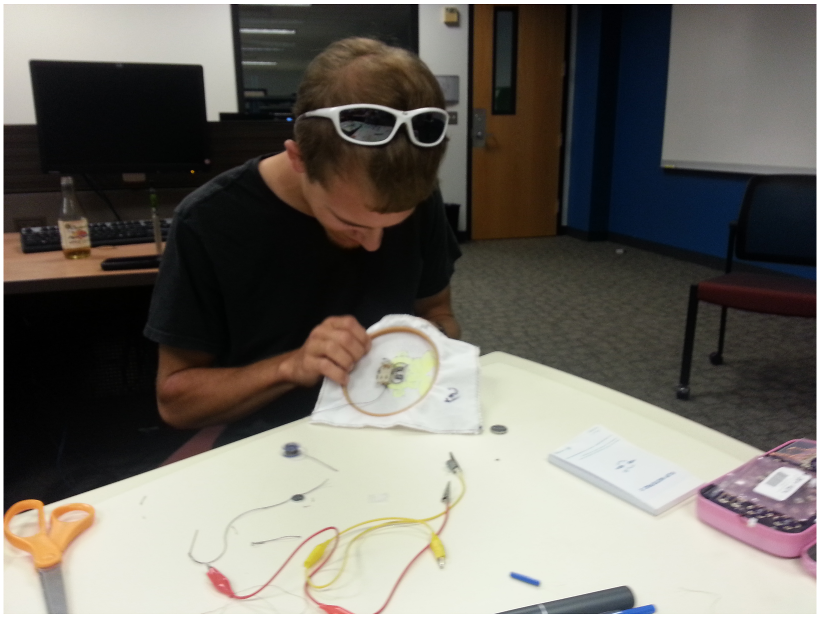


Figure 27. Student using embroidery hoop brought from home.

Although the pilot participants differed from the study participants because they were undergraduate students and also because they were CIS majors, the pilot showed that the materials and methods for collecting data were workable. A study of the data collection in the main study continues below.

Data Collection

Data was collected using the previously described instruments in the following manners.

E-textile self-efficacy questionnaire. See Appendix C. This questionnaire was given twice: once at the completion of the F2F meeting, and once at the end of the learning module four weeks later. The questionnaire was administered through the Qualtrics system and contained participant names. There was a question for students to indicate that they declined to participate.

Computer experience questionnaire. See Appendix B. This questionnaire was given at the beginning of the learning module to determine prior experience with computation. The

questionnaire was administered using the Qualtrics system and contained participant names. There was a question for students to indicate that they decline to participate.

Computational attitudes questionnaire. See Appendix D. This questionnaire was given at the beginning and end of the learning module to determine attitudes toward computation. The questionnaire was administered through the Qualtrics system and contained participant names. There was a question for students to indicate that they decline to participate.

Student evaluation of the learning module. See Appendix F. An anonymous evaluation was requested at the end of the learning module to obtain student feedback that could be used to improve the design. This paper and pencil evaluation was given at the end of the first design scenarios, and was collected by the cadre graduate assistant. There was a check box for students to indicate that they decline to participate.

Computational thinking rubric for examining students' project work. See Appendix E. The computational thinking rubric was applied, by the researcher, to the data collected through recordings, forum posts, and final reflection papers. Project notebooks and blogs, wiki entries, and course projects also added insight to the students' computational thinking. The collection of this data is described more fully in the following paragraphs.

Recordings of pairs participating in six debugging design scenarios. When given an opportunity whether or not to participate, all students in the EDLT740 course agreed to be part of the study and granted permission for voice recordings of their design scenario work. Livescribe pens, a type of digital recorder, recorded their interactions as they worked together on the six debug design scenarios. There were nine student pairs working simultaneously, so much of the interaction took place outside the presence of the researcher. The recordings allowed the

researcher to look for evidence of computational concepts, practices, and perspectives as the pairs worked together to find the bugs.

Forum posts about debugging experiences. Over the six weeks that the course focused on Arduino projects, the participants were asked to post to an online course forum where they discussed their experiences with creating and debugging their own personally designed projects. The researcher observed how the students supported each other's efforts to debug their projects, and also looked for evidence of computational concepts, practices, and perspectives. Additionally, the researcher requested input about how the F2F aspect of the learning module connected to the students' work on their own Arduino projects.

Entries to class Wiki. Students were asked to contribute to a class Wiki about resources that were useful to them in their project investigations. The Wiki was available to the researcher. The two students with the most prior experience in computing made contributions to the Wiki near the beginning of the learning module. Several other students made entries at the end of the learning module.

Course project analysis. The final course Arduino projects and video presentations were also examined for evidence of computational thinking.

Table 5 summarizes how the instruments were administered.

Table 5

Schedule of Instrument Administration

Instrument	F2F1, Before Design Scenarios	F2F1, After Design Scenarios	F2F2, Week 7	Ongoing
Computer Experience	X			
Computer Attitudes	X		X	
E-Textiles Self-Efficacy		X	X	
Computational Thinking Rubric				X
Student Evaluation of Learning Module			X	

Project notebook and blog posts. One of the study requirements was that students maintain a project notebook or blog. The researcher examined the project notebooks and blogs for insight into the students' project creation.

Data Analysis Plan

A mapping of key outcomes to the instruments described in the preceding discussion is shown in Table 6. Then, detailed examples of what computational concepts, practices, and perspectives were anticipated to look like in the LilyPad Arduino environment are presented. Actual examples from the debug exercise transcripts are given in Chapter 4.

The design scenario interview recordings, the forum and Wiki posts, reflection papers, and project notebooks and blogs were examined for evidence of computational thinking concepts, practices, and perspectives. The Computational Rubric for Assessing Students' Work was used to collect the data. Brennan & Resnick (2012) gives examples of each of the concepts, practices, and perspectives from the Scratch programming language perspective. The same format is used here to show what the computational concepts, practices, and perspectives might look like in the LilyPad Arduino programming environment. The findings from the inspection of the actual artifacts, recordings, and posts are discussed further in Chapter 4.

Identifying computational concepts. The presence of computational concepts can be noted by examining artifacts for related programming structures, but whether or not there is understanding of the concept is better identified by interviewing the designer about his or her artifact. This is because the designer might have used some coding blocks, but not have truly understood all that they did. This is particularly true due to the many examples that are easily available on the Internet. In this study, students' debugging sessions were recorded and

transcribed to see the concepts being used, and students were also debriefed following their participation in the debugging exercises.

Table 6

Desired Outcomes Correlated to Instruments

Purpose or Key Outcome	Instrument	Notes
Create e-textile activities that support the development of computational thinking		Initial design built upon review of literature. Next iteration of activities was informed by student evaluations and researcher observation.
Identify assessment characteristics and procedures to show development of computational thinking concepts, practices, and perspectives.	E-textile Self-Efficacy Questionnaire (SEQ), Computational Attitude Questionnaire (CAQ), Learning Module Evaluation (LME), Computational Thinking Rubric (CTR)	This happened in the analysis of the design, at the end of the iteration.
Use feedback to improve learning module design	LME, Observation	This happened in the analysis of the design, at the end of the first iteration.
Contribute to the development of computational thinking concepts	CTR	
Contribute to the development of computational thinking practices	CTR	
Contribute to the development of computational thinking perspectives	CTR	
Increase confidence in computing abilities.	SEQ, CTR	
Appeal to diverse non-computer science majors.	LME, CAQ, CTR	All assessments identify participant gender as demographic data.

Concept: Sequences. The concept of sequence is one of the most fundamental of computational thinking. It means that computational instructions and actions occur as a series of steps, one following the other. In a LilyPad e-textile project, a sequence of instructions might be used to have different lights appear in a certain pattern.

The image in Figure 28 shows an e-textile project next to part of its program code. One place in the code that sequence is illustrated is in lines 4 to 13. The LED connected to pin 4 is turned on after the LEDs connected to pins 6 and 2 are turned on and off.

```
1   digitalWrite(led6,HIGH);  // turn the LED on
2   delay(waitTime/2);        // wait 1/2 second
3   digitalWrite(led6, LOW);  // turn the LED off
4   digitalWrite(led2, HIGH);
5   digitalWrite(led4, LOW);
6   delay(waitTime);          // wait for a second
7   digitalWrite(led6,HIGH);
8   delay(waitTime/2);
9   digitalWrite(led6, LOW);
10  digitalWrite(led2, LOW);
11  digitalWrite(led4, HIGH);
12  delay(waitTime);
13  digitalWrite(led4, LOW);
```

Figure 28. Example of computational concept–sequence.

Concept: Loops. When a sequence of steps occurs repeatedly, it is possible to use a loop to do the repetition rather than manually repeating the instructions. Loops express repetitive sequences of instructions more succinctly (Brennan & Resnick, 2012). Because LilyPad programs are loaded into the processor while the processor is connected to the computer, then the connection to the computer is removed, the whole program that resides in the processor is a loop that occurs over and over. Students had to use this main loop to be able to program their objects to behave in determined ways. Some projects had additional opportunities to use loops.

The code in Figure 29 shows an example for loop that turns lights on, in sequence, three times.

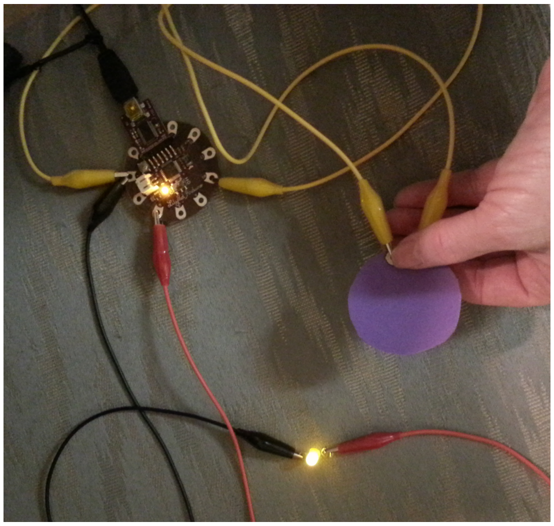
```
for (int countIt = 1; countIt < 4; countIt++) {
   // turn the pins on:
   digitalWrite(led2, HIGH);
```

```
    delay(waitTime);
    digitalWrite(led4, HIGH);
    delay(waitTime);
    digitalWrite(led6, HIGH);
    delay (waitTime);
    // turn the pins off:
    digitalWrite(2, LOW);
    digitalWrite(4, LOW);
    digitalWrite(6, LOW);
}
```

Figure 29. Example of computational concept–loops.

Concept: Events. Events are the cause and effect of computation. Events are used as triggers to cause other actions to occur. While the Arduino programming language is primarily a sequential rather than an event driven language, there are ways that events can be programmed. In the LilyPad Arduino e-textile environment, circuits being completed by button pushes or other means, or input from other types of sensors can trigger events. Lines 1 to 15 in the code in Figure 30 check for a button push. If one occurs, the state of the LED is changed. Please note that because a physical button push causes the electrical signal to bounce rapidly from low to high several times before it settles, slightly more complicated logic has to be included to determine if there has been a change of state in the button. Electrical engineers call this 'debouncing'.

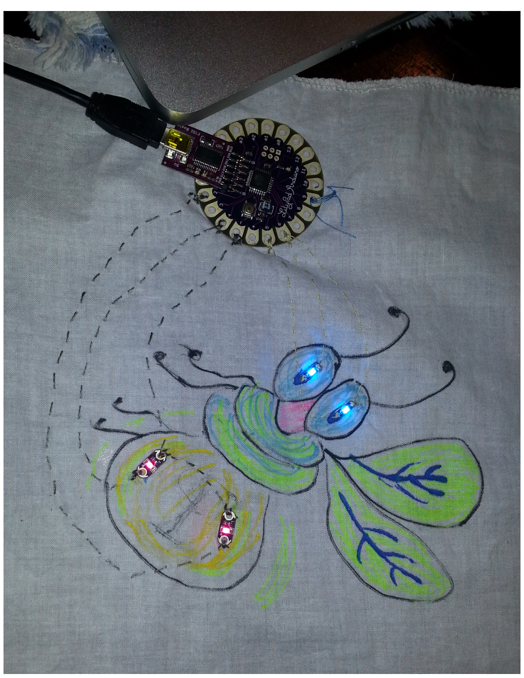
Concept: Parallelism. The concept of parallelism is when actions occur simultaneously. In an e-textile project this might appear as two separate sections of lights, tied to different LilyPad output pins, coming on at the same time in response to an event.



```
//  Detect button push case
  chgLED = ((latchedButtonState == HIGH) && (buttonState == LOW));
 latchedButtonState = buttonState;

  // set the LED
  if (chgLED)
  {
    newLedState = !ledState;
    digitalWrite(ledPin, newLedState);
    ledState = (newLedState);
  }
  // save the reading.  Next time through the loop,
  // it'll be the lastButtonState:
  lastButtonState = reading;
}
```

Figure 30. Example of computational thinking concept–events.

In the code shown in Figure 31, lines 17 to 20 are an example of how parallelism might look in the LilyPad Arduino environment. If the counter is an even number, all the LED's are lit.



```
1   if   (fmod(countIt,2) == 1)  // Is it odd?
2    {
3      Serial.println("countIt = ODD");
4      // turn the eye LED's on
5     digitalWrite(led6, HIGH);
6
7     delay(waitTime);
8
9     // turn the pin off:
10    digitalWrite(led6, LOW);
11
12     delay(waitTime);
13   }
14
15  else
16  {
17    Serial.println("countIt = EVEN");
18    digitalWrite(led4, HIGH);
19    digitalWrite(led2, HIGH);
20    digitalWrite(led6, HIGH);
21    delay(waitTime);
```

	  digitalWrite(led4, LOW);
	  digitalWrite(led2, LOW);
	  digitalWrite(led6, LOW);
	   delay(waitTime);
	}
	 countIt++;

Figure 31. Example of computational thinking concept–parallelism.

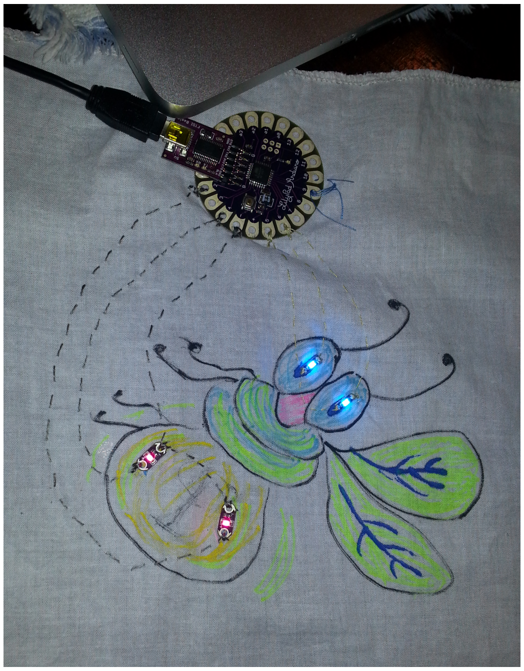
Concept: Conditionals. Conditional refers to the ability to take different actions based on something else, the ‘condition’. The condition could be the value of a variable, or the state of a component. Being able to take different actions based on the condition of something is what gives programs the power of behaving differently in different situations. The most common way of using conditionals is with an if statement.

Lines 1 to 27 in Figure 32 represent a conditional statement that takes different actions depending on the value of a variable called countIt. (The function fmod on line 1 looks at the remainder after division of the two arguments. Since the dividing number is two, this is a way to check if a number is odd or even.) The variable is incremented by one on each pass through the loop, so the behavior of the LED’s alternates.

Concept: Operators. Operators allow manipulations of numbers and expressions. The Arduino environment is based on the C programming language and has many mathematical and text operators that are available for use. A simple example of the + operator is shown in line 16 of the following code snippet in Figure 33.

Concept: Data. The computational concept of data is about storing, retrieving and updating useful values using variables (Brennan & Resnick, 2012). Examining the setup area of a LilyPad Arduino program can give a good example of the variables being created for storing

data. In the following code snippet shown in Figure 34, the data stored in the counter is displayed on the screen, which is a very useful technique for debugging. (See line 17 & 18.)



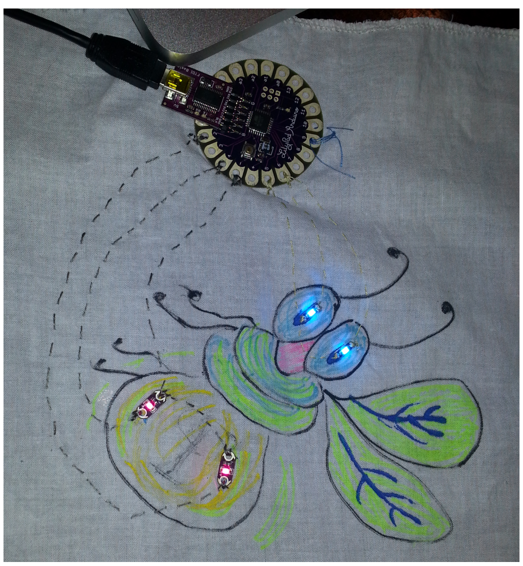
```
1   if  (fmod(countIt,2) == 1)  // Is it odd?
2    {
3       Serial.println("countIt = ODD");
4       // turn the eye LED's on
5      digitalWrite(led6, HIGH);
6      delay(waitTime);
7
8      // turn the pin off:
9      digitalWrite(led6, LOW);
10      delay(waitTime);
11   }
12
13  else
14  {
15     Serial.println("countIt = EVEN");
16     digitalWrite(led4, HIGH);
17     digitalWrite(led2, HIGH);
18     digitalWrite(led6, HIGH);
19     delay(waitTime);
20
21     digitalWrite(led4, LOW);
22     digitalWrite(led2, LOW);
23     digitalWrite(led6, LOW);
24      delay(waitTime);
25  }
26    countIt++;
27
```

Figure 32. Example of computational thinking concept–conditionals.

```
1   void loop() {
2
3    //
4      digitalWrite(led4, HIGH);
5      digitalWrite(led2, HIGH);
6      digitalWrite(led6, HIGH);
7
8      delay(waitTime);
9
```

```
    digitalWrite(led4, LOW);
    digitalWrite(led2, LOW);
    digitalWrite(led6, LOW);

    delay(waitTime);

    countIt = countIt + 1;
    Serial.print("countIt = ");
    Serial.println(countIt);

}
```

Figure 33. Example of computational thinking concept–operators.



```
void loop() {
  //
   digitalWrite(led4, HIGH);
   digitalWrite(led2, HIGH);
   digitalWrite(led6, HIGH);

   delay(waitTime);

   digitalWrite(led4, LOW);
   digitalWrite(led2, LOW);
   digitalWrite(led6, LOW);

   delay(waitTime);
   countIt = countIt + 1;
   Serial.print("countIt = ");
   Serial.println(countIt);
}
```

Figure 34. Example of computational thinking concept–data.

Identifying computational practices. While at least the presence of computational concepts can be noted by examining artifacts for related programming structures, computational practices may not be evident in the artifact. Brennan & Resnick (2012) indicates that concepts alone do not cover all aspects of computational thinking.

> From our interviews with and observations of young designers, it was evident that framing computational thinking solely around concepts insufficiently represented other elements of designers' learning and participation . . . Computational practices focus on the process of thinking and learning, moving beyond what you are learning to how you are learning (p. 6-7)

Practice: Being incremental and iterative. Computational projects progress in steps. The Scratch programmers interviewed by Brennan and Resnick (2012) "described iterative cycles of imagining and building – developing a little bit, then trying it out, and then developing further based on their experiences and new ideas" (p. 7). Evidence of the computational practice of being incremental and iterative was searched for in the project notebooks and forum posts of the study participants as well as in the interview recordings.

The practice of being incremental and iterative is valued in computation because it can help designers establish a baseline of what works that they can build upon. By adding a little bit at a time, the designer is able to isolate problems in the newly added code. Additional ways that e-textile designers can be incremental is by adding the actual LilyPad components incrementally, or by validating the actual soft circuits as they sew them one by one.

Practice: Testing and debugging. This study was particularly interested in the computational practice of testing and debugging. The participants had hands-on debugging experiences early in their introduction to e-textiles, and they participated in a class forum where they discussed their own debugging experiences and supported each other's debugging efforts. It was expected that as they gained more experience with the LilyPad Arduino environment and their own e-textile projects, their testing process and debugging skills would become more refined. Collected data revealed specific debugging and testing practices, which were identified as emergent sub-practices of testing and debugging in Chapter 4.

Practice: Reusing and remixing. Programmers frequently reuse their own code or code created by others. It is viewed as a matter of efficiency. Reusing and remixing is one of the computational practices identified by Brennan & Resnick (2012). The article affirms that "reusing and remixing support the development of critical code-reading capacities and provoke important questions about ownership and authorship" (Brennan & Resnick, 2012, p. 8). Interview questions explored the participants' reuse and remix practices, and the participants' pair interactions posts were examined for further enlightenment.

Practice: Abstracting and modularizing. Brennan & Resnick (2012) characterizes the practice of abstracting and modularizing as "building something large by putting together collections of smaller parts" (p. 9). Modularizing can make it easier for computational thinkers to focus on higher-level artifact behaviors by encapsulating the lower-level instructions. It can also make code easier for others to understand. Modularization works well with the practice of reusing code. Abstraction is one of the computational thinking practices common to most definitions of computational thinking (Grover & Pea 2013; Polin, 2012).

Identifying computational perspectives. The third dimension of computational thinking is computational perspective. As people become computational thinkers, their perceptions of the world around them change. Brennan & Resnick (2012) found that "young designers describe evolving understandings of themselves, their relationships to others, and the technological world around them" (p. 10). An example of this is someone who looks at a familiar object and wonders what sequence of commands would program its behavior. In the discussion about computational thinking assessment, Brennan & Resnick (2012) points out that it is challenging to understand changes in computational thinking perspectives, but that some social and psychological insights occurred in interview conversations. Since the participants in the

research undertaken here were doctoral students in learning technologies, it was hoped that they would have a meta-awareness of changes in their own perspectives that they would share in the interviews and their posts. Evidence of computational thinking perspectives is discussed in Chapter 4.

Perspective: Expressing. Expressing is when a person sees a computational medium as something that he or she can use to create an object or express him or herself. Expressing is different than just using technology as a consumer. Interview questions and project videos about the participants' own projects added enlightenment about how well they felt they were able to express their creative ideas.

Perspective: Connecting. Connecting, as used by Brennan and Resnick (2012), refers to social connections with others, either in person or online. Computational thinkers share achievements and challenges with others and are energized by sharing their creations. There are multiple ways to connect with others interested in e-textiles, from Maker groups to online forums. The student participants in this study shared about their experiences connecting with others in their forum posts, interviews, facebook groups, blogs, and questionnaires. The class Wiki was also a way to help students connect within the cohort group. This study has extended the perspective of connecting to include connecting to how technology works and how it is used.

Perspective: Questioning. As people become computational thinkers, they "feel empowered to ask questions about and with technology" (Brennan & Resnick, 2012, p. 11). An example of this would be someone who learns about circuits in e-textiles and then wonders about the circuit connections they see in a flashlight. Another example would be if a person begins to do laundry and wonders if doing laundry and writing a paper is an example of parallel processing.

Coding the data. The collected data was coded for the presence of each of the computational concepts, practices, and perspectives described before, as well as interest, confidence, and attitude. As new types of concepts, practices, and perspectives emerged, they became part of the codes that were applied. After the last transcript was coded, another pass through all the transcripts was conducted, to catch examples of the newly emerged codes. Reliability was established by having a person familiar with these themes independently review samples of the coding. Any differences in coding were discussed with the goal of arriving at consensus. The discussions revealed minor issues in identifying themes, and those were addressed.

Ethical Considerations (Human Subject Protections)

The students in the EDLT740 course would receive an introduction to e-textiles, create an e-textile project, document their processes, and participate in discussions throughout the term, whether or not this study took place. However, the multiple data collections of questionnaires and design scenario interviews added to the amount of time that participant students were required to put into this part of the course. There was an additional emphasis on debugging that would not have been part of the course if the study were not taking place. However, this could prove to be a benefit to the students, over time, if they continue to create e-textile projects. If they do not, the knowledge they acquired will not be harmful to them.

Students had the option of being part of the study or not. All the EDLT740 students enrolled in the 2013 class chose to be part of the study and received an invitation to participate in the study (Appendix G), a research information sheet (Appendix H), and signed informed consent forms (Appendix I). Because the EDLT740 course and the research activities were tightly integrated, Table 7 identified, for the students, which aspects were course requirements

and which were parts of the research study. This table was included in the research information sheet and the consent forms.

Table 7

Activity Requirements for Course and Research

Activity	Required for EDLT740	Required for Research Study	Audio Recording (optional)	Images Captured (optional)
Introduction to E-Textiles	X			
6 Debugging Design Scenarios at F2F 1	X		X	X
Self-Efficacy Questionnaire		X		
Computer Experience Questionnaire		X		
Computational Attitude Questionnaire		X		
Student Evaluation of Learning Module		X		
Group Interview		X	X	
Debug Forum	X			
Class E-textile Wiki	X			
Project Notebook		X		
Arduino Project	X			X
Multi-bug Design Scenario at F2F 2	X		X	X

E-textile projects involve the use of needles for sewing and involve small parts such as LEDs and magnets. During the introductory time in the online sessions, the students were informed of basic safety measures they could use with the needles to protect themselves and their families from sharp encounters.

Due to the face-to-face interviews and forum discussions, the participants' identity was NOT anonymous, except in their responses to the student evaluation of the learning module. Although the students were not anonymous for the researcher, she was not involved in grading students. References to students are by pseudonyms in the published study. The course professors were not aware of which students had consented to participate in the study so that their participation or lack of participation in the study would not influence their grades.

EDLT740 teachers were also affected by the research study that was integrated with course. The teachers of the course received additional feedback about their students' understanding of computational concepts due to the availability of the debug discussions and

project notebooks. They also spent additional time because of the study being part of the course, but the additional time was not excessive.

Assumptions

This study was designed based upon the assumptions detailed in the following paragraphs.

E-textiles are a rich computational environment. One of the steps that can support the development of computational thinking is using rich computational environments (Lee et al., 2011). These rich environments are described as "ones in which the underlying abstractions and mechanisms can be inspected, manipulated and customized" (Lee et al., 2011, p. 35). E-textiles combine soft circuits, sensors, and code, allowing multiple types of inspection, manipulation, and customization; therefore, it was assumed that using e-textiles would provide a rich computational environment.

Use-Modify-Create is appropriate for adult learners as well as youth. The three-stage progression described as use-modify-create was proposed specifically as a method for engaging youth with a rich computational environment (Lee et al., 2011). The use-modify-create precepts were applied in this design of the e-textile debugging learning module under the assumption that this model would be efficacious for adult learning as well as youths.

Limitations

One of the limitations of this design study was that there was not an opportunity for a full second iteration of the design within the time limitations of the dissertation. The evaluation of the design here was based upon a pilot study and a proof-of-concept iteration only.

There are both advantages and limitations associated with the characteristics of the student participants of this study. They were learning technologies doctoral students, so they had

a higher than average interest in learning and technology. This means that their reactions and experiences may not be representative of how other graduate education students would experience and react to the learning module. Also, as doctoral students in education, they were used to thinking about and discussing learning. This meant that they were more likely to view their experiences from a meta level, which proved useful as they were interviewed and as they posted to the forum, project notes and blogs, and Wiki. This may not be representative of all non-computer science graduate students.

Learning Module Design

It was anticipated that most of the students in the course would not have prior experience with electricity and circuits. Before the learning module focus was turned to the debugging design scenarios at the F2F meeting, an introduction to general principles of polarity, serial and parallel circuits, was conducted online. The students used materials such as LED's, coin cell batteries, circuit diagrams, alligator clips, and the LilyPad Protosnap board. A copy of the learning module guide for these introductory activities is included in Appendix A.

Lee et al. recommend the use-modify-create pattern for engaging learners with rich computational environments (Lee et al., 2011). The six debug design scenarios allowed the students in the learning module an opportunity to use and modify existing basic e-textile projects. The six bug types were selected based upon the prior work of Fields et al. (2012), which identified the most common types of engineering and coding challenges they encountered in the workshops that they had facilitated. Introducing the common e-textile bugs early in students' learning experience was planned to help them learn essential e-textile design principles as well as debugging techniques that they could use in future projects. The timing of the F2F

session was later than originally anticipated but was beyond the researcher's control. Thus, the benefits of the debugging experience did not occur as early in the semester as was planned.

Learning has a strong social component (Bandura, 1977; Lave & Wenger, 1991; Vygotsky, 1978). Many opportunities for exchanging ideas and sharing projects were built into the learning module. The debugging forums, the class Wiki, and the initial design scenario pair-work were ways that the learning module encouraged the social aspect of learning e-textiles. Another facet of the social component of learning e-textiles and Arduino is the active Maker community. The Arduino community has created many helpful sites for learning techniques and project inspiration. At http://lilypadarduino.org/ there are links to sewing tutorials and other helpful sites. Leah Buechley, designer of the LilyPad modules, has posted an introduction to programming the LilyPad Arduino that was helpful to the students as they began experimenting with e-textiles and the Arduino programming environment. For example, the introduction at http://web.media.mit.edu/~leah/LilyPad/03_arduino_intro.html has information about installing and using the Arduino environment and instructions for selected projects. The participants in the EDLT740 course were encouraged to involve themselves with the active online e-textile and Arduino community and to utilize resources that they found on the web. In addition, there was a course requirement that the students visit a Maker space.

One of the purposes for including a course requirement to create an e-textile or Arduino project of the student's own design was to allow the students to feel one of the design perspectives, expression. While the debug scenarios may have been with projects that the students did not find particularly interesting, the opportunity to create their own project allowed them the opportunity to tap into their interests and creativity. There was a wide-ranging selection of projects that the students created and presented in their project videos. These

projects are discussed in the Chapter 5. In addition, while creating their projects the students had the opportunity to construct knowledge about electronics and programming.

Table 8 summarizes some of the design decisions that were incorporated into this learning module.

Table 8

Design Decision Chart

Design Aspect	Reason	Related Literature	Questions and Considerations
E-Textile Projects to develop computational thinking	Build and play with items rooted in contemporary culture Low threshold, high ceiling Rich computational environment	Papert & Harel (1991) Repenning, Webb, & Ioannidou (2010) Lee et al. (2011)	
6 Bug Types: Engineering Challenges • Short Circuits • Electronic Topology • Polarization Coding Challenges • Constant versus variable pins • Control flow • End-State definition	Identified as design challenges in prior research.	Fields, Kafai, Searle, & Min (2012)	One bug per textile project in initial design scenarios. Final debug design scenario incorporates all six.
Use design scenarios to present bugs	Encourage thinking around solutions BEFORE coding	Grover & Pea (2013)	
Choose project designs that are not strongly gendered	Course has both males and female students. Computational Thinking Tools Checklist –support equity.	Repenning, Webb, & Ioannidou (2010)	Survey question about impact of bug designs.
Focus on debugging, which will be part of every project	Support further learning	Brennan & Resnick	
Encourage students to use Maker community resources	Support further learning	Brennan & Resnick	

(continued)

Design Aspect	Reason	Related Literature	Questions and Considerations
Students design and make individual project.	Construction of knowledge with construction of personally meaningful artifact	Kafai & Resnick	Students can pick their own designs but will have certain design considerations that need to be met, such as required to have some aspect of interactivity.
	Use-modify-create	Lee et al.	
	Designing own project will allow the students to express themselves	Brennan & Resnick (2012)	
Design Scenario with multiple bugs at end of unit	Incorporate Artifacts	Brennan & Resnick	
Interview students about design processes	Illuminate processes	Brennan & Resnick	
Students keep a project notebook or blog	Illuminate processes	Brennan & Resnick	
Debug design scenarios and interviews at beginning and end, discussions, blog and Wiki in middle	Check in at multiple waypoints	Brennan & Resnick, Wang & Hannafin	
Students debug existing projects and create a project of their own design	Value multiple ways of knowing	Brennan & Resnick	Do they help other students debug their own projects?
	Use-modify-create	Lee et al.	
Look at other students' comments about projects, Teachers views of projects	Include multiple viewpoints	Brennan & Resnick	

Chapter Summary

This chapter has elaborated the methods that were used to design and evaluate an e-textile debugging learning module for students in a learning technologies doctoral program. An emphasis on introducing the material following the use-modify-create model guided the order of the activities, and opportunities for students to engage with others were built into the module. Multiple data sources were examined for computational thinking concepts, practices, and perspectives to determine if the learning module did provide a rich environment for developing computational thinking. Design-based research methodology guided the study. A pilot study and proof-of-concept iteration are covered by this dissertation, and future research will continue to refine the development of the learning module.

Chapter 4: Results

Overview

This study is a design-based project in which a set of computational thinking activities was designed and created to lead non-computer science graduate education students into engagement with computational thinking. The initial module design was based on work and concepts covered in the literature review (Chapter 2). The activities were conducted with a graduate class of 18 students pursuing a doctorate in education.

Data were gathered from those students to determine the success and the obstacles in getting to computational thinking. The analyses of those data are presented in the following sections. Results of the findings are discussed in Chapter 5, in terms of their implications for the revision of the activities set and for the study of computational thinking. A revised set of the activities, based upon analysis of the first iteration, is described in Chapter 5. Chapter 5 also contains a discussion of the issue of computational thinking for non-computer science graduate students in education and wider implications of the study.

Computational Thinking Activities

To briefly recap, the following tasks comprised the first version of the computational thinking activities set:

1. Design scenarios for three e-textile engineering challenge bugs (Bugs 1 to 3) that the students worked on in pairs. Each design scenario contained a single bug. The students worked on these debug challenges at the first F2F meeting. Audio recordings were created of the pairs' interactions, and Bug 1 was selected for analysis.

2. Design scenarios for three e-textile coding challenge bugs (Bugs 4 to 6) that the students worked on in pairs. The students worked on these debug challenges at the first F2F meeting. Two of the coding challenges had a single bug, but one coding challenge (Bug 5) had multiple bugs because of unintended errors in the code. Audio recordings were created of the pairs' interactions, and Bug 5 was selected for analysis.
3. An Arduino project of the student's choice that was presented mid-semester through a student-made video. Students were encouraged to reach out to maker communities for solutions to project difficulties. Students brought their partially completed projects to the first F2F meeting.
4. A design scenario for an e-textile project containing a combination of the three engineering and three coding bugs previously introduced (Bug 7). Students worked on the design scenario in pairs at the second F2F meeting. Audio recordings were created of the pairs' interactions and were analyzed.

Further insight into the students' experiences with computational thinking was gained by examining additional course elements. In varying degrees, students participated in debug and other project forums within their course shell. They also wrote blogs about their projects. A Wiki was available within the course shell but was not highly utilized. The researcher attended the course online synchronous sessions where projects were sometimes discussed. Another communication channel used by the students was their cadre facebook group. At the end of the course, students wrote reflection papers about their experiences in the course.

Data Collected to Determine Impact

As described in greater detail in Chapter 3, the following measures were used to gather data to indicate the degree of success the activities were having in engaging students in computational thinking:

1. Computer Experience Survey – See Appendix B for a copy of the survey. Respondents entered names. The results of the Computer Experience Survey are included in the description of the participants in Chapter 3.
2. Computer Attitudes Survey (Pre and Post) – See Appendix D for a copy of the survey. Respondents entered names. The results are discussed in this chapter under the section Appeal to non-computer science majors.
3. E-Textile Self-Efficacy Survey (Pre and Post) – See Appendix C for a copy of the survey. Respondents entered names. The results are discussed in this chapter under the section Confidence in computing abilities, the E-Textile self-efficacy questionnaire subsection.
4. Evaluation of the Learning Module Questionnaire – See Appendix F for a copy of the questionnaire. Responses to the Learning Module Evaluation questions were anonymous. The results are discussed in this chapter under the section Evaluation of the learning module.
5. End of Course Reflection Papers – Posted in the course forums, names included. Examples from the papers are discussed in various sections of Chapters 4 and 5.
6. The Computational Thinking Rubric for Examining Students' Project Work – Applied by researcher to analysis of pair conversations during e-textile debugging activities and course forum posts. Bugs 1, 5, and 7 were selected for analysis based on

the type of bug (Bug 1 was an engineering bug, Bug 5 was a coding bug, Bug 7 had both engineering and coding bugs) and time when the students worked on them (Bugs 1 and 5 were done at the first F2F meeting, Bug 7 was given at the second F2F meeting). Sample questions from the rubric can be found in Chapter 3 and a complete copy of the rubric is also included in Appendix E.

7. Group Debrief Discussion – A group discussion conducted by the course professor after the students completed the first six debug activities.

Data Analyses

The data analyses that follow are presented in relation to the study's desired key outcomes. These key outcomes are:

- Evidence of computational thinking concepts, practices and perspectives
- Increased student confidence in computing abilities
- Appeal to non-computer science majors

Evidence of Computational Thinking

One of the key outcomes looked for in this design research study was evidence of computational thinking concepts, practices, and perspectives in student interactions as they participated in the e-textile activities. The following sources were examined for evidence of computational thinking:

- Pair interactions during the debugging exercises
- Group debrief comments
- Student posts in the forums
- End-of-term reflection papers

CT Evidence in Debugging Activities

This section presents evidence of computational thinking concepts, practices, and perspectives in student interactions as they participated in the e-textile debugging activities. The EDLT740 course in which the students were enrolled was offered in a blended learning format. The students had weekly synchronous online sessions, participated in online forums, and met twice during the semester for F2F sessions that lasted five days. The F2F days were shared among the three courses in which the students were enrolled.

The photo in Figure 35 shows five of the nine groups engaged in the first set of debugging exercises. At the first F2F meeting, each group received a packet of six e-textile bugs, a LilyPad, a Livescribe pen and notebook, and USB cord. Students also had their Arduino kits, which contained coin cell batteries, an FTDI connector, small conductive magnets, LED's, a set of alligator clip cords, needles, and conductive thread. Students had previously installed the Arduino platform on their laptops. At the second F2F, when the students worked on Bug 7, the first six bugs were removed from the packets, and FTDI connectors, alligator clips, USB cord, scissors, needles, and conductive thread were added to the packets, since the students did not have their own Arduino kits with them.

Figure 35. Groups working on debugging activity.

Pair work of students working on Bug 1, Bug 5, and Bug 7 was transcribed for each of the nine groups and coded for the computational thinking concepts, practices, and perspectives

identified by Brennan and Resnick (2012). During the transcription and coding phase of data analysis, certain types of computational thinking not specifically mentioned by Brennan & Resnick (2012) emerged and have been included in the data report. These are identified by an * next to the name. These emergent computational thinking principles and perspectives are presented in this chapter, and their significance is discussed in Chapter 5. Chapter 3 discusses the transcription and coding methods utilized.

While pair work on all seven bugs was recorded, only three bugs were selected for coding. After listening to all recordings, Bug 1 was selected because it was representative of the three bugs that were engineering challenges. These bugs focused on engineering challenges with the electronic circuits and did not involve programming. Bug 5 was included for coding because it contained multiple programming challenges; additionally, it was the bug that proved most difficult for the students. Finally, Bug 7 was included because it was the bug the students worked on at the second F2F meeting, after they had completed their own personal Arduino projects. Bug 7 combined all six of the previous bug types -- three engineering challenges and three coding challenges.

The following section of computational thinking evidence is grouped by bug.

Bug 1. The engineering challenge in Bug 1 was a short circuit. The LilyTiny that controlled the circuit behavior was preprogrammed to have four different blinking and flashing behaviors, one on each of the four numbered pins. The signals to the pins were sent randomly. Pictures of Bug 1 are shown in Figure 36. In the close up picture the short circuit caused by the crossing threads is outlined. The associated diagram for Bug 1 is shown in Figure 37, since the stitching on the patterned background of the fabric is sometimes hard to distinguish. Students did not have the diagrams showing the circuit lines for the bug in their design scenario instructions,

but they did have a diagram showing the LED placement and the LilyTiny, with a description of the desired behavior. See Appendix J for the design scenarios that the students received. In the diagram below, the short circuit is near the letter A. The red lines in the diagram represent positive connections and the black lines are negative connections.

Figure 36. Bug 1, short circuit.

The researcher expected that the students would debug and test, one of the computational thinking practices, while working on Bug 1, as well as use the electronics principles of establishing circuit correctness and demonstrate understanding that crossed threads cause short circuits. In addition, she expected to see the students using the visibility of the behavior to aid in their debugging.

Bug 1 CT coding. Table 9, Bug 1 Computational Thinking Coding Summary, contains a summary of the coded computational thinking concepts, practices and perspectives observed in Bug 1. It is organized in descending order of numbers of observations. Because of the specific application of computational thinking in e-textiles, electronics principles were also coded. The table contains only the observed instances. The column 'Partnerships Where Found' shows the groups where the code was present. As mentioned before, the Testing and Debugging group was divided into sub-practices. A complete rubric containing all concepts, practices, and perspectives is available in Appendix E.

Figure 37. Bug 1 schematic.

Circuit power and polarity. The most frequently coded observation was students discussing and using electronics principles of circuit power and polarity.

Table 9

Bug 1 Computational Thinking Coding Summary

Code	Group	Subgroup	Partnerships Where Found	Count
* Circuit Power, Polarity	Electronics Principle		Groups1, 2, 3, 4, 5, 6, 7, 8	32
* Establish Circuit Correctness	CT Practice	Testing and Debugging	Groups1, 2, 3, 4, 5, 6, 7, 8,9	25
* Verify Correct Outcome	CT Practice	Testing and Debugging	Groups1, 2, 3, 4, 5, 6, 7, 8, 9	18
* Crossed Threads Cause Short Circuit	Electronics Principle		Groups1, 2, 3, 4, 5, 6, 7, 9	18
* Independent Behavior Requires Different Pin	Electronics Principle		Groups1, 2, 3, 4, 5, 7, 8, 9	14
* Documenting	CT Practice		Groups1, 3,6,7, 9	13
* Plan Debugging Strategy	CT Practice	Testing and Debugging	Groups1, 3, 6, 7, 9	12
* Refer to Specifications	CT Practice	Testing and Debugging	Groups1, 2, 3, 4, 6, 7, 9	10
Testing and Debugging, general	CT Practice	Testing and Debugging	Groups 1, 2, 5, 6, 8	10

(continued)

Code	Group	Subgroup	Partnerships Where Found	Count
Incremental and Iterative	CT Practice		Groups 2, 4, 6, 8, 9	9
* If Connected, Same Polarity	Electronics Principle		Groups 2, 3, 6, 7	8
* Predict Expected Behavior	CT Practice	Testing and Debugging	Groups 1, 2, 3, 4, 6	7
* Benchmark	CT Practice	Testing and Debugging	Groups 2, 6, 7, 8	6
* Use Visibility of Behavior	CT Practice	Testing and Debugging	Groups 1, 4, 6	5
Parallelism	CT Concept		Group 2	1
* Able to Adapt to Others Code	CT Perspective		Group 1	1
Expressing	CT Perspective		Group 1	1
Connecting	CT Perspective		Group 6	1
* Start Over Again	CT Practice	Testing and Debugging	Group 4	1
* Control Lights by Varying High and Low Signal	Electronics Principle		Group 4	1
TOTAL				216

Note. * Indicates coding category not found in the original literature-derived list of computational thinking elements.

The following dialogue is an example of the types of conversations that occurred concerning circuit power and polarity. In the participant aliases that are used, if the alias begins with an M it is a male student and if the alias begins with an F, it is a female student. The two digits following the initial letter refer to the group number they are in. The final two digits represent which of the students in the pair they are. These final numbers were assigned based on who spoke first in the first transcript of the group.

> M0502: This is my question…OK. There’s a plus . . . there’s a negative.
> F0501: OK. So that’s right.
> M0502: And this one is connected to this and the negative. Which is right.
> F0501: Mmm-hmm.
> M0502: But why is this still here?
> F0501: See, that’s the negative…you see how all the negatives are connected together? Which is fine.
> M0502: Yeah.

F0501: They can all be connected together.
(Group 5, Bug 1 Transcript)

An additional example of a conversational snippet where a student thinks aloud about power and polarity is given below:

M0301: So, pin two . . . is . . . sending power to B and C.
M0301: Which is good. That is what we wanted it to do.
M0301: Uhhh . . . and then goes back to ground, but . . . it also . . . appears to be going here to D.
M0301: And I don't understand why it would be doing that.
M0301: That doesn't make any sense to me.
M0301: Oh, OK. No, wait a minute. I'm . . . I'm seeing it now. The –
M0302: It's going through here.
(Group 3, Bug 1 Transcript)

As a result of coding observations, the testing and debugging computational thinking practice was expanded to have several sub-practices. These include establishing circuit correctness, verifying the correct outcome, planning debugging strategy, referring to specifications, predicting expected behavior, benchmarking, using the visibility of the behavior, starting over again. Combining the observations for these sub-practices gives a total of 94 of the 216 coded behaviors in Bug 1 (43.5%) in the testing and debugging category. This is not surprising, given that the design scenarios were about debugging existing materials.

Establishing circuit correctness. Establishing Circuit Correctness was the second highest individual code found across the transcripts of Bug 1. The following exchange between the two students in Group 1 illustrates the practice.

M0101: So, let's see if something is wrong with the circuit.
F0102: Do they behave differently than the other ones do?
F0102: They do, don't they?
M0101: Let's look over here on this side -
F0102: These behave differently than those, right –
M0101: Look over here. Where does this thing go?
. . .
M0101: This one . . . is going –
F0102: Is it negative or positive? –

M0101: It's . . . it's going to the negative, but then for the positive . . .
F0102: That's the problem.
M0101: It doesn't go . . . It looks like it crosses this.
M0101: Doesn't it look like it crosses?
M0101: And it touches?
F0102: Yeah, maybe.
(Group 1, Transcript Bug 1)

Verifying solution. The following conversational snippet shows students verifying that their solution had the desired behavior:

M0301: We're just looking to make sure, and we are . . .
M0301: So, we're verifying that, yes, we do in fact have different behavior on A, on B and C, and on D and E.
(Group 3, Bug 1 Transcript)

Predicting expected behavior. The code 'Predicting Expected Behavior' was used when the students reasoned that certain actions would cause a particular behavior then looked for verification. The following conversational snippet shows a student predicting expected behavior.

F0601: All right! So, we've got two lighting up…one inside the flower…and one on our drink! So, you can see right here…this isn't connected…the positive to the positive. So that's our first…sewing error. I think my…let's try this guy…this guy…to this guy…see those should light up…no matter what. OK. Those could not be lighting up because they're touching something?
(Group 6, Bug 1 Transcript)

Documenting. A computational thinking practice not specifically discussed by Brennan and Resnick (2012), but which emerged during coding, was documenting findings and observations. Students documented their findings and changes they were making in their notes and in statements made in the audio recordings. Later, while working on the coding challenge bugs, students also demonstrated the practice of documenting by making comments in the code.

The following dialogue illustrates a pair of students documenting a finding:

M0701: I think it is that . . . because there is that thing . . . so . . . I guess with the circuits . . . I guess the debugging process is . . . always follow the loop . . . from like one pin . . . follow a loop out through the components of the loop and then follow the

loop back in . . . so that was our debugging process . . . follow the loop . . . we saw that there was the crossing circuit there.
F0702: Yeah…should I write that down? Well, I think…
M0701: I guess it's already recorded, right?
F0702: Yeah. It's recorded . . . So, should we fix that right now?
(Group 7, Bug 1 Transcript)

Incremental and iterative. The practice of being incremental and iterative is one that is fundamental to computational thinking. Even in the bugs featuring only engineering challenges, the practice of being incremental and iterative played a role. A pair of students is engaged in the practice of being incremental in the following conversational exchange:

F0801: OK. Tell me what to do next after I connect it.
F0802: OK. Do we need to take all of these out?
F0801: Well, I'm just going to test this one first.
(Group 8, Transcript Bug 1)

Independent behavior, different pins. An electronics principle that is frequently used in e-textiles is that if lights are going to have different behaviors they need to be connected to distinct pins. This was coded as 'Independent Behavior Different Pins'. This principle was observed 14 times across the different groups during their work on Bug 1. The students in the following interchange use this electronics principle. In the interchange below, student F0202 begins by reading from the design scenario.

F0202: OK. For the tropical drinks e-textile, Vicky decides that she wants the led at point A to have a certain behavior.
F0202: Which one is the A . . . on this thing?
F0202: That's one. So . . . that's A.
F0201: A's right there.
F0202: And, then, the led at point B and C to behave the same.
F0201: This is B and C.
F0202: Right here. Okay.
F0202: And then . . . but different from Led at point A.
F0202: LED's at point D and E to be as the same as each other, but different from all the others.
F0202: So, we need to have three groups.
(Group 2, Bug 1 Transcript)

Visibility of behavior. While the researcher expected one of the most frequent practices would be using the visibility of the behavior, only five instances were coded across the nine groups working on Bug 1. The following dialogue is one of the instances:

> F0401: Is that the only one that is crossing something?
> F0402: I only see one crossing.
> F0401: OK. So…this is wrapping around to this one…where is this one going? To that one? Do we just override it? Or…let's see if it works…I just pulled that up a bit…
> F0402: I just saw that one come on…do it again…
> (Group 4, Bug 1 Transcript)

While only a few overt instances of using the visibility of the behavior were coded, it is quite likely that the students were basing many of their debugging decisions on what they observed as they made changes.

Bug 1 was an engineering challenge with a short circuit. Transcripts revealed that students were highly engaged in the computational practice of testing and debugging. Eight testing and debugging sub-groups emerged and were coded, with the most common being establishing circuit correctness and verifying the correct outcome. Additionally, the computational thinking practice of being incremental and iterative was evident. Students were observed using electronics principles such as circuit power, polarity, and electronic topology. As expected with this engineering bug where the students did not interact with programming code, many computational concepts and practices were not observed.

This bug was a good choice for the first of the design scenarios that the students experienced because the bug was relatively simple to locate. The short circuit was on the front side of the project and most groups were able to find and correct it without much difficulty. Students eliminated the short circuit by cutting the thread, then reconnecting by sewing or using alligator clips. One innovative group used a coffee stirrer to separate the positive and negative threads, as shown in Figure 38.

Figure 38. Bug 1 with coffee stirrer.

The transcripts revealed that some of the difficulties experienced with Bug 1were due to lack of familiarity with the materials and uncertainty about the directions. These aspects of the learning module will be considered as suggestions for the next iteration of the learning module, which is discussed in Chapter 5.

Bug 5. The coding challenge for Bug 5 was designed to be control flow, as named in Fields et al. (2012). The control flow error was created by not initializing the positive side of an LED that was not connected in any other way to positive current. This control flow error was created in Bug 5, however several additional, unintended bugs, were introduced due to some last-minute changes to the code. Bug 5 proved to be the most challenging for all the groups, due to multiple factors: the difficulty of the original control flow error itself (it was the least solved challenge in the original work of Fields et al. (2012)), the unintended bugs, the inclusion of a modulo function with which the students were unfamiliar, and the fact that multiple LED actions were occurring at the same time, rather than in sequence. Although the coding of the fmod

modulo function was correct and didn't need to be altered, its complexity affected the students' ability to debug the program. Additionally, the simultaneous but varied behavior of the different LED's increased the complexity even further. A copy of the full program used in Bug 5 can be found in Appendix K. The corrected errors are commented in the corrected version.

The researcher expected the students to be able to focus in on the one error that was causing the problems, but the introduction of the labeling errors made isolating the error was more difficult. In addition, the modulo function, fmod, caused the students to spend more time as they tried to figure out how it functioned. The comments in the code were not enough to help most students understand how it functioned.

One other factor that made Bug 5 difficult was that there were several simultaneous actions taking place on different LED's. While the physical layout of the debugg'em in Fields et al. (2012) was available, the exact code that was used was not. The code used for the debug activities was written to meet the described coding challenges. Since there were three coding bugs in the study, the researcher tried to create three different lighting patterns to make it more interesting. This resulted in the unintentionally difficult Bug 5.

The first of the images in Figure 39 shows the electronic wiring that was used in Bug 5, and the second image shows is what was included in the student design scenario handout. In the circuit, the red color is used for the positive current and the black indicates the negative current. Letters identified the LED's, but the circuits were not indicated.

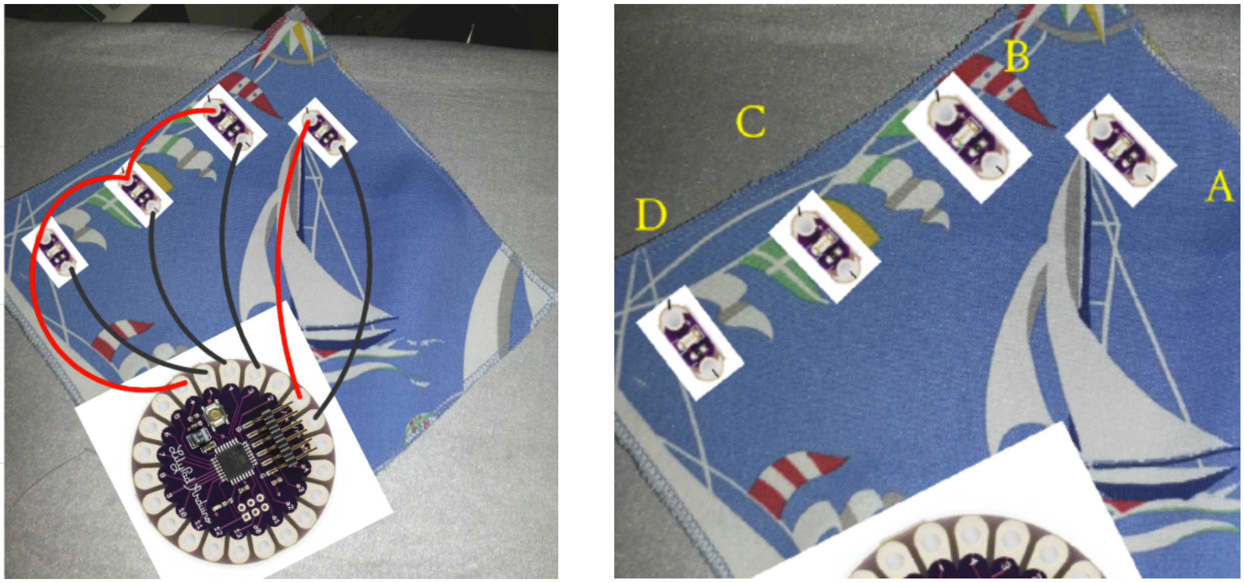


Figure 39. Bug 5 schematics.

Bug 5 CT coding.

Table 10, Bug 5 Computational Thinking Coding Summary, contains a summary of the coded computational thinking concepts, practices and perspectives observed in Bug 5. As in the similar chart for Bug 1, it is organized in descending order of numbers of observations. Comparing the total of 216 codes for Bug 1 to the 646 codes for Bug 5 gives an indication of the increased difficulty and additional time with Bug 5. Also, as with Bug1, only the instances that were coded in Bug 5 are in the table. See Appendix E for the complete rubric.

Table 10

Bug 5 Computational Thinking Coding Summary

Code	Group	Subgroup	Partnerships Where Found	Count
* Debug Strategy	CT Practice	Testing and Debugging	Groups 1, 2, 3, 4, 5, 6, 7, 8, 9	65
* Benchmark	CT Practice	Testing and Debugging	Groups 1, 2, 3, 4, 5, 6, 7, 9	42
Reusing and Remixing	CT Practice	Testing and Debugging	Groups 1, 2, 3, 4, 5, 6, 7, 9	41

(continued)

Code	Group	Subgroup	Partnerships Where Found	Count
* Predict Expected Behavior	CT Practice	Testing and Debugging	Groups 1, 2, 3, 4,5, 7, 9	40
* Control Leds High Low	Electronics Principle		Groups 1, 2, 3, 4, 5, 7, 9	40
* Use Visibility of Behavior	CT Practice	Testing and Debugging	Groups 1, 2, 3, 4, 6, 7, 9	36
* Refer to Specifications	CT Practice	Testing and Debugging	Groups 1, 2, 3, 4, 5, 6, 7, 8, 9	36
* Documenting	CT Practice		Groups 1, 2, 3, 4, 5, 6, 7, 8, 9	35
* Walk Through Code	CT Practice	Testing and Debugging	Groups 1, 2, 3, 4, 5, 6, 7, 9	31
Being Incremental and Iterative	CT Practice		Groups 1, 2, 3, 5, 6, 7, 9	30
Conditionals	CT Concept		Groups 1, 4, 7, 8, 9	22
Loops	CT Concept		Groups 1, 2, 3, 5, 6, 7, 9	22
Operators	CT Concept		Groups 3, 4, 5, 6, 7, 9	22
Testing and Debugging, General	CT Practice	Testing and Debugging	Groups 1, 2, 3, 5, 6, 7, 9	22
Abstracting and Modularizing	CT Practice		Groups 1, 2, 3, 5, 6, 7, 8, 9	20
* Physical Setup Determines Code	Electronics Principle		Groups 1, 4, 5, 6, 7, 9	20
* Verify Correct End	CT Practice	Testing and Debugging	Groups 1, 2, 5, 6, 7, 9	16
* Start Over Again	CT Practice	Testing and Debugging	Groups 1, 3, 4, 6, 7	15
* Establish Circuit Correctness	CT Practice	Testing and Debugging	Groups 1, 3, 4, 5, 8, 9	14
* Power, Positive Negative	Electronics Principle		Groups 1, 3, 4. 9	12
Sequences	CT Concept		Groups 2, 3, 4, 5, 6, 7	11
Connecting	CT Perspective		Groups 1,3,4, 5, 8, 9	8
* Plan Debugging Strategy	CT Practice	Testing and Debugging	Groups 1, 3	8
Data	CT Concept		Groups 4, 7, 9	5
* Need Delay to See Behavior	Electronics Principle		Groups 4, 5	5
* Able to Adapt to Others Code	CT Perspective		Groups 5, 9	4
* If Connected, Same Polarity	Electronics Pr.		Groups 4, 5, 8	4
Expressing	CT Perspective		Groups 6, 7, 8	3
Parallelism	CT Perspective		Groups 5, 7	2
Questioning	CT Perspective		Group 8	2
* Many Ways to Solve	CT Perspective		Group 9	2
* Independent Behavior, Different Pins	Electronics Principle		Group 4	1
TOTAL				636

Note. * Indicates coding category not found in the original literature-derived list computational thinking elements.

Debug strategy. For Bug 5, the most frequently coded observation was Debug Strategy. Debug Strategy is one of the categories that emerged as a further breakdown of the Testing and Debugging practice identified by Brennan and Resnick (2012). The following explanation by a member of Group 3 illustrates an example of the code Debug Strategy recorded during Bug 5 work.

> M0302: So that didn't work.
> M0302: What we were trying to do is to find some kind of symmetry with the code, to make sure that everything was accounted for.
> M0302: I am assuming that the stitching is all legit, and we don't have to jump anything.
> (Group 3, Bug 5 Transcript)

When different groups would get stuck and ask for assistance, that assistance was usually provided in the form of suggesting debug strategies to try. This is illustrated in the following conversation between PKIM, a hardware engineer who was assisting with the logistics of the design scenarios, and the members of Group 5.

> PKIM: OK. So, the LED has a positive and negative side, right?
> M0502: Uh hum.
> PKIM: So just to focus on getting it to turn on –
> (Group 3, Bug 5 Transcript)

The students in Group 8 use labeling the LED's as one of their debug strategies, as shown in the conversation snippet below.

> F0801: I've got to label this stuff.
> F0802: Oh, yeah.
> (Group 8, Bug 5 Transcript)

The image shown in Figure 40 is of Group 8's labels; they put tape next to the LED's and marked the tape with the identifying letter used in the design instructions and the behavior that the light was supposed to have. This was a useful debugging strategy which helped Group 8 avoid the necessity of repeatedly going back to read the instructions to remind them of the desired behavior of the LED's, as observed in most of the other groups.

Figure 40. Bug 5 with tape labels.

The members of Group 1 propose different debug strategies for beginning to find the error in Bug 5.

> F0102: So, we are looking at the program now. So, let's see if the program's right.
> F0102: You want to test this out first, or you just . . . We could probably just look at the program.
> M0101: Let's test our connections, first.
> (Group 1, Bug 5 Transcript)

One of the strategies used by the members of Group 7 was to make small changes and see the outcome.

> M0701: Let me just . . . Let's try with those little changes first.
> F0702: OK.
> M0701: And then let's see what happens.
> (Group 7, Bug 5 Transcript)

Benchmark. Benchmark is the second most frequently coded observation in Bug 5. Benchmark was coded when the students worked to establish a point of reference for understanding how something worked or where they were in their process.

In the interchange that follows, we see Group 6 setting a benchmark about the starting state of Bug 5.

<checking with the other group>
F0601: Hey, guys? When you got five -
Other Group: We haven't gotten five. We are starting on it.
F0601: When you got it out of the bag, and stuck on your Arduino thing, did the mast light up?
Other Group: It did not light up.
F0601: OK. I just wanted to be sure that is how it started for you, too.
Other Group: It did not light up out of the bag.
(Group 6, Bug 5 Transcript)

The following conversation from Group 9 illustrates the students benchmarking where they are in the debugging process.

M0902: All right. Let me read back what it's supposed to be doing.
M0902: The bottom flag to stay on.
M0902: So that is staying on. That's one's fine.
M0901: That flag's good.
M0902: Middle flag to blink quickly.
M0902: It's quick enough.
M0902: Top flag to blink slowly, which it's not doing.
M0901: Not doing it.
M0902: Mast to blink twice, then pause, then repeat.
M0902: It's not doing that.
M0901: It's not doing that, either.
M0901: OK. All right.
(Group 9, Bug 5 Transcript)

In their efforts to understand the fmod function that was controlling the patterned blinking, the members of Group 3 used comments to isolate parts of the code, and benchmark the behavior.

M0301: I mean I… Let's look at it.
M0302: Where it says every third time the LED . . .
M0302: So if you took the comment off of this, what would happen?
M0302: Good. So that seemed to remove any kind of cycle on this at all.
M0302: So, if we, mmm . . . So, we know that that code controls this.
(Group 3, Bug 5 Transcript)

As this same group finishes up Bug 5, they talk about the experience and reference their benchmarking strategy.

> M0302: Oh man, that was brutal.
> M0301: That was brutal.
> M0301: So, let's upload that.
> M0302: That was a head banger on that one.
> M0302: I mean, we had the right idea, though. I feel good about that. And we were able to isolate it and switch some things around.
> M0301: That was one key I think was isolating the different parts of the code.
> (Group 3, Bug 5 Transcript)

Reuse and remix. Brennan and Resnick's (2012) computational thinking practice of Reusing and Remixing was the third most coded item in Bug 5. Reusing and Remixing was frequently used with Bug 5, as students copied chunks of code for one LED to another LED. The following suggestion of a Group 5 member illustrates this practice.

> F0102: So <partner name>, what if we copy and paste this code . . . For the same
> F0102: Why don't we get this to work the same as this, for now, and then we can start . . . playing with this code to make it stop.
> (Group 1, Bug 5 Transcript)

Not only chunks of code within the same program were reused and remixed, but other programs were also referenced, as shown in the following conversation of Group 2. They are referring to Blink, a sample program included in the Arduino example library.

> F0201: So, what do we have on 19 to do that we don't have on 17?
> F0202: Slow…
> F0202: I am going to open Blink
> F0202: I am going to try to open Blink.
> F0202: Do we want to do this?
> F0202: digitalWrite (LED, HIGH). Delay.
> F0201: Yeah, do that. Yeah. Copy that. Yeah.
> (Group 2, Bug 5 Transcript)

Predict expected behavior. Predict Expected Behavior is another frequently occurring code in Bug 5. This is another of the codes that is a refinement of the Testing and Debugging computational thinking practice from Brennan and Resnick (2012). The code is used when the

students predict the behavior their code changes will cause, or when they predict what type of coding or circuit conditions would cause a certain behavior. The following interchange between Group 3 members is an example where the students are expecting a particular behavior in response to their changes. This leads to more consideration about what could be wrong, and ultimately to the solution of that part of the debugging challenge.

M0302: So, what I can't figure out is how is that when we added 16 and put turn LED on, there was no affect.
M0301: Right. That is kind of (unintelligible) -
M0302: That doesn't even make sense.
M0302: That should have at least made umm . . .
M0301: Oh, Wait a minute . . .
M0302: Did we combine all that?
M0301: Wait a minute, wait a minute, wait a minute.
M0301: Led 11 is low,
M0302: OK.
M0301: Led 16 is low.
M0301: That means that there is no power going into that circuit at all.
M0302: So one of them should be High?
M0301: So 16 -
M0302: 16 should be High.
M0301: Let's try that.
M0301: Yeah!!! OK -
M0302: We got it going. So-
(Group 3, Debug 5 Transcript)

In the following exchange between Group 4 members and PKim, PKim helps the students predict expected behavior so they can figure out the bug.

PKIM: Can you show me what your fix was that you've made so far?
F0402: Yeah, we added –
F0401: Oh, God. We changed this output to input.
PKIM: Oh, that might be what's causing the dim thing, changed –
F0402: Back to output?
PKIM: Well I don't know the exact solution right now, but something like that might cause the . . .
(Group 4, Bug 5 Transcript)

Abstract and modularize. While opinions of what constitutes computational thinking vary, Abstracting and Modularizing seems to be one of the commonly accepted elements across

different researchers (Polin, 2012). In Bug 5 there is evidence of the students using abstraction and modularization.

M0302: Floating point module. I don't know what that …
M0301: Wow. This is completely f'ing Greek.
M0301: Uhhh . . .
M0302: Well, what if we don't need to … know what it is? Is there a way for us to just say, "OK, let's say it does something." It's a piece of code that does something, and then can we…
M0302: I am just talking out loud.
M0301: Yeah, yeah. I know. I hear you.
M0302: And then can we just see if it affects any of the other ones and if it is identical to that . . .
(Group 3, Bug 5 Transcript)

The conversational snippet below shows students of Group 9 using modularization to make steady debugging progress on Bug 5.

M0902: Can't we do something, so we don't mess with these 3, because they're doing the right thing.
M0901: Yeah, yeah.
M0902: I would prefer not to mess with what we just did . . .
(Group 9, Bug 5 Transcript)

Group 1 takes the matter of modularizing even further. They modularize even the visual aspect of Bug 5 by covering up all the LED's, except the one they are focusing on.

F0102: Let's look at the behavior of the oth . . . of the LED's.
F0102: Now this . . .
F0102: OK.
F0102: Let's look at the each behavior of the LED.
F0102: Then we'll know to switch the code.
F0102: OK. So I am going to cover this up, so we can just see one code at a time. Is that OK with you?
M0101: Yes.
F0102: I'll get it ready, here.
(Group 1, Bug 5 Transcript)

Connect. While the computational thinking perspective of Connecting is only coded eight times in Bug 5, perspectives are one of the harder things to capture (Brennan & Resnick,

2012). Thus, examples of computational thinking perspectives can be interesting because of their rarity.

The following exchange shows students making connections to readings and discussions their class had had about Maker Culture.

> <R, from other group>: You guys are totally breaking stuff.
> F0802: I mean, aren't we supposed to be?
> F0801: That's what making is.
> F0802: Making is breaking.
> <R>: That's Elizabeth's Arduino model.
> F0802: Yeah, it is.
> (Group 8, Bug 5 Transcript)

Able to adapt to others' code. A computational thinking perspective that emerged during coding was the ability to adapt to others' code. The stitching and coding on Bugs 4, 5, and 6 was created to match the circuit layout of the debugg'em in Fields et al. (2012). The circuits were stitched so that the positive terminals were chained together, and the LED's were turned on and off by sending Low and High signals to the pins associated with the negative terminals of the LED's. This was contrary to the way that the students first learned about controlling LED's, which was to keep the negative side of an LED steadily low, and then to send high and low signals to the positive terminals. The different set-up was brought to the students' attention and discussed at the beginning of the debugging exercises, but students struggled in varying degrees with changing their previous method of interacting with the LED's. Some were able to adapt to the code and physical set-up, as given, while others went so far as cutting all the stitching and rewriting the code. The image in Figure 41 is a diagram that was used to explain the two different methods for turning on and off the LED's.

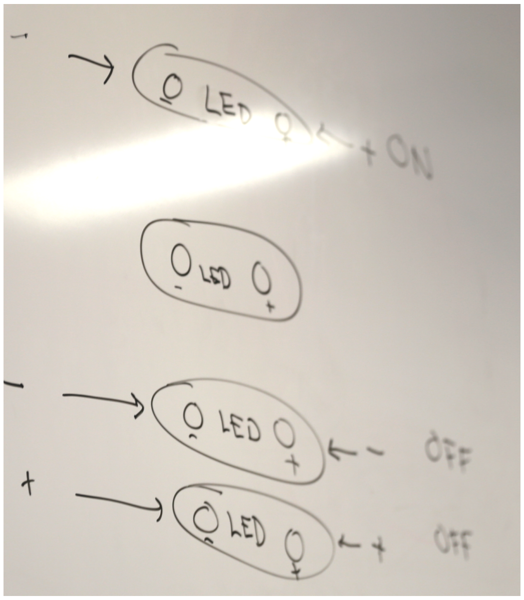


Figure 41. Explaining the ways to turn LEDs on and off.

The wiring set-up for Bug 5 is shown again in Figure 42, for reference. The red lines represent positive and the black lines represent negative.

The members of Group 9 dealt with the code and set-up as it was given to them but remarked that it made the process slower for them.

> (VKim explains the set-up with the High/Low signals to the negative terminal.)
> M0902: We figured that out. It's just making the process longer for us. It feels like it's doing everything backwards.
> (Group 9, Bug 5 Transcript)

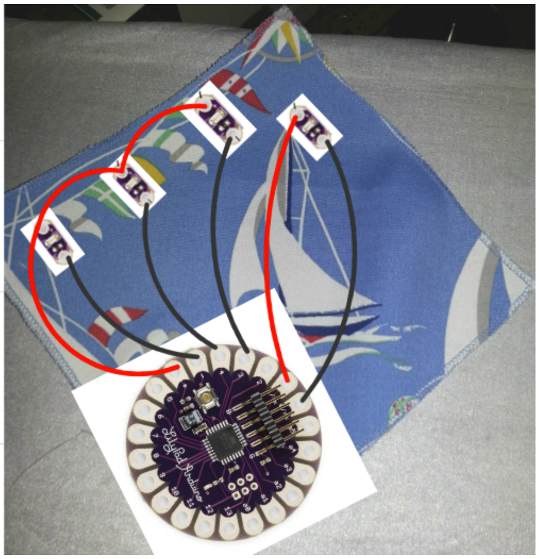


Figure 42. Bug 5 schematic.

Group 5 members had a more difficult time adapting to the changed set-up, as shown in the following interchange.

> M0502: Here's something wrong -
> F0501: Can we even do that?
> M0502: That's the wrong part.
> F0501: Can we do that?
> M0502: No.
> F0501: No.
> M0502: Why is negative . . .? The wiring is all wrong.
> (Group 5, Bug 5 Transcript)

Many ways to solve. Another computational thinking perspective that emerged during coding was the understanding that there are many ways to solve a problem. An example is given in the following comment, as the students discuss a not equal (!=) statement. The code to which the students are referring are:

```
//Every third time, the mast LED will not come on
 if (fmod(cycleCnt,3) != 0)
 {
  digitalWrite(led17, LOW);
 }
```

M0901: Yeah. So, yeah, we could have written this as another if . . . the fmod = 0, with the two equals signs . . . and that . . . that could've worked, right?

Bug 5 was a coding challenge with a missing setup statement, which was required to set the positive side of an LED to high. In addition to the missing statement, the code contained unintentional errors. One was that the coding for two LED's was switched, so that the behavior requested for one LED was coded for a different LED. While the code was switched, the program comments were not, so there were misleading program comments that did not match the actual code.

Multiple errors increased the difficulty level of Bug 5, but students were able to make slow progress with their debugging efforts. The unintentional complexity did give rise to the highest number of coded instances of computational thinking and electronics principles.

Bug 7. The students participated in the final debugging challenge, Bug 7, at their second Face-to-Face (F2F) meeting of the semester. This was five weeks after the first F2F meeting. Students worked with the same partners that they worked with for the previous 6 debugging challenges, and made audio recordings of their debugging interactions using the same Livescribe pens used in the first meeting. In the intervening five weeks the students had completed their individual Arduino projects and had begun to focus on programming Android devices using App Inventor.

The researcher expected that the previous experience with debugging exercises would make the activity go more smoothly on Bug 7, logistically, and since the same bugs were in Bug 7 as the Bugs 1 to 6, that students would be able to locate them without too much difficulty. In addition, by the time of the second F2F meeting, students had completed their individual Arduino projects, so most of them would have more experience with coding. Most of the

intervening five weeks the students had been working on their App Inventor projects, however, so they had been away from the Arduino platform for several weeks.

At the beginning of the first set of debugging activities, the researcher spent time explaining the use of the Livescribe recording pens and verifying that the groups were using them correctly. This did not occur at the second F2F meeting. Group 1 did not create a recording of their debugging process on Bug 7.

In the previous six debugging exercises, some students did not interpret the directions for the end behavior in the intended way. To avoid this problem, a short video demonstrating the end behavior for the Bug 7 e-textile was included with the copy of the starting code given to the students. However, one of the unintended consequences was that at least some students used the video to find sewing differences. Group 9 discusses this with the researcher.

> M0901: So, Vicky . . . Vicky…I think we're done…is this the correct behavior? We didn't do it the way you did it in the video…
> Vicky: Uh, have you . . .
> M0901: We didn't sew it the way you did in the video…
> Vicky: Oh . . . Does that . . . does that show in the video? <laughs>
> M0901: Yeah, I mean, because . . .
> Vicky: No, no. There is no reason why that would be THE solution.
> M0902: We improvised, we adapted, we overcame.
> (Group 9, Bug 7 Transcript)

Bug 7 errors. Bug 7 contained three engineering challenges and three coding challenges. The six challenges in Bug 7 were the same challenges that the students experienced in Bugs 1 to 6, all combined into one e-textile project. The students were told that there were multiple bugs in the project, but they were not told that the project contained the same prior six bugs; interestingly, none of the transcripts showed the students noticing that they were the same bugs that they had seen before. The first image in Figure 43 shows a layout of the circuits in Bug7, as

given to the students, and the second image shows a correct layout. As before, red indicates positive current and black indicates negative.

One of the circuit layout errors shown in the first image in Figure 43 is the black connection between LED D and LED B. Linking the Android's arm and leg together prevents them from having the independent behavior requested in the design instructions (the arms and legs were supposed to alternate their flashing). A second circuit layout error is shown in the black line from LED B to LED C. This is an error because it links a positive and negative terminal. The third circuit layout error occurs at LED A. A short circuit exists, because the positive and negative terminals of LED A are connected by stitching on the backside of the fabric. This short circuit error was anticipated to be one of the most difficult physical bugs for the students to locate, because it was on the back of the fabric, and there wasn't an overt crossing of threads.

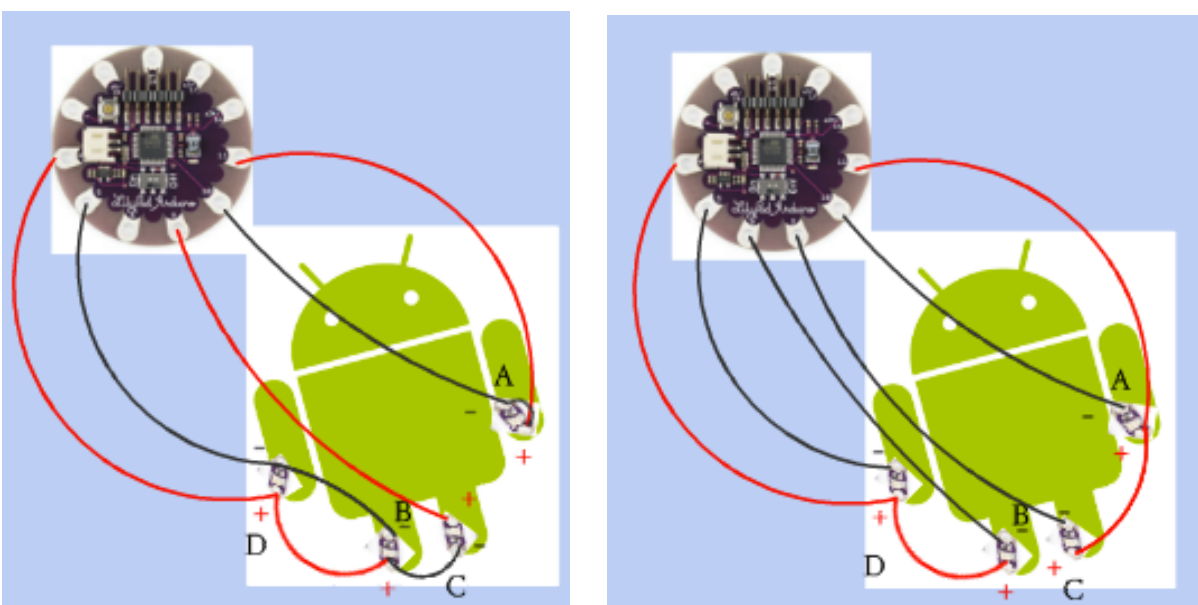


Figure 43. Bug 7 schematics, incorrect and correct.

Bug 7 CT coding. The chart in, Bug 7 Computational Thinking Coding Summary, displays a summary of the computational thinking and electronic principle codes that were found in the transcripts for Bug 7.

Debug strategy. As with Bug 5, the most frequently coded observation in Bug 7 was Debug Strategy. Debug Strategy is one of the categories that emerged as a further breakdown of the Testing and Debugging practice identified by Brennen and Ressnick (2012). Examples of the Debug Strategy code were given earlier, in the description of Bug 5 CT coding.

Predict expected behavior. The second most frequently coded item for Bug 7 was Predicting Expected Behavior, also one of the most frequently coded items for Bug 5. In the conversation below, one of the Group 9 members predicts where the bug might be.

> M0902: Is the code done already?
> M0901: Code's done…yeah…as far as I can tell…I mean…once we get it all hooked up…and running, then we'll see…she said there might be difficult bug for us to discover.
> M0901: I'm guessing it might have something to do with being sewn on to the felt…but I don't know…I don't see anything else, really…
> (Group 9, Bug 7 Transcript)

Another example of Predicting Expected Behavior from the Bug 7 transcripts follows.

> M0301: And what should be happening, is that it should be cycling low and high…
> M0302: OH…OK…
> M0301: To get them to blink.
> (Group 3, Bug 3 Transcript)

Power, positive and negative. The third most frequent code was an electronics principle: Power, Positive Negative. The same method of chaining the positive terminals and controlling the LED's by sending signals to the negative terminals used in Bugs 4, 5 and 6 was used in Bug 7. It continued to be a source of difficulty for the students. The switched circuit style led to the conversation below, between members of Group 7.

M0701: Well . . .this one should . . . there shouldn't be any connection between those two at all . . . well actually . . . no . . . THIS is ok.
F0702: This is OK?
M0701: Because this is just the power . . . this is the positive . . . it's the same positive power . . . you know . . . this is just . . . ground . . . but she has it reversed . . . like . . . positive 5 volts . . . positive 5 volts . . . and this should also be 5 volts . . .
(Group 7, Bug 7 Transcript)

The conversation between Group 2 members shows how their examination of the positive and negative led them to the discovery of one of the bugs.

F0202: Except that . . . shouldn't that be . . . plus?
F0201: Is that not plus?
F0202: This is min . . . this is negative.
F0202: 10 . . . shouldn't that go to . . .
F0201: It should go to plus.
F0202: Yeah. That's what I think.
F0201: So 10 . . . goes to plus.
F0202: So that needs to be flipped?
F0201: Yeah.
(Group 2, Bug 7 Transcript)

Table 11

Bug 7 Computational Thinking Coding Summary

Code	Group	Subgroup	Partnerships where found	Count
* Debug Strategy	CT Practice	Test & Debug (T & D)	Groups 2, 3, 4, 5, 8, 9	35
* Predict Expected Behavior	CT Practice	T & D	Groups 2, 3, 4, 5, 7, 8, 9	33
* Power, Positive Negative	Electronics Pr.		Groups 2, 3, 4, 5, 7, 8	30
* Establish Circuit Correctness	CT Practice	T & D	Groups 2, 3, 4, 5, 7, 8, 9	29
Testing and Debugging, General	CT Practice	T & D	Groups 4, 5, 7, 8	26
* Independent Behavior, Different Pins	Electronics Pr.		Groups 2, 3, 5, 7, 8, 9	23
* Benchmark	CT Practice	T & D	Groups 2, 3, 4, 5,	21
* Documenting	CT Practice		Groups 2, 3, 4, 5, 8, 9	19
Connecting	CT Perspective		Groups 7, 8, 9	18
Events	CT Concept		Group 3, 5, 8, 9	16
* Able to Adapt to Others Code	CT Perspective		Groups 2, 5, 8	15
Incremental and Iterative	CT Practice		Groups 2, 3, 4, 5, 7, 8	15
* Control LEDs, High Low	Electronics Pr.		Groups 2, 3, 5, 7, 8	15
* Crossed Threads, Short Circuits	Electronics Pr.		Groups 4, 7, 8, 9	14
* Refer to Specifications	CT Practice	T & D	Groups 2, 3, 4, 5, 7, 8	14
* Physical Setup Determines code	Electronics Pr.		Groups 2, 3, 5, 8	13
Loops	CT Concept		Group 2, 3, 5, 8	12
* If Connected, Same Polarity	Electronics Pr.		Groups 2, 3, 4, 7, 8	12

(continued)

Code	Group	Subgroup	Partnerships where found	Count
* Walk Through Code	CT Practice	T & D	Groups 2, 4, 5, 8, 9	9
* Many Ways to Solve	CT Perspective		Groups 2, 5, 8, 9	8
* Use Visibility of Behavior	CT Practice	T & D	Groups 2, 4, 5, 8, 9	8
Reusing and Remixing	CT Practice		Groups 2, 3, 4, 7	7
* Plan Debugging Strategy *	CT Practice	T & D	Groups 3, 4, 7	7
* Start Over Again	CT Practice	T & D	Groups 2, 4, 5, 8	6
* Verify Correct End	CT Practice	T & D		6
* Need Delay to See LED Behavior	Electronics Pr.			6
Sequences	CT Concept			5
Expressing	CT Perspective			4
* Keep Code Clean	CT Practice		Groups 3, 9	3
Parallelism	CT Concept		Group 9	1
Conditionals	CT Concept		Group 8	1
Data	CT Concept		Group 3	1
TOTAL				432

Note. * Indicates coding category not in the original literature-derived list computational thinking elements.

Independent behavior, different pins. One of the electronic principle codes that appeared frequently in Bug 7, but not in Bug 1, nor Bug 5, was Independent Behavior, Different Pins. This is because debugging the problems of Bug 7 required recognition that two pins needed to be separated in order for them to have independent behavior. The Android robot's arms and legs were supposed to flash alternately, but in the original circuit layout a pair of the arms and legs was linked together. Group 2's interaction below is an example of Independent Behavior, Different Pins.

> F0202: But my point is…pins…control the lights, right?
> F0201: uh-huh.
> F0202: And then our job is to make these two light up in alternation.
> F0201: Right.
> F0202: Right?
> F0201: uh-huh.
> F0202: So, then I'm assuming the…legs…should be on a different pin. Why are they on the same pin?
> (Group 2, Bug 7 Transcript)

Another example of Independent Behavior, Different Pins is found in the conversation between Group 7 members.

> M0701: Um . . . And then 6 for the robot's right leg . . . there isn't anything even on 6.

F0702: <chuckle> Oh. See here is where the error is…
M0701: Yeah . . . these two should not be connected.
F0702: Should not be connected.
(Group 7, Bug 7 Transcript)

Computational thinking perspectives that were coded in Bug 7 were, in descending order of coded instances: Connecting, Able to Adapt to Others' Code, Many Ways to Solve, Expressing.

Connect. An example of Connecting comes from a conversation between the members of Group 8.

F0802: Actually, I have a whole new appreciation for coding.
F0801: Yeah.
F0802: I wish I would have learned this a long time ago . . . You know what I mean?
F0802: And it is good to know.
(Group 8, Bug 7 Transcript)

Adapt to others' code. The following interchange between Group 2 members shows their preference for using their own way of laying out circuits, rather than adapting to others' code and set-up.

F0202: So, we need to cut THIS. And then resew this to 6. And we shouldn't have cut THAT. Well…whatever.
F0201: But the positive and the negative shouldn't be going to the positive, right?
F0202: I think we should resew . . . or at least like . . . let's use the wires though. You know what I mean? I don't want to sew.
F0201: Where are we now?
F0202: We cut everything and…and use wire <laugh>. Problem solved.
(Group 2, Bug 7 Transcript)

Many ways to solve. In the following example of the code Many Ways to Solve, the members of Group 5 seem excited as they realize they have some options about how to make a change.

F0201: It's flipped.
F0202: OK. Or not….
F0201: Or maybe we just switch it in the program.
F0201: Maybe we just switch it in the program.

F0202: COOL! Yeah that's true! We could do that!
(Group 2, Bug 7 Transcript)

Express. The computational thinking perspective of Expressing is one of the perspectives identified in Brennan and Resnick (2012). The following example of Expressing is found in the transcript of Group 8.

F0801: That was fun (referring to her home-made frame).
F0801: I was like, I am not going to buy a frame for this. I'm a maker, fool!
F0802: It's true, though. Those projects are like magnets to kids…
F0802: Like Sadie was like, <? ? Staring>
F0801: Yeah, I know. My niece and nephew were like . . .
F0802: When she saw the lights, she was like (gesture ???) . . . And I'm like . . . she can't be in here.
K <from other group>: Like moths to the flame.
(Group 8, Bug 7 Transcript)

Summary, Evidence of CT in Debugging Activities

Through the process of transcribing and coding Bugs 1, 5, and 7, examples were found of many of Brennan and Resnick's computational thinking concepts, practices, and perspectives. Table 12 gives counts totaled across the three bugs of the coded instances of Brennan and Resnick's (2012) identified concepts, practices, and perspectives.

Table 12

Computational Thinking Across Bugs 1, 5, 7; CT Elements from Brennan and Resnick (2012)

Element	CT Dimension	Total Coded
Sequences	Concept	16
Loops	Concept	34
Events	Concept	16
Parallelism	Concept	4
Conditionals	Concept	23
Operators	Concept	22
Data	Concept	6
Incremental and Iterative	Practice	54
Testing and Debugging	Practice	636
Reusing and Remixing	Practice	48
Abstracting and Modularizing	Practice	20
Expressing	Perspective	8
Connecting	Perspective	27
Questioning	Perspective	2

While some of the counts are low, examples were coded for each of the literature-identified computational thinking concepts, practices, and perspectives. As was expected, the instances were heavily weighted on testing and debugging, since that was the nature of the tasks given to the students.

Table 13 presents total instances across Bug 1, Bug 5, and Bug 7 for the computational practices and perspectives that emerged during the coding process. The testing and debugging practice was refined into sub-practices. Electronics principles were also added to the coded elements since the e-textile projects also involved computational thinking exhibited through the use of electronic circuits.

These practices and perspectives that emerged as students worked together on debugging e-textile designs could lead to a greater understanding of computational thinking, especially as used by students in debugging and testing e-textile projects.

Table 13

Computational Thinking Counts Across Bugs 1, 5, 7; Emergent CT Elements

Element	CT Dimension *	Total Coded
Adapt to Others Code	Perspective	20
Many Ways to Solve	Perspective	10
Document	Practice	67
Keep Code Clean	Practice	3
Benchmark	Practice (T & D)	69
Debug Strategy	Practice (T & D)	143
Establish Circuit Correctness	Practice (T & D)	68
Predict Expected Behavior	Practice (T & D)	80
Refer to Specifications	Practice (T & D)	67
Use Visibility of the Behavior	Practice (T & D)	49
Verify Correct End	Practice (T & D)	27
Walk Through Code	Practice (T & D)	40
Control LED Lights, High/Low	Concept (E)	56
Crossed Threads Cause Short Circuits	Concept (E)	33
If Connected, Same Polarity	Concept (E)	24
Independent Behavior, Different Pins	Concept (E)	39
Need Delay to See LED Behavior	Concept (E)	11
Physical Setup Determines Code Requirements	Concept (E)	33
Power/Positive/Negative	Concept (E)	74

*CT Dimension
Practice (T & D) is a sub-practice of Testing and Debugging
Concept (E) is an electronics principle

CT Evidence in Group Debrief

At the conclusion of the first six debugging exercises, the course professor asked the students various questions about their experiences with the exercises. Students gave feedback about things they learned, strategies they used to debug, about how the exercises related to their own projects, suggestions for the next iteration, and about their feelings. As students discussed their experiences with the debugging exercises in the group debrief, they shared some of the ways they had worked out the solutions. Several of these examples of computational thinking are illustrated below.

Practice: Abstract and modularize. Student M0301 commented on something that he noticed, or in other words, a principle he had abstracted from the experience, relating to how the circuits could be grounded. He said,

> One example of something that I learned today that could have potentially been useful: I was fixated on the idea that stuff that is going to go to ground has to go back to the ground pin. It never occurred to me that I could set one of the digital pins as a ground. That would have changed things how I lined up things on my board.
> (Group Debrief, 02:03)

The technique of using programmable digital pins as ground within the bug exercises was not something that was discussed with the participants. It was something that this student noticed and felt was significant because of his own experiences with the layout of the circuits in his project. In other words, he abstracted the principle from what he saw.

Practice: Benchmarking. Student M0101 discussed a strategy that he and his partner used while working on the complicated Bug 5. This strategy was identified as an emergent

computational practice, benchmarking, during the coding phase of the debug transcripts discussed previously. He said,

> Since I had forgotten how to do a blink, since I don't need a blink for my tie, I just started typing in different numbers and values and seeing what would happen. How long would it blink for? How fast was the blinking frequency? Part of my trying to trouble shoot was just trial and error.
> (Group Debrief, 05:58)

Practice: *Versions. Student M0301 also discussed something that he and his partner learned from Bug 5, which is identified here as an emergent computational thinking practice.

> One thing that stuck out for <partner's name> and me on project 5, was . . . the absolute vital importance of preserving the original code. (Laughter and agreement from the other students.) We could get pretty far down a blind alley, and, uh . . . if we had not had the ability to go 'Nope. We are going back to zero' we would never have made it.
> (Group Debrief, 07:40)

The asterisk next to the practice versions is because it is an emergent practice.

Practice: Incremental and iterative. In the group debrief students discussed how they used commenting out sections of code to make incremental progress. Group 9 used alligator clips, a battery, and an LED as a crude voltage meter to isolate where in the thread the voltage dropped. They walked the ends of the alligator clips over different sections of the stitching, and thus were able to locate an area where a knot had come untied on the back of their project. This is a physical analogy to commenting out sections of code.

In the following comments, student F0202 discusses Group 2's incremental process.

> We never isolated anything. We'd keep uploading and see if it worked. I think we uploaded one code ten times. And then we'd fix a little piece, and then we'd say, "Let's see if it works."
> (Group Debrief Audio, 00:16:58)

Practice: Reusing and remixing. Another strategy mentioned by the students in the debriefing was using the Arduino Blink code as a reference.

Practice: Debug, comparing circuit and code. Group 1 used the relationship between the circuit set-up and code to gain enlightenment. Student M0101 said,

> <Partner name> and I, every time we had a new project, we would look at all of the wiring, and follow the thread. And if something didn't go like we thought it should, we would look at the code and try to figure out why was it going to that thing, because it should be going to positive, and something like that. So we looked at it, going back and forth, between the physical and with the code.
> (Group Debrief Audio, 00:16:29)

CT Evidence in Forums

Students, professors, and teacher's assistants all had the capability of creating threaded discussions in the forum section of the course shell for EDLT740. Threaded discussions were created for exploring ideas from the readings, posting about the projects, and for feedback. Some of the students' posts reflected computational thinking.

Practice: Testing and debugging. Student F0102 discussed the value of debugging with another person.

> Debugging out loud with another individual is so much different than working on your own. You are able to feed off of each other's ideas and ask questions to someone else other than yourself. Seems debugging is a fancy word for problem solving.

Perspective: Connecting. Student F0802 mentions in passing an example that illustrates the computational thinking perspective of connecting. She wrote, "This was the code in which I learned the 'grammar' of coding." The student is a journalism teacher, and she connected the structure and syntax of a computer program to the grammar of a written work.

Practice: Debugging and testing. Student M0901 talks about how he and his partner debugged a more difficult challenge.

> We had a different problem with one of the sewing ones--the last one, I think--which was frustrating because the other two had been fairly straightforward to figure out. The trouble with the last one seemed to be our sewing: though everything was connected correctly, the lights would not turn on as described. I tried using the battery and a pair of alligator clips to provide power to individual LEDs to see if one was broken, and they all

> worked fine, but the circuit still did not work. Doing this, I did discover that one of the knots I had tied had come loose, so that was good to know, but I never did really discover what the trouble was, even though we knew what was wrong and how to fix it.

CT Evidence in Reflection Papers

Students wrote reflection papers at the end of the 16-week semester. They discussed their experiences and their thoughts as they learned about making, computational thinking, social learning, and created Arduino and Android App Inventor projects.

Perspective: Connecting. Student M0901 included the following in his reflection:

> I can recognize now, looking back, that those same feelings came back in full force while working on Arduino projects: my mind goes all dreamy and everywhere I look I see Arduino applications and gadgets waiting to happen. I think this is the power of easy entry consumer DIY systems like Arduino, pulling the curtain back from the magical electronic systems around us and making it possible for us to create some of the magic ourselves. It empowers learners with tools that they typically don't have access to, along with a community of co-learners and experts that can guide them along and assist them as they make their way through difficult projects.

Perspective: Expressing. In the following observation, student M0302 shows the computational thinking perspective of expressing.

> Each set of tools had its own prescribed processes and associated coding to learn. However, once the code was cracked, the creativity grew geometrically, for each set of tools branched into a multitude of possibilities.

Another example of the computational thinking perspective of expressing is found in the reflection of student M0101.

> Besides the encouragement that I received during each conversation, there was an fascinating dynamic that occurred as new ideas that welled up inside my brain regarding the process that was needed to build a clothing accessory that no one on the planet had ever counted as an object in their wardrobe.

Perspective: *Failure is just a step forward. From student M0701's reflection paper emerges another computational perspective: Failure is just a step forward. A part of his reflection follows.

> Making is an iterative process, with multiple cycles of build, test, debug, fix, and retest. Going through the full process of making something helps students to learn not to give up at the first sign of failure, but to understand that failure itself is part of the building and learning process.

The asterisk next to ‘Failure is just a step forward’ is to identify it as one of the emergent computational thinking perspectives.

Summary, Evidence of Computational Thinking

The debugging activities in the learning module elicited many instances of the computational thinking concepts, practices, and perspectives contained in Brennan and Resnick (2012). In addition, new computational practices and perspectives emerged during the coding process. Examples of the most frequently coded computational thinking elements were pulled from the transcripts of the debugging exercises and discussed in this section.

During the group debrief students explained their thinking and debugging processes and gave feedback about the experience. Examples of computational thinking from the group debrief were presented in this section. A new computational thinking practice of versions emerged from the group debrief.

Forum posts from the EDLT740 course were examined for instances of computational thinking. Student posts from the forums that illustrated computational thinking were presented.

Lastly, student reflection papers were examined, and additional examples of computational thinking were extracted and shared. The Computational thinking perspective of Failure is Just a Step Forward emerged from the reflection papers.

Confidence in Computing Abilities

In addition to seeking evidence of computational thinking, this design research study outcome looked for increased student confidence in computing abilities. The following sources were examined for indications of confidence or lack of confidence:

- Pair interactions during the debugging exercises
- Self-Efficacy Survey

Indications of Confidence in Debug Exercises

Three items related to confidence were coded in the examination of the transcripts of Bugs 1, 5, and 7. The codes were Confidence, Lack of Confidence, and Comparing to Other Groups. The data is presented in the following table, by group, and by code. The instances were not coded to individuals but were coded for the group. A confidence ratio was calculated by dividing the number of expressions of confidence by the number of expressions of lack of confidence. A ratio greater than one indicates more expressions of confidence than lack of confidence. A ratio less than one indicates more expressions of lack of confidence than confidence.

Table 14

Confidence Counts from Bug 1, 5, 7 Transcripts, By Group

Group	Compare to Other Group	Confidence	Lack of Confidence	Total	Confidence Ratio
Group 1 M0101, F0102	4	16	0	20	>16 *
Group 2 F0201, F0202	6	10	0	16	>10 *
Group 3 M0301, M0302	0	10	6	16	1.66
Group 4 F0401, F0402	44	14	16	74	0.875
Group 5 F0501, M0502	16	22	6	44	3.67
Group 6 F0601, F0602	6	2	6	14	0.33
Group 7 M0701, F0702	20	6	14	40	0.43
Group 8 F0801, F0802	34	14	70	118	0.2
Group 9 M0901, M0902	12	8	2	22	4
Totals	142	102	120	364	0.85

* Since division by zero is undefined, the ratio cannot be calculated.

Table 14 shows 142 coded instances of comparing to other groups, 102 expressions of confidence, and 120 expressions of lack of confidence. The lowest confidence ratio is .2 (Group 8) and the highest is >16 (Group 1, no expressions of lack of confidence recorded).

An example of comparing to other groups occurs in the following conversation between the members of Group 5, during their work on Bug 5.

M0502: Survey the room . . . we are with everyone else.
F0501: Huh?
M0502: <name> is fast.
F0501: Yeah. <name>'s fast.
M0502: No, they're not. They are still behind us.
F0501: No, they're . . . they're in the previous one.
F0501: Let's not compare. They've finished number 4, so they are probably the same.
F0501: OK. Let's hurry.
(Group 5, Bug 5 Transcript)

Group 8 is also paying attention to the progress the other groups are making, as shown in the following excerpt.

R (from another group): So, Vicky we are done -
F0802: Quit saying it, over and over again. Just to make us feel bad . . .
(Group 8, Bug 5 Transcript)

The following quote taken from Group 4 as they were working on Bug 1 shows the group members comparing themselves to another group and judging themselves.

F0401: How are they doing that?
F0402: I feel stupid.
F0401: How are they doing that?
(Group 4, Bug 1 Transcript)

Students made expressions of confidence about e-textiles, electronics, programming, and other general statements of confidence. The following statements were from Group 9 as they worked on Bug 7.

M0902: We improvised, we adapted, we overcame.
Vicky: You CAN improvise.

M0901: You had these LED's going individually to separate pins, and we realized they were going on at the same time, and there was no reason not to put them in parallel.
M0901: So, we just . . . so we just did that.
(Group 9, Bug 7 Transcript)

While it can be seen from the Confidence Count Table 14 that Group 8 had many more expressions of lack of confidence than any of the other groups, they did have some expressions of confidence as well. The following quote shows one of these.

F0801: Isn't it good we've progressed to the point where we're not scared of electrocuting ourselves while we are working on this?
F0802: Let's be honest…I'm still a little scared of it…
(Group 8, Bug 7 Transcript)

Group 5 members developed confidence in their ability to sense where the errors were located.

M0502: Check the code one more time on 5…. low?
F0501: Uh-huh.
F0501: I just don't think it's the code.
M0502: It's NOT the code.
F0501: I think it's the…
M0502: The wiring.
F0501: The LED light or the wiring.
(Group 5, Bug 7 Transcript)

Group 2, who ended up being the group that finished the six debug exercises first, express pride in their work.

F0202: I am so proud of us.
F0201: Cool.
F0202: <name> we are awesome …
F0201: We are on our last one. We get to go to lunch early.
(Group 2, Bug 5 Transcript)

The members of Group 8 express some of their feelings of inadequacy as they speak to a classmate who has helped them with their projects.

F0801: We don't function without you, <name>.
F0802: I know you didn't sign up for this.

F0802: It's amazing that we have jobs where you're not there. <laughter>
(Group 8, Bug 7 Transcript)

Group 4 perhaps had their biggest struggle with Bug 1. As they saw other groups working through it, they had the following conversation.

F0402: See they're not even using the board…they're just using the battery…
F0401: One . . . is this positive or negative?
F0402: But these are sewing errors. I feel so dumb . . .
F0401: How do we get them…let's just take all these off…we need to see if ANYTHING lights up…that's the problem.
F0402: Vicky! We don't know what we're doing. We can't make it light up.
(Group 4, Bug 1 Transcript)

One of the members of Group 6 also expressed lack of confidence in her understanding as she and her partner worked on Bug 5.

F0602: OK. So, I've got to write down what the heck we did.
F0602: So, let's see here. Um . . . So . . .
F0602: Bug five was a programming issue with the top flag and mast flag.
F0601: Honestly, I don't think I really understood what we did.
F0602: Well, maybe as we talk through it.
(Group 6, Bug 5 Transcript)

Table 15 shows the counts for the codes comparing to other groups, confidence, and lack of confidence, sub-totaled by bug. Contrary to the researcher's expectation, the hardest bug was not the one that had the lowest confidence ratio; Bug 5 actually had the highest confidence ratio.

Table15

Confidence Counts from Bug 1, 5, 7 Transcripts, By Bug

Bug	Compare to Other Group	Confidence	Lack of Confidence	Total	Confidence Ratio
Bug 1	42	10	14	66	0.71
Bug 5	48	40	24	98	1.67
Bug 7	52	44	76	172	0.58
Totals	142	102	120	364	0.85

Summary, Indications of Confidence in Debug Exercises

Examination of the transcripts revealed that the debug exercises brought out both expressions of confidence and lack of confidence, with a notable number of comparisons to other groups.

A confidence ratio of the counts of confidence to the counts of lack of confidence was created for each bug, with a ratio greater than 1 indicating there were more expressions of confidence than lack of confidence. A ratio less than 1 indicates that there were more expressions of lack of confidence than confidence. The confidence ratios for Bug 1 (.71) and Bug 7 (.58) are both less than one, indicating that the students expressed more lack of confidence than confidence as they worked on the bugs. The confidence ratio of Bug 5, the most difficult bug, was 1.66, indicating that there were more expressions of confidence than lack of confidence.

The results regarding confidence in the transcripts show that the debug exercises did bring out expressions of confidence and lack of confidence. The overall confidence ratio of .85 shows there were more expressions of lack of confidence than confidence. Notably, the members of Group 8 made over half of the total expressions of lack of confidence. With the number of expressions of confidence and lack of confidence expressed during the debugging exercises, it seems that there is an opportunity to address confidence through debugging exercises.

The information obtained from the examination of expressions of confidence in the debugging exercises will be used in the planning of the next iteration of the debug exercises.

Indications of Confidence in E-Textile Self-Efficacy Questionnaire

Another measure of confidence used in this study was the E-Textile Self-Efficacy Questionnaire. Appendix C contains a copy of the survey to which the EDLT740 students

responded. The survey was administered twice, once at the F2F meeting, before the debugging exercises were presented, and once at the end of the learning module, approximately 4 weeks later. Fifteen of 18 students completed the E-Textile Self-Efficacy Survey at the end of the module, but only 8 of the 18 students completed both the pre and post surveys. Student names were included with the responses so that pre and post responses could be matched. A sample question is shown here to illustrate the style of the questionnaire.

1	I could complete an e-textile project using the LilyPad Arduino environment if there were **no one around** to tell me what to do	Strongly Disagree	1	2	3	4	5	6	Strongly Agree

Figure 44. Sample question from the e-textile self-efficacy instrument.

Changes in self-efficacy, overall. The pre and post scores on the e-textile self-efficacy questionnaire were compared to see if changes occurred, and the overall e-textile self-efficacy of the students at the end of learning module was examined. The change scores were calculated by subtracting the pre test score from the post test score for those students who completed both pre and post tests. Average change scores were created for each of the support types in the survey.

Table 16

Average Change Scores in E-Textile Self-Efficacy

Support Type	Males (3)	Females (5)	Combined
No One Around	2.3	-0.2	0.8
Manuals for Reference	1.7	-0.6	0.3
Seen Someone Else	0.7	-0.2	0.1
Call on Someone Else	-0.7	0.0	-0.4
Help Me Get Started	-0.3	0.0	-0.1
Lot of Time	0.3	0.0	0.1
Built-In Help Facility	0.7	0.8	0.8
Someone Showed Me How	-0.3	0.0	-0.1
Used Similar Programming Environments	0.0	0.6	0.4
Combined Tools Above	-0.7	0.3	-0.1

Table 16 shows small change scores in most areas. It is interesting to note that the male students' average change score for being able to complete an e-textile project with no one around rose 2.3, while the females average change score for being able to complete an e-textile project with no one around dropped by .2. The relatively large change score average for the males in that category was due to one student's score, which rose 5 points. Since there were only three male students who completed both pre and post e-textile self-efficacy surveys, this one student's response had a big effect on the average.

Self-efficacy changes, by individual. While averages scores can give insight, looking at the individual changes for the students who completed both the pre and post self-efficacy questionnaire can also be enlightening.

No one around. Figure 45 shows the pre score on the self-efficacy questionnaire for completing an e-textile project with no one around, by student, with the rise or fall of their post score. Green indicates a rise in the self-efficacy score for the student and red indicates a fall in the self-efficacy score. Four students experienced a rise in their self-efficacy score for completing an e-textile project with no one around, two experienced a drop, and two students' pre and post scores stayed the same. For those students who experienced a rise, the bottom of the green rectangle indicates their pre score, and the top of the green rectangle represents their post score. For those students who experienced a fall, the top of the red rectangle represents their pre score and the bottom represents their post score.

The first five students shown in the chart are females (F____) and the last three are males. (M____). Student M0101 experienced the greatest rise, from one to six, while Student F0801 experienced the greatest fall, from five to one. She was a member of Group 8, which was

the group previously mentioned that had more than half the total expressions of lack of confidence.

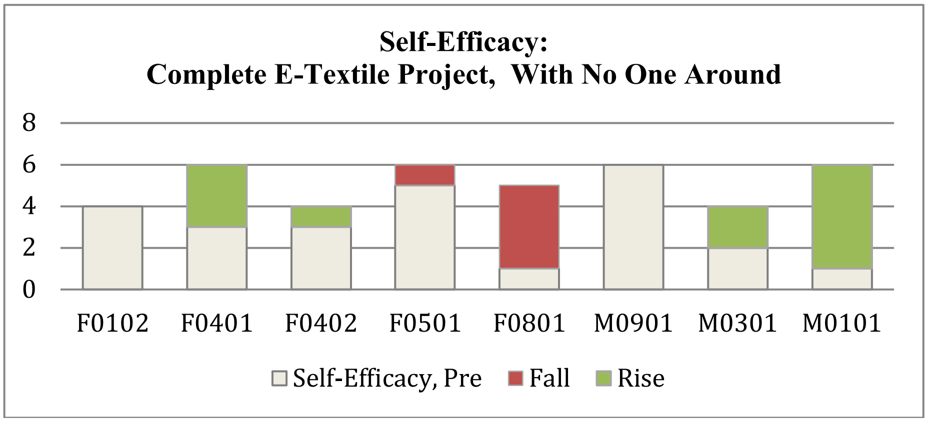


Figure 45. Self-efficacy to complete e-textile project with no one around.

With manuals only. Figure 46 shows the pre score on the self-efficacy questionnaire for completing an e-textile project with manuals only, by student, with the rise or fall of their post score. As in the prior diagram, green indicates a rise in the self-efficacy score for the student and red indicates a fall in the self-efficacy score. Four students experienced a rise in their self-efficacy score, one experienced a drop, and three students' pre and post scores stayed the same. The method of representing the rise and fall is the same as for the previous diagram and will be the same for the remaining diagrams. Three of the four students who experienced an increase had scores that rose only one point. The fourth student, M0101, had a score that rose five points.

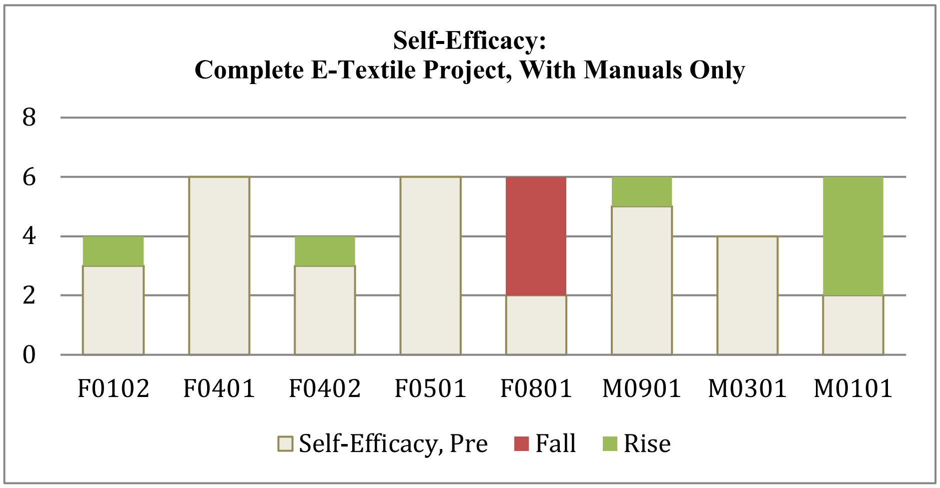


Figure 46. Self-efficacy to complete e-textile project with manuals only.

Seen someone. Figure 47 shows the pre score on the self-efficacy questionnaire for completing an e-textile project if someone was seen doing one, by student, with the rise or fall of their post score. Three students experienced a rise in their self-efficacy score, two experienced a drop, and two students' pre and post scores stayed the same. The five scores that changed differed by only one point each.

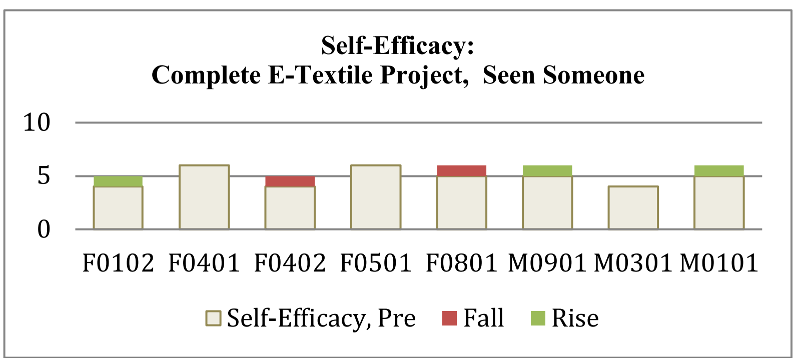


Figure 47. Self-efficacy to complete e-textile project if seen someone.

Able to call on someone. Figure 48 shows the pre score on the self-efficacy questionnaire for completing an e-textile project if able to call on someone, by student, with the rise or fall of their post score. One student experienced a rise in his self-efficacy score, four

experienced a drop, and three students' pre and post scores stayed the same. The one student whose score rose had self-efficacy scores that rose or stayed the same for all help-types.

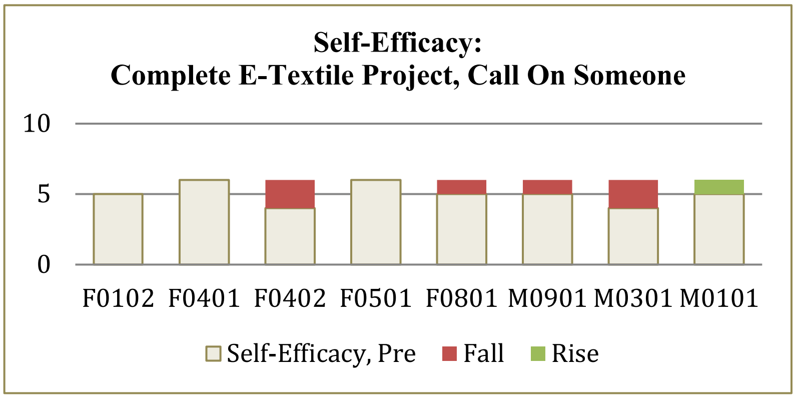


Figure 48. Self-efficacy to complete e-textile project if call on someone.

Help to start. Figure 49 shows the pre score on the self-efficacy questionnaire for completing an e-textile project if given help to start, by student, with the rise or fall of their post score. One student experienced a rise in his self-efficacy score, two experienced a drop, and five students' pre and post scores stayed the same. The one student whose score rose had self-efficacy scores that rose or stayed the same for all help-types.

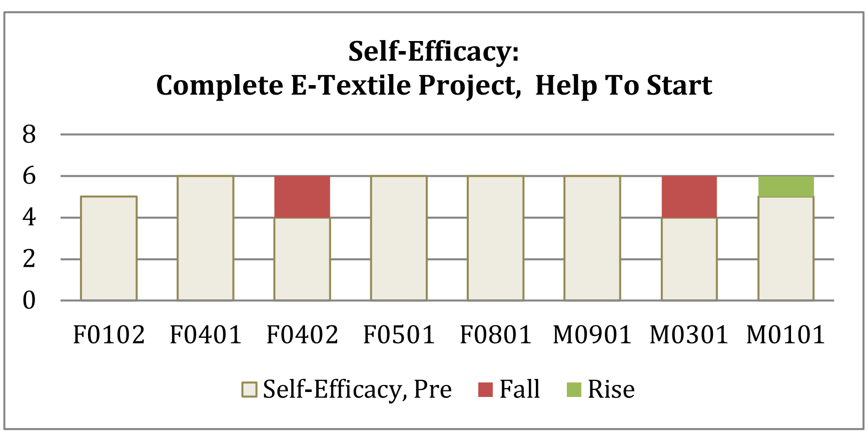


Figure 49. Self-efficacy to complete e-textile project if helped to start.

A lot of time. Next, Figure 50 shows the pre score on the self-efficacy questionnaire for completing an e-textile project if given a lot of time, by student, with the rise or fall of their post

score. One student experienced a rise in his self-efficacy score, one experienced a drop, and six students' pre and post scores stayed the same. Again, the one student whose score rose had self-efficacy scores that rose or stayed the same for all help-types.

Built-in help tools. Figure 51 shows the pre score on the self-efficacy questionnaire for completing an e-textile project with the built-in help tools, by student, with the rise or fall of their post score. Four students experienced a rise in their self-efficacy scores, two experienced a drop, and two students' pre and post scores stayed the same. Three of the post scores were six, the maximum, two were five, and two were four. Student F0501 had an increase in her post self-efficacy score of five points, and student F0801's score fell three points.

Someone showed how. Figure 52 shows the pre score on the self-efficacy questionnaire for completing an e-textile project if someone showed how to do it, by student, with the rise or fall of their post score. One student experienced a rise in her self-efficacy score, two experienced a drop, and five students' pre and post scores stayed the same. Six of the post scores were six, the maximum, and the two scores that fell, fell to four.

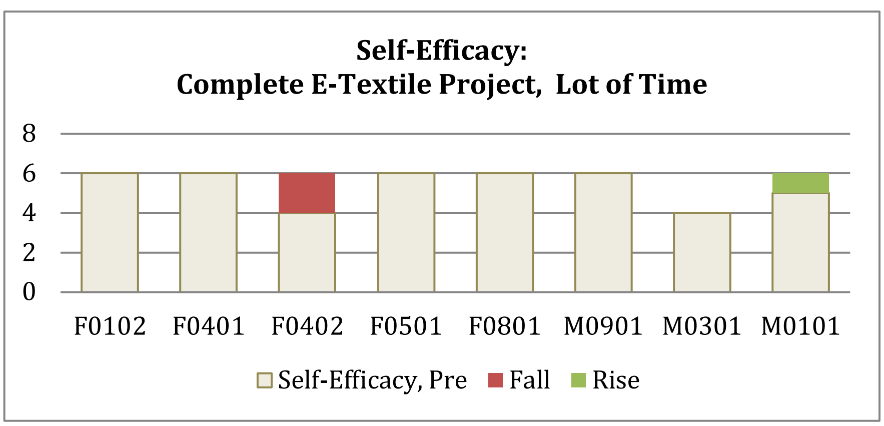


Figure 50. Self-efficacy to complete e-textile project if lot of time.

Previously used similar technology. Figure 53 shows the pre score on the self-efficacy questionnaire for completing an e-textile project if they had previously used similar technology,

by student, with the rise or fall of their post score. Two students experienced a rise in their self-efficacy score, one experienced a drop, and five students' pre and post scores stayed the same. Five of the post scores were six, the maximum score, one score fell to four, and one score remained at four. Student F0501's score rose from three to six.

Combination of tools. The final self-efficacy score figure, Figure 54, shows the pre score on the self-efficacy questionnaire for completing an e-textile project given a combination of tools, by student, with the rise or fall of their post score. Two students experienced a drop, and five students' pre and post scores stayed the same. The missing score is because the student left that question blank on the pretest. Her posttest score was five. Five of the post scores were six, the maximum score, and two scores fell to four.

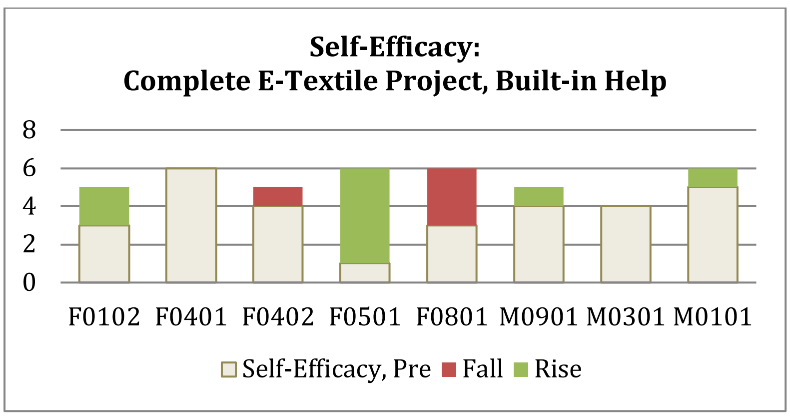


Figure 51. Self-efficacy to complete an e-textile project using built-in help.

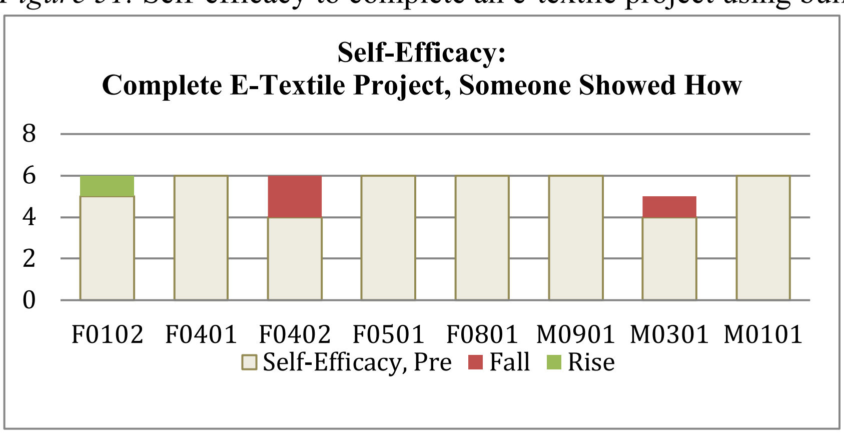


Figure 52. Self-efficacy to complete e-textile project if someone showed how.

Total change scores. **Error! Reference source not found.** shows total change scores, by student. One student had a total positive change greater than ten, and two students had total negative changes greater than ten. While the total change score of +3 gives us some idea of the overall effect on the group, it doesn't represent the changes that were experienced by these eight students. If we consider the amount of change, without considering the direction (the absolute value of the change), there were 61 points of change experienced by the eight students for whom that completed both the pre and post e-textile self-efficacy test.

Table 17

Total Change Scores in E-Textile Self-Efficacy, By Student

Student	Total Change
F0102	+ 6
F0401	+ 3
F0402	-11
F0501	+ 7
F0801	-13
M0901	+ 2
M0301	- 5
M0101	+14
Total Change	+ 3

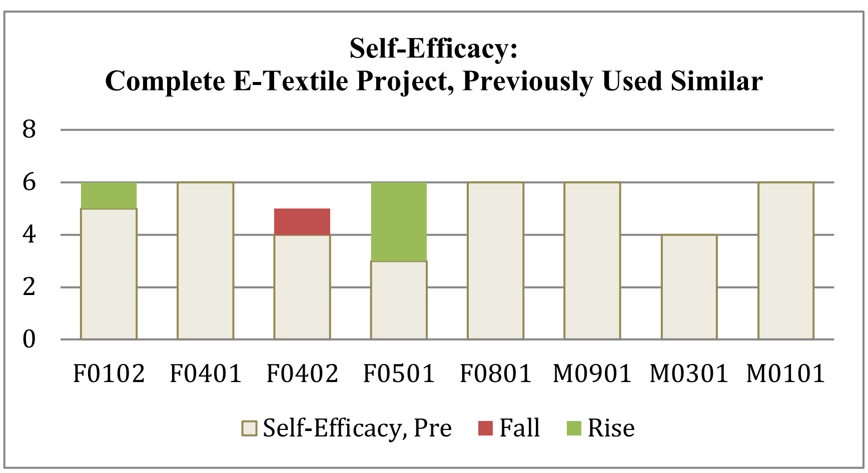


Figure 53. Self-efficacy to complete e-textile project if previously used similar.

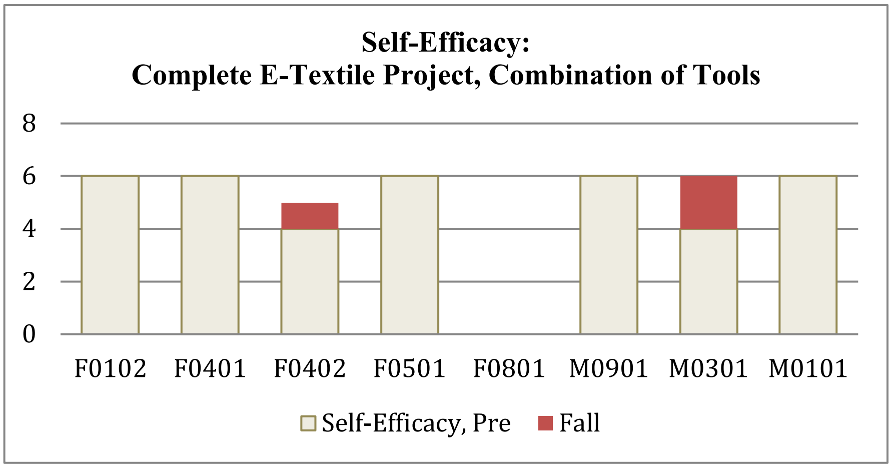


Figure 54. Self-efficacy to complete e-textile project if combination of tools.

Overall e-textile self-efficacy. Table 18 shows an overall e-textile self-efficacy score obtained by averaging all the scores for all the support types. It was calculated using all 15 of the post e-textile self-efficacy surveys.

Table 18

E-Textile Self-Efficacy, Post

Measure	Males (6)	Females (9)	Combined
Self-Efficacy Score	5.1	5.2	5.1

It can be seen in Table 18 that e-textile self-efficacy was high (greater than 5 out of a possible 6) at the completion of the unit for both male and female students. The Learning Module Evaluation had several questions related to e-textiles. The students were asked if they had the needed materials, if the background was distracting, if they liked experimenting with e-textiles, and if they would have been more interested in using traditional Arduino. The average responses scores for these questions are shown in Table 19. The students have an overall agreement score of 5.0 of liking experimenting with e-textiles. The overall agreement score about being more interested in traditional Arduino was 3.4.

Table 19

Learning Module Evaluation, E-Textiles (Disagree to Agree, 1 to 6)

Gender	Had Needed Materials	Debug Sets Background Distracted	Liked Experimenting with E-Textiles	More Interested in Arduino Using Soldering.
F (11)				
Avg	4.2	2.8	5.2	3.4
Min	2	1	3	1
Max	6	6	6	6
M (7)				
Avg	5.1	2.1	4.7	3.6
Min	4	1	3	2
Max	6	6	6	6
Overall	4.6	2.6	5.0	3.4

Summary, Indications of Confidence in E-Textile Self-Efficacy Questionnaire

Students' responses to the E-Textile Self-Efficacy Questionnaire were examined in several ways. Eight students took both the pre and post questionnaires. The changes in their scores were examined individually, for each kind of support type, and also averaged as a group. The majority of the changes were small. The overall average change was a -.1, with a possible range of -5 and 5. The pre-post change scores showed that individual students did experience a gain or loss in their e-textile self-efficacy.

Appeal to Non-Computer Science Majors

The third key outcome looked for in this design-based study was to create activities that appealed to non-computer science majors. The following sources were examined for evidence of whether or not the activities appealed to the students:

- Computational Attitudes Survey
- Reflection Papers
- Group Debrief
- Learning Module Evaluation

- Forum Posts

Evidence of Appeal in Computational Attitudes Survey

A copy of the Computational Attitudes Survey to which the EDLT740 students responded is found in Appendix D. The survey was administered twice, once at the beginning of the learning module and once at the end of the learning module, approximately 8 weeks later. Fourteen of 18 students completed the computational attitudes survey at the end of the module, with 12 of the 18 students completing both the pre and post surveys. Student names were included with the responses so that pre and post responses could be matched.

Changes in attitude. Table 20, Table 21, Table 22, and Table 23 present the change in attitudes from pre to post survey. The tables are consolidated by technology field, and show the averages grouped by gender and then combined as an overall average. Possible survey responses ranged from 1 to 6, with 1 representing the less favorable attitude. A change score was calculated for each student, for each category, by subtracting the response in the pre survey from the response in the post survey. Thus, a positive number represents a move toward a more favorable attitude, and a negative number represents a move toward a less favorable attitude. The averages were calculated from these change scores.

Computing attitudes. As shown in the Table 20, only small average changes occurred; this was anticipated because of the short time lapse between the pre and post measures. The most extreme change was a student who had a change score of -3 for the question about computing being unimportant or important. The majority of the change scores were +/-1. (22 of 27 changes.)

Table 20

Change Scores in Attitudes about Computing, Post - Pre

Gender	Mundane/ Fascinating	Unappealing/ Appealing	Unimportant/ Important	Hard/ Easy	Boring/ Exciting
F (7)	0.71	0.14	-0.71	-0.14	0.43
M (5)	0.00	0.00	0.20	0.00	0.20
Comb	0.42	0.08	-0.33	-0.08	0.33

The individual changes for the twelve students who took both the pre and post Computational Attitudes Survey are shown in

Figure 55 for the Unappealing to Appealing scale regarding computing. The possible ratings ranged from 1 to 6. For those students whose ratings rose, the bottom of the green bar is where their rating began, and the top of the green bar is where their rating ended. For those students whose ratings fell, the top of the red bar is where their rating began, and the bottom of the red bar is where their rating ended. Eight students experienced no change, two students had a slight rise and two students had a slight fall. Because there were only a few weeks between the pre and post surveys, it is not surprising that there was little change.

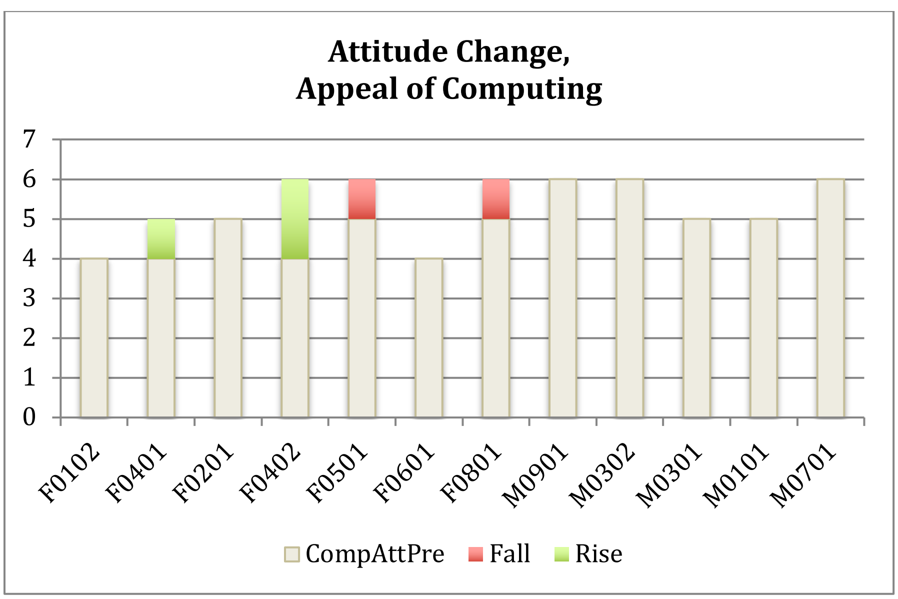


Figure 55. Attitude change, appeal of computing.

Electronics attitudes. Table 21 shows change scores in attitudes about electronics. The changes are the difference between the pre and post ratings on the Computational Attitudes Survey. Again, as shown in Table 21, only small average changes occurred. In this case, the change scores were all +/-1 or +/-2. The majority of the change scores were +/-1. (28 of 35 changes.)

Table 21

Change Scores in Attitudes about Electronics, Post - Pre

Gender	Mundane/ Fascinating	Unappealing/ Appealing	Unimportant/Im portant	Hard/ Easy	Boring/ Exciting
F (7)	0.14	0.14	-0.14	0.43	0.29
M (5)	-0.20	0.40	1.20	0.00	0.20
Comb	0.00	0.25	0.42	0.25	0.25

Figure 57 shows the individual changes for the same twelve students who took both the pre and post Computational Attitudes Survey. The chart shows the Unappealing to Appealing scale regarding e-textiles. The possible ratings ranged from 1 to 6. As before, for those students whose ratings rose, the bottom of the green bar is where their rating began, and the top of the green bar is where their rating ended. For those students whose ratings fell, the top of the red bar is where their rating began, and the bottom of the red bar is where their rating ended. Six students experienced no change, four students had a slight rise and two students had a slight fall.

E-Textile attitudes. Consistent with the findings about change in computing and electronic attitudes, Table 22 only small average changes occurred in attitudes regarding e-textiles. In this case, the change scores ranged from +/-1 or +/-4. One student had change scores of -3 for unimportant to important and boring to exciting. Another student had a change score of 4 for hard to easy. However, the majority of the change scores were still +/-1. (27 of 36 changes.)

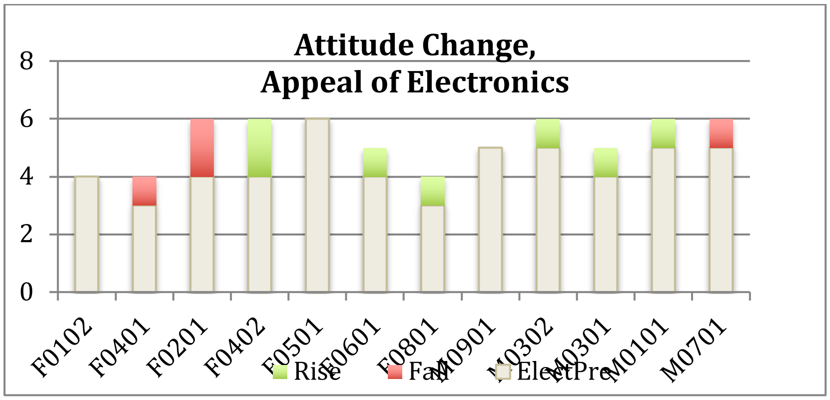


Figure 56. Attitude change, appeal of electronics.

Table 22

Change Scores in Attitudes about E-Textiles, Post - Pre

Gender	Mundane/ Fascinating	Unappealing/ Appealing	Unimportant/ Important	Hard/ Easy	Boring/ Exciting
F (7)	0.57	0.29	0.29	-0.14	-0.14
M (5)	0.00	-0.60	1.20	-0.80	-0.40
Comb	0.33	-0.08	0.67	-0.42	-0.25

Technology attitudes. The average change scores about technology in Table 23 again show only small average changes. The change scores were +/-1 or +/-2. The majority of the change scores were still +/-1 (18 of 20 changes.)

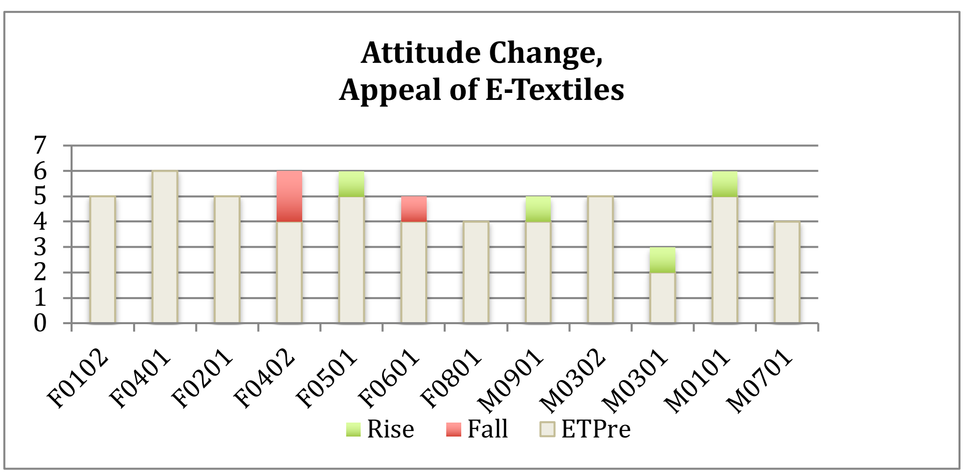


Figure 57. Attitude change, appeal of e-textiles.

The chart shows the individual changes for the twelve students who took both the pre and post Computational Attitudes Survey. Figure 56 shows the attitude changes on the Unappealing to Appealing scale regarding electronics. The possible ratings ranged from 1 to 6. As before, for those students whose ratings rose, the bottom of the green bar is where their rating began, and the top of the green bar is where their rating ended. For those students whose ratings fell, the top of the red bar is where their rating began, and the bottom of the red bar is where their rating ended. Three students experienced no change, six students had a slight rise and three students had a slight fall.

Table 23

Change Scores in Attitudes about Technology, Post - Pre

Gender	Mundane/ Fascinating	Unappealing/ Appealing	Unimportant/ Important	Hard/ Easy	Boring/ Exciting
F (7)	0.14	0.14	0.29	0.14	0.00
M (5)	-0.20	0.20	-0.20	-0.20	-0.20
Comb	0.00	0.17	0.08	0.00	-0.08

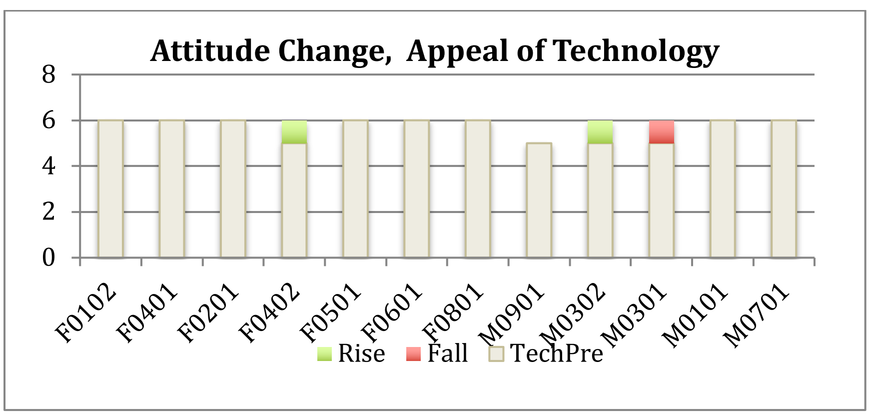


Figure 58. Attitude change, appeal of technology.

Again, the individual changes for the same twelve students who took both the pre and post Computational Attitudes Survey are shown. The chart shows the Unappealing to Appealing scale regarding technology. The possible ratings ranged from 1 to 6. The rise and fall columns

are constructed as in the preceding individual charts. Nine students experienced no change, two students had a slight rise and one student had a slight fall. Eight students' ratings remained at the maximum of six, two others rose to six, and one fell from six to five.

Overall attitudes toward computing, electronics, e-textiles, and technology. Table 20, Table 21, Table 22, and Table 23 presented data for changes in attitude measured by the computational attitudes survey. Rather than focusing on the changes, Table 24 shows averages of the EDLT740 students' attitudes toward computing, electronics, e-textiles and technology at the end of the learning module. Since the data does not compare pre and post attitudes, data from all 14 of the completed computational surveys are included. The scores were calculated by averaging the scores for the mundane/fascinating, unappealing/appealing, unimportant/important, and boring/exciting responses, by technology field. In each of the four ranges, a score of six represented a favorable attitude, so averaging these responses gives a single score, representing attitude toward that technology. The hard/easy response was not included in this average, since it is a different affective dimension.

Table 24

Average Attitudes Toward Field, Post

Gender	Computing	Electronics	E-Textiles	Technology
F (8)	5.1	5.0	4.8	6.0
M (6)	5.5	5.1	3.8	5.8
Comb	5.2	5.0	4.4	5.9

As would be expected from a group of students who have chosen to major in learning technology, the overall attitudes towards the technology fields are high. A score of 6 represents the biggest positive score that a student could select as a response, with a score of 1 being the

lowest score a student could select. While not necessarily a low attitude score, the e-textile score is the lowest of the measured attitudes and shows the biggest difference between the genders.

Evidence of Appeal in Reflection Papers

Each of the EDLT740 students wrote a reflection paper at the end of the semester as part of their required course. The reflection papers explored the impact that the course had on the students and what they learned. Several students specifically mentioned the debug activities.

Toughest moment. Student F0401 said "My toughest moment would be the debugging scenarios during our face-to-face meetings. When working with my partner I felt so rusty about my own skills that it was unnerving to hear their solutions but not be able to follow the reasoning behind the solution."

Influence on project. Student M0902 spoke of his debug experience and how it influenced his project.

> The debugging workshop got me sewing and before I knew it, not only was I immersed in sewing my project, but I actually kind of enjoyed it. The most rewarding moment of the entire course came when I finished sewing and turned my project on for the first time, and everything worked as it was supposed to! It was a feeling of wonder and amazement I might have to reach back to childhood for a proper comparison.

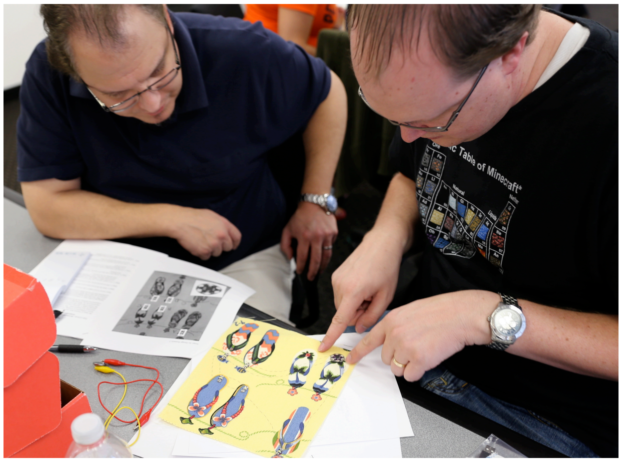


Figure 59. Group 9 debugging.

Another student commented on feeling that the debugging experience had impacted her project. Student F0402 said, “Once I was able to problem solve the coding issues during our face-to-faces, I felt like I had a better handle on my own Arduino project.

Competitive atmosphere. Student F0702 commented on the competitive atmosphere she felt during the debugging exercises. “I didn’t like the competitive atmosphere of people finishing early. I noticed some people didn’t feel good because they couldn’t finish quickly.” She makes the following suggestion, “For next semester I suggest allowing people to debug at home and then having the discussion in the forums and in class as preferable to an in-class activity.”

Figure 60. Group 7 collaborating.

Length of time to do debug exercises. Another mention of the debug exercises in the Reflection papers came from student F0802. She commented, “I also think the debugging exercises could have been more concise, in order to maximize on the energy and attention of the students in the class.” This student’s statement is a suggestion for how to change the debug exercises, but also reveals that during the course of the exercises she felt a loss of attention and energy.

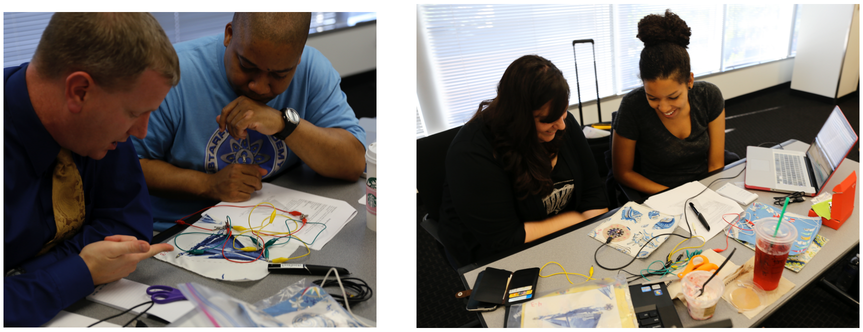


Figure 61. Group 3 and Group 4 checking e-textile behavior.

Reflections on the larger e-textile experience. Many of the student reflections were about the larger e-textile experience, including the making of their own projects. Some of these reflections are given below.

Take CT to others. Student F0201 wrote of the impact that her computational experiences in the course had on her.

> I wish that in my earlier schooling I had taken classes in computer programming, computer science or anything that had to do with computational thinking. I was never exposed to any of this in school and really didn't know what it was all about. I think that all children should be given an introduction to these skills. I don't feel like I even had the option to explore this in high school or college because I was never exposed. Exposing children to computational thinking through making things with Arduinos and LilyPads or making apps through app inventor would be a great introduction . . . When I have a little extra time (definitely after I finish this program) I want to create a Makers club, here in Denver, for children who may not get this through their schooling.

Another student, F0202, was also moved to create a Makers club, but she didn't wait. She started a club during the same semester.

> Although I personally struggled to create elegant artifacts, I succeeded in creating a maker space for my students to explore and gain knowledge from my struggles. When I began the IHS Coding Club in August of 2013, I never imagined how successful it become. Their little community has flourished beyond my wildest imagination. Every Saturday, students come to school to work on coding and programming because they want to, not because they have to. Seeing them so excited has been the best outcome from taking this class.

Textile aspect. Student F0102 talks about how the textile part of e-textiles drew her into the project.

> The Arduino project was an extension to my learning and a nice addition to my talents as a seamstress. E-textiles were a perfect fit. I struggled with the coding, terminology, engineering, and electronic components of the design but was very comfortable with completing the final project. I knew that once I had the coding figured out and a "storyboard", I was home free.

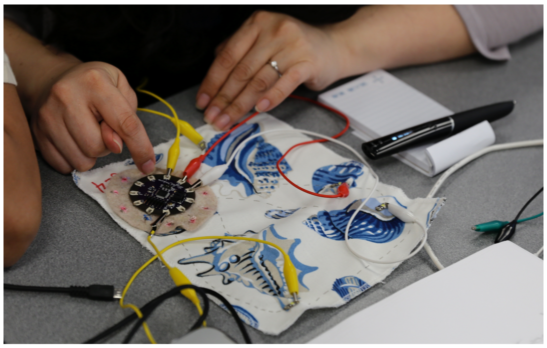


Figure 62. Pointing to the LilyPad controller.

Power of community. Student F0702 spoke of the power of community she experienced as she worked on her project.

> I truly enjoyed this class. Learning how to both use the Arduino and create a working project with an academic lens was not only very interesting, but also it became a project that I loved working on. My project was very much like a puzzle that needed to be solved. I had to reach out to several communities for help and met some interesting people along the way. I was truly amazed by how friendly and motivated people were to help me with my project. It became a problem that they wanted to solve too. I also appreciated the forums on the Adafruit site. Any question I posted was usually answered within a few hours.

Importance of making. Student F0402 wrote of how she learned the importance of making while she worked on her project.

> I was excited for the opportunity to get out of my comfort zone again and stretch my learning, I saw the Arduino and AppInventor portions of this course as a perfect means to that end. While making to learn, I began to understand the importance of exploring, creating, reworking, applying, and problem solving to create products that I am proud of and can happily present to others as worthwhile learning experiences.

Effect on choice of projects. One student, F0501, commented on how her experience would affect her future choices of projects she assigned to her own students.

> In this case, I had the choice to make it something I was interested in, as complex or easy as I needed it to be, using materials that I determined, while finding resources that were unique to my projects. Although many people used the LilyPad component, only one other person utilized the temperature sensor, which was difficult to understand and program. However, having the opportunity to design projects unique to our needs not only taught me that my ideas matter, but most importantly, that other's ideas matter as well. As an instructor, I am guilty of narrowing a project down so that all students in the group can accomplish it. This exercise made me think more broadly and allow for flexibility in future project I may assign.

Stressed by CT experiences. Not all the student experiences were positive. Student M0502 wrote of the experience in a way that shows how he was stressed by his experiences.

> What made the project even more arduous was how I wish we could have spent more time in class learning the coding language needed to create these projects. The thinking involved brought memories of math courses in high school, and a dreary feeling left me nauseated until the completion of my project.

After two of his cadre mates helped him make progress with his project's code, the student wrote more positively:

> I found this course intimidating yet satisfying. I experienced great victories from small gains by attempting to code without previous background. In the end, all our tasks for EDLT 740 left me satisfied with what I accomplished on my own.

Effect on identity. Student F0801 wrote of spoke of how the challenges of the course affected her identity.

> My identity may have taken a swift punch to the gut this semester, as I felt like a novice unwelcome to an elite and foreign practice called the Maker Movement. The low baseline knowledge I had, poor sense of efficacy and general lack direction often made the work feel like labor sans learning, without any hope of attaining mastery.

Student F0601 wrote of more positive changes to her identity.

> By the end of the semester, I found myself in conversations with others in the community discussing my journey as moving from tinkering to making. I chatted with engineers about my efforts to mix conductive paint and was told I was thinking like an electrical engineer. I spoke with artists about my pop-up book plans, and I was asked if I was a bookmaker. I am not an electrical engineer or a bookmaker, but I am now a confident maker. I look forward to continuing to tinker, and make, and think about how both might be better integrated into the community and formal education.

Evidence of Appeal in Group Debrief

Students commented during the debrief session that working on someone else's work was both helpful and challenging. There was general agreement on the value of doing the exercises with a partner. Several suggested changes to the learning module that were mentioned during the group debrief give an indication of the things that the students felt could or should be improved.

- Debug the first three engineering challenges at a different time than the last three coding challenges
- Sew the bee first, make it required
- Lessen the impact of other groups (both competition and distraction)

Evidence of Appeal in Learning Module Evaluation Questionnaire

Several questions from the Learning Module Evaluation Questionnaire were related to whether the learning module appealed to the students.

Most enjoyable. The students were asked, "What part of the e-textile learning module did you find most enjoyable?" The things the students enjoyed most are listed below.

- Having a working project (4)
- Coding and Debugging (3)
- Getting the lights working
- Problem solving
- Problem solving with my partner
- NOT coding

Other to share. The students were asked, "Do you have anything else you would like to share? Some of their responses that are related to the appeal of the learning module are listed below.

- Fun and enjoyable
- Fun and tough
- Excellent experience. Plan to use it
- Enhanced my appreciation for coding
- Fun, hands on, problem-solving, creative, integrates all parts of learning
- Being left to learn coding completely on our own was frustrating

Evidence of Appeal in Forums

Students made some comments about their experiences as they posted in the forums.

Student M0701 stated:

> While the specific tools that we played with during this course-- Arduinos and App Inventor--were new to me, the DIY/Maker mentality is very much a part of my life, and I greatly enjoyed the opportunity to jump into this familiar mindset for the projects in this course. The various components and assignments helped me to think metacognitively about this "Maker mentality" and arrive at several character qualities that making and tinkering seem to instill in those who practice these skills.

Evaluation of the Design

Results in a design-based study include feedback about the effectiveness of the design.

The effectiveness of this learning module will be evaluated by examining:

- Learning Module Evaluation Questionnaire
- Student remarks made during the group debrief
- Student statements in their reflection papers
- Student posts in the course forums
- Student comments during the debug activities
- Researcher Observations

Design Evaluation, Learning Module Evaluation Questionnaire

Results in a design-based study include feedback about the effectiveness of the design. The students responded to an anonymous questionnaire evaluating the learning module, and also gave feedback in a cadre discussion.

Questionnaire results: Close-ended questions. Appendix F contains a copy of the Student Evaluation of the Learning Module questionnaire. All 18 students responded to the questionnaire. The student evaluations of the learning module provided input for the changes to the activity set that are discussed in Chapter 5.

The data from the close-ended questions on the evaluation questionnaire are summarized in several tables, and then the responses to the open-ended questions are discussed. Question 4 from the Learning Module Evaluation is shown for reference. Following question 4, Table 25 presents average response scores

4	I had all the materials I needed to learn about e-textiles.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree

Figure 63. Sample question from learning module evaluation.

E-Textile elements. Table 25 presents the questions from the Student Learning Module Evaluation that directly relate to the e-textile aspects of the activity set. The question about whether the students liked experimenting with e-textiles had an average agreement score of 5 (6 was the highest possible score), indicating that the students found e-textiles an enjoyable technology. The average agreement score about being more interested in the traditional Arduino type of project that involves soldering was similar for both the male (3.4) and female students (3.6).

While the debug design scenario activities were done with e-textiles, several class members chose to do more traditional Arduino-type projects, so all students did have some exposure to those type of projects. An average agreement score of 4.6 showed that students generally felt they had the needed materials. The average agreement score of 2.6 shows that most students did not find the backgrounds of the debug sets to be overly distracting.

Table 25

Learning Module Evaluation, E-Textiles (Disagree to Agree, 1 to 6)

Gender	Had Needed Materials	Debug Sets Background Distracted	Liked Experimenting with E-Textiles	More Interested in Arduinos Using Soldering.
F (11)				
Avg	4.2	2.8	5.2	3.4
Min	2	1	3	1
Max	6	6	6	6
M (7)				
Avg	5.1	2.1	4.7	3.6
Min	4	1	3	2
Max	6	6	6	6
All				
	4.6	2.6	5.0	3.4

Weighting of course elements. Another aspect of the evaluation was concerning how the course emphasis was weighted. The students were asked to select their level of agreement with questions regarding the amount of time spent on programming and on circuit sewing. They were also asked about whether programming or circuit sewing was more difficult. Table 26 shows the average agreement scores from the study participants to these questions.

The low overall average agreement score of 2.2 for the question about spending too much time with programming and the same score for the question about spending too much time with

e-textiles indicates the students felt they could have used more time with both these aspects of the design scenario activities.

The overall average agreement score for needing more time with e-textiles was 4.1 and for needing more time with programming was 4.9. Student comments in both the group debrief and the open-ended learning module evaluation especially emphasized their feeling that they needed more instruction and time learning about programming.

Table 26

Learning Module Evaluation, Balance of Elements (Disagree to Agree, 1 to 6)

	Needed More Time with E-Textiles	Spent Too Much Time with E-Textiles	Needed More Time with Programming	Spent too Much Time Programming	Programming Was Harder than Circuit Sewing	Circuit Sewing Was Harder than Programming
F (11)						
Avg	4.8	2.0	5.0	2 .4	4.0	2.5
Min	3	1	3	1	2	1
Max	6	3	6	6	6	5
M (7)						
Avg	3.0	2.4	4.9	1.9	3.6	3.4
Min	2	1	4	1	1	1
Max	4	5	6	3	6	5
All (18)						
	4.1	2.2	4.9	2.2	3.8	2.9

Usefulness of elements. Students were asked to rate whether different elements of the learning module were useful, again using a scale from 1 to 6, with 1 indicating complete disagreement and 6 indicating complete agreement. Table 27 shows a summary of the student responses.

The overall agreement scores were 4.6 for the Debug Forum being useful, 3.6 for the class Wiki being useful, and 3.2 for the Project Notebooks being useful.

These course elements, along with the students' opinions of their usefulness, were considered in design changes suggestions for future iterations of the debug activity set. These design changes are discussed in Chapter 5.

Table 27

Learning Module Evaluation, Usefulness of Elements (Disagree to Agree, 1 to 6)

	Debug Forum Useful	Class Wiki Useful	Project Notebook Useful
F (11)			
Avg	4.4	3.1	3
Min	2	1	1
Max	6	5	6
M (7)			
Avg	4.9	4.1	3.4
Min	4	3	1
Max	6	6	5
All (18)			
	4.6	3.6	3.2

Questionnaire results: Open-ended answers. In addition to the 16 close-ended questions of the Learning Module Evaluation Questionnaire, where the students indicated level of agreement, the questionnaire had six open-ended questions to which the student could respond. All 18 students completed the questionnaire, but not every student answered every open-ended question. Student responses to each question are summarized below.

Suggestions for improvement. The students were asked, "What suggestions do you have to improve this e-textile learning module for future students?" Many of the suggestions were about providing more instruction. Five people said more programming instruction, three people said more circuit instruction, and one person said more Arduino instruction.

There was a suggestion to have all students start with a common, easy project. This suggestion was also stated in the group debriefing that occurred at the end of the first six debugging exercises. When the suggestion was voiced, there seemed to be a lot of general assent expressed throughout the group.

A practical suggestion was to have future groups of students hold off buying Arduino parts until after they had decided on a project. A general list of Arduino parts had been created and was listed as required on the course syllabus. The actual projects created by the students were much broader than originally anticipated, so there were parts from the list that they did not use, and many additional parts that they had to buy.

One student made a suggestion to differentiate between people who were beginners and those who were more advanced. While the students did complete a Computer Experience Questionnaire, this was not done until several weeks after the course started, and the questionnaire results were not used in any way to make a difference between the projects that they completed, how those projects were graded, how the debug exercise partnerships were formed, or the debug exercises that they did. Since the background of the students in Pepperdine's Learning Technology doctoral program varies widely, it was actually not known what the computational experience level of the students would be, prior to the beginning of the course and as the debug exercises were planned. It was anticipated that many of the students would not have prior coding experience.

There were certain aspects of using e-textiles that contributed to making the debugging more challenging. Some of these aspects will be discussed in Chapter 5, as design considerations for future iterations. One student made the suggestion to use a more standardized application rather than hand-made e-textiles for debugging exercises.

One suggestion was to have music playing in the background as the groups worked on the debugging exercises and another suggestion was to have easier assignments. The student did not say if the easier assignment suggestion was in reference to the debug exercises or to their individual project.

Most enjoyable. The students were asked, "What part of the e-textile learning module did you find most enjoyable?" The responses to this question are discussed in the section of this chapter called Appeal to non-computer science majors.

Change in attitude towards computational thinking. The students were asked, "Did your attitude towards computational thinking change?" Seven students responded yes, two responded no they experienced no change, because their attitudes were already favorable before, and two students simply said no. Several of the students who said yes added how their attitude had changed. These changes were:

- More difficult than I assumed
- Has positive applications for K – 12
- Made me think in phases
- I knew this was hard, but excited that I learned as much as I did
- I liked problem solving in context

Outside sources. The students were asked, "What outside sources did you use to help you with your e-textile project?" The students used a variety of sources, as can be seen in their responses below.

- Cadre Mates (5)
- YouTube Videos (3)
- DIY Videos (2)

- Arduino Cookbook (2)
- MIT (2)
- Google (2)
- Arduino Website and Forum (2)
- Reading Blogs
- Learning Partner
- Codebender.org
- Other teams
- Other Experts
- Arduino Code Library

Most used resource. The students were asked, “What resource did you turn to most often?” The replies are listed below.

- Cadre Mates (7)
- TA/Researcher (2)
- Arduino Website (2)
- LilyPad Protosnap Forum
- Sketchpad
- Mit.edu
- Instructables and YouTube
- Colleagues

Other to share. Finally, the students were asked, “Do you have anything else you would like to share? Some of the responses to this question had to do with how the learning module appealed to the students. These responses are discussed under the section Appeal to non-

computer science majors in this chapter. Other responses were suggestions for improvement of the learning module and those responses are listed here.

- We should have started with the bumble bee project (The LilyPad Lightning Bug)
- Less stressful with a F2F learning experience and the entire semester to work on the project.
- Being left to learn coding completely on our own was frustrating

Summary: Design Evaluation, Learning Module Evaluation Questionnaire

As will be seen in the following discussion of the Group Debrief, many of the points emphasized in the questionnaire were also brought up during the group debriefing. However, the Learning Module Questionnaire was anonymous; students were able to completely express their feelings of frustration and/or enjoyment, without being influenced by what the professor, researcher, or cadre mates would think. The questionnaire gave insight into the student experience with the debug experience and brought out suggestions for improvement that will be incorporated in future iterations of the activities.

Design Evaluation, Group Debrief

As described in an earlier section, students participated in a group debrief with the course professor at the conclusion of the first six debugging exercises. Student feedback that relates to evaluating the learning module is given below.

Comparison of debug exercise and project. Students were asked how the debug activities related to their experiences with their own projects. Student F0802 said:

> The first three that we did, I thought what I had done before made that relatively simple, but the debugging was so different from how I had constructed mine or coded it, I was really confused.
> (Group Debrief Audio, 00:00:19)

Later the topic of working on someone else's materials comes up again. Student M0901 compares the experience to writing.

> I think that is a really helpful skill . . . I know that when I work with writers, one of the things that I like to have them do is to read someone else's work . . . and I think that when you deal with something that is completely new, it takes you back to that position where I have no idea what is going on here, so I have to go to the very basics. Let's go look at the wiring, let's go look at the code and try and figure out what they were trying to do . . . Step through it and see where the problems are. So, I think that this is a great learning tool for approaching the content.
> (Group Debrief Audio, 00:04:59)

This student's comparison of the process of creating through coding to creating through writing is actually an example of the computational thinking perspective of connecting. The comments are also related to the emergent computational thinking perspective, Adapt to Other's Code, which was discussed in a prior section of this chapter.

Student F0601 and the course professor discussed the comparison between the debug exercises and her project:

> F0601: I just feel like, for me, that I learned a lot today, and I probably would have done the project differently. I couldn't . . . I didn't really understand what the capabilities were until today. And . . . I know we were told, but still being told and fully understanding . . .
> Professor: So, what kind of capabilities, for instance, are you thinking of?
> F0601: Um . . . Just, I guess, how much I could . . . have so many different things going on at once.
> (Murmurings of agreement in the background.)
> (Group Debrief Audio, 00:01:12)

The student's realization of how multiple things could occur at the same time was probably due to exposure to the variety of blinking patterns presented in the different design scenarios, particularly with Bug 5.

Working with partners. The debug activities were designed for pairs of students. Once purpose of this design aspect was to be able to have students externalize what they were

thinking. While there were times in the debug transcripts that it was obvious that partners were pulling in opposite directions with their ideas of how to debug, student F0501 said,

> I think it was really good to have that other person, that other set of eyes, looking at the code . . . verbalizing the process really helped.
> (Group Debrief Audio, 00:08:22)

Student F0202 also made an observation about partners. She commented about how she and her partner were able to capitalize on each other's strengths.

Effect of Environment. Student M0302 spoke about how the environment of working in the same room as the other groups affected the experience for him.

> I wonder how it would be if it was more in an isolated environment, like, if we were all in separate rooms . . . with your partner. Because it was irritating having <cadre mate name> go, "Oh, we got it!" I'm half joking . . . But it is kind of interesting to think of it too. We had a lot of variables to consider while doing this. What is interesting is that this exercise was being done with a lot of input in the ambience of the classroom and the atmosphere. We were also trying to write down and record with Livescribe and the Echo pen at the same time you are talking through your documentation and you are working through the problem with one person holding something and someone else trouble shooting over there. So, I think it says a lot, at least for the people in this program, what we are capable of doing, given that all of this stuff was done under incredible pressure, because you are adding all of these things. What was nice about it was that you have the work, but none of it was really yours. And I think that element is an important element. Because if this had been my project, that I was working on, with all this crazy stuff, I wonder if my response to that would have been much more frustrating or something, than just 'Victoria wants the flag to light up . . .'
> (Group Debrief Audio, 00:08:44)

In response to the previous students' feelings, student F0601 commented that she didn't feel stressed at all, and that she thought they were all happy and engaged.

Reusable LilyPad. Because of the number of bugs and groups, the debug exercises were designed to have each group use the same LilyPad for Bugs 4, 5, and 6, in order to reduce cost. This was accomplished by sewing the LilyPad to felt that had metal snaps on the bottom, and then sewing the other side of the snaps to the textile portion of the 3 bugs. Conductive thread connected the LED's to the bottom snaps, and conductive thread connected the petals of the

LilyPad to the snaps on the felt. Thus, when a powered LilyPad was snapped on, the snaps conducted the power from LilyPad petals to the LED's.

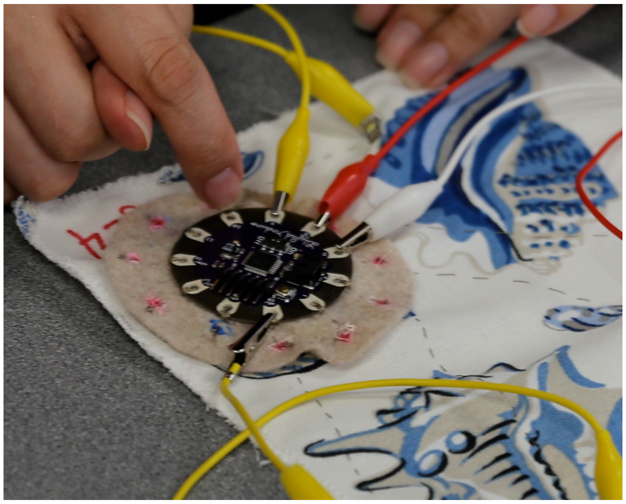


Figure 64. Reusable LilyPad.

Rather than having the snaps being in a perfect circle, one snap (near the blue dot in the photo above) was slightly closer to the center. This was meant to mark the placement of the LilyPad on the material. Basically, the design was trying to do a homemade version of the product LilyPad SimpleSnap and SimpleSnap Protoboard shown below.

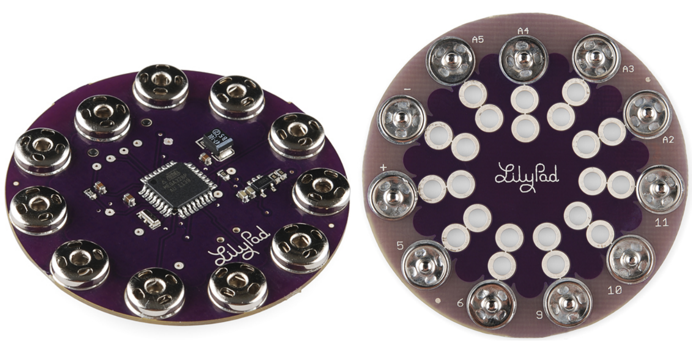


Figure 65. LilyPad SimpleSnap product.

This clue about the placement was not clearly communicated to every team, however, and figuring out the placement became one of the challenges that some students dealt with. Student F0402 is referring to turning the LilyPad to find the proper position in her remarks.

> And one of the things, when we got to Bug 6, we had to keep turning the LilyPad. Because at some point it labeled what should go where. You know, what should be snapped into what spot, but I think 5 and 6 didn't. So, we had to talk t it through and keep twisting and turning and turning, until we finally lined it up correctly. Because the directions in the picture gave us some of the information, but the rest of the information to label it, or snap it into place, came from the code itself.
> (Group Debrief Audio, 00:18:49)

Attitude Change. Student F0202 discussed how her attitude changed about working with Arduino materials.

> In the beginning of this project, three or four weeks ago, I was so afraid to plug in anything and do anything. And then after my Flora Board just blew up, I was like, 'I don't even care. I'll get another one.' I had this attitude of, 'whatever'.
> (Group Debrief Audio, 00:17:16)

Suggestions. During the group debrief, several suggestions were made for future iterations of the course.

One suggestion was to break up the debug experience into two parts: have the first three, engineering challenges near the beginning of the course and then do the coding challenges toward the middle of the course. (Group Debrief Audio, 00:02:57)

Another suggestion was that it should be required that everyone should sew the 'bee' first. The 'bee' is really the LilyPad Lightning Bug project that comes with the LilyPad Sewing Kit, with LED's, battery, button and a switch. A picture of the Lightning Bug project is shown below. (Group Debrief Audio, 00:03:56)

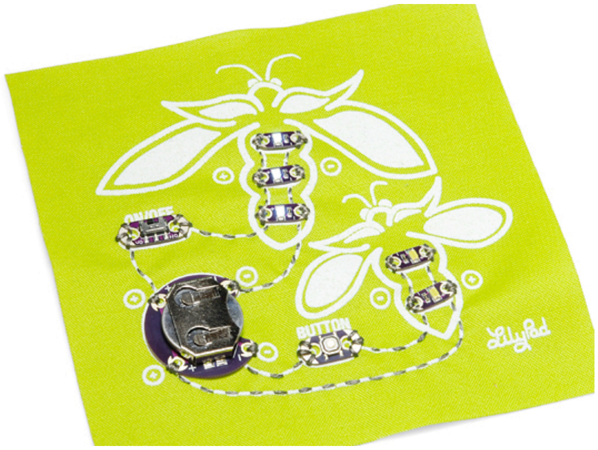


Figure 66. LilyPad sewing kit lightning bug.

Design Evaluation, Reflection Papers

Some of the reflection papers contained comments that help evaluate the learning module. Excerpts are given below.

Length of time on debug exercises. Student F0802 commented, "I also think the debugging exercises could have been more concise, in order to maximize on the energy and attention of the students in the class." The length of time to complete the six debugging activities was a strain for some students.

Student F0802 also made the following recommendation:

> **Begin Arduino experimenting early.** I would recommend an earlier, and perhaps more collaborative, start to the Arduino project. In an organized manner (with a clear plan of attack), recommend students to start tinkering with their Arduino components throughout the month of August. I would also recommend that the first two or three sync sessions be spent creating a basic circuit (in class, together). These small steps would have helped me immensely as I struggled with my Arduinos.

Design Evaluation, Forum posts

After the second F2F meeting where the students worked on the final bug, Bug 7, there was not time for another group debrief. Instead, several questions were posted in the course forum. Students' responses that relate to evaluating the learning module are included here.

Clarity of instructions. The following post from student M0901 refers to some of the instructions in the design scenario for the infamous Bug 5.

> I echo Joey's comments about number 5 being the hardest to debug, probably because there was the mislabeling of the one LED, but also because my partner and I (well, okay, mostly me) had questions about the definition or intended function of the one that "blinked slowly." The one LED was blinking quickly, on and off for the same amount of time. When we got the "slowly blinking" one going, it was on for a long time but off for a short time. To me, this wasn't blinking slowly; it was just blinking less often. So, I kept wanting to go back and get it to blink slowly, off and on for the same amount of time, just twice as long as the first (or something like that). I was...compelled...to move on. ;)

Design Evaluation, Debug Exercise Transcripts

In the process of debugging, students made numerous comments that were used for input on the next iterations of the learning module that are discussed in Chapter 5. Some of their comments are presented here to evaluate the design.

Learned new. Several students gave indications that they learned new things during the debug exercises. For example, the students learned about a way to easily prototype by using magnets to attach a battery to the alligator clips.

> F0201: She’s telling them what the magnets are for.
> F0201: What are the magnets for?
> VKim: Um. . . It’s a way . . . an easy way to connect to your . . .coin cell battery.
> F0201: Oh. Instead of putting it in the little thingy?
> VKim: Yeah. Yeah. You can do it . . . that way . . .
> VKim: That’s just another way.
> F0201: But you can use the little magnets?
> Vkim: Yeah.
> (Group 2, Bug 1 Transcript)

The following student, working on Bug 7, solidifies the idea that separate behavior requires separate pins.

> F0702: And is this because it’s by itself? So we need to have this . . . um . . .
> F0702: two pins?
> (Group 7, Bug 5 Transcript)

Student F0602 got her first hands on experience with e-textiles during the debug experience, since her project had been with paper circuits; she said:

> F0602: So I did all paper circuits…so this is kind of teaching me as we're going
> (Group 6, Bug 1 Transcript)

Applied previous learning. Not only did students learn new things as they worked on the bugs, they also had a chance to apply things they had learned previously. Student F0602 applied what she had learned about preventing short circuits.

> F0602: Can we test it by interrupting it? Can we put something in between so they're not touching? Does that make sense?
> F0601: Yes.
> F0602: What can we put in between? Like on my paper circuit, I used tape. So that they can both work…so what could we use?
> (Group 6, Bug 1 Transcript)

Students not only applied what they had learned in working on their projects, they also applied what they had learned in working on previous bugs in the activity set. The student in the following quote had been misled about an earlier bug's behavior because he and his partner had made code changes that were not uploaded to the Arduino.

> M0301: We are not going to make the same mistake we made last time … We are going to upload the stinking code now.
> (Group 3, Bug 7 Transcript)

Celebrated success. The groups consistently celebrated their success as they finished each bug. This is a partial indication that they were engaged in the debugging activities.

> M0901: All right. Wahoo!
> M0902: OK! So that's problem 1 . . . solved . . .
> M0901: Done. All right . . . pull that puppy off . . .
> M0902: On . . . to . . . BUG . . . NUMBER . . . 2 . . .
> (Group 9, Bug 1 Transcript)

Group 2 also celebrated success when they finished debugging.

> F0202: I am so proud of us.
> F0202: Don't you want to take a picture?

F0202: I actually want to record it.
(Group 2, Bug 5 Transcript)

Figure 67. Groups 5 and 2 celebrating success.

Frustration. Some aspects of the debug exercises led the students to feel frustrated. An example is given below.

F0801: These things so don't match up. It's, like, f'in impossible.
<R, from another group>: What do you mean - they don't match up?
F0801: LED 17, but it's already coded into something. But it's, like, attached to a different label.
(Group 8, Bug 5 Transcript)

Group 5 also expressed some frustration as they worked on Bug 5.

M0502: That's a problem, right there.
F0501: So this is wrong.
F0501: This is minus. No, that's wrong.
F0501: Sigh . . . OK. Let's do this again.
(Group 5, Bug 5 Transcript)

Figure 68. Group 8 members sewing a fix.

One source of frustration for the students was that were elements in the code with which they were unfamiliar. The following is an example of this.

> M0302: Yeah, I am just throwing it out there cause it's the only character that I don't even know what the hell that means.
> M0302: Like it doesn't seem to mean anything to me.
> M0302: So, I am wondering. You could have two errors. One could be a typo.
> (Group 3, Bug 5 Transcript)

Humor. There were instances of humor as well as frustration as the students participated in the debugging exercises.

> M0701: OK. So . . . what did I do . . .
> M0701: To make that come on?
> F0702: Yeah. What did you do, over here, where we had it here . . . ?
> Laughter
> F0702: Taken a picture.
> M0701: I know.
> (Group 7, Bug 7 Transcript)

Another example of humor is seen in the following example where a student talks about his project.

> M0902: I couldn't believe my project . . . I plugged it in and said . . . let me see where I messed up . . . it worked right away.
> M0901: Nice . . . Miracles happen . . . yeah . . .
> (Group 9, Bug 7 Transcript)

Students in Group 2 use humor as they discuss solutions for solving the bug.

> F0201: Where are we now?
> F0202: We cut everything and . . . and use wire <laugh>. Problem solved.
> <laughs>
> (Group 2, Bug 7 Transcript)

Design Evaluation, Researcher Observations

In addition to the previous sources based on participant feedback, researcher observations are also input to evaluating the learning module. Some elements of the delivery of the learning module were within the control of the researcher, but others were not. The original design for

the learning module was based on a review of literature, as described in Chapter 2. The learning module guidebook in Appendix A contains the design, as planned, however the constraints of conducting the study as part of the EDLT740 course required modifying some aspects of the original design.

Effect of timing constraints. EDLT740 was a blended learning course, with most of the class meetings being held online. Two F2F sessions of five days each were held during the semester. One of the constraints that had a large impact on the design was that the first F2F meeting of the Fall 2012 EDLT740 class was held several weeks later than normal. The hands-on part of the course was split into two main parts, with the first part focusing on an Arduino application and the second part focusing on an App Inventor application. The first F2F took place about half way through the Arduino part of the course. This meant that the planned introduction to Arduino and coding could not take place in person, at the F2F meeting. As an alternative, several of the online sessions of the course were used to introduce the LilyPad and the Arduino programming environment. However, the online sessions were held only once a week and were only one hour long. This was not an optimal way to be introduced to the Arduino technology, and many of the student comments indicate that they would have appreciated more programming instruction.

Another constraint was how the time with the students at the F2F was structured. The students were meeting with professors of all three of their courses over the five days of the F2F. In order to accommodate the needs of another professor, the meeting times for the EDLT740 course were combined to be all on one day, rather than being spread out over two or three days. This meant that some of the intermediate activities couldn't be done, and that the students had to do all six activities in one long session.

Competitive atmosphere. As shown by the number of times groups compared themselves to others while they were working (142 instances in the three bugs), students reacted competitively to the debug exercises. This was in contrast to the intension of the designer, which was that the working atmosphere would be low-key, fun, and focused on learning. Having all the exercises done consecutively probably contributed to the competitive atmosphere. Also, to aid in the construction of the bugs, and to add variety, each bug was made on a different fabric. This made it possible for the groups to scan the room and quickly tell which bug each of the other groups was working on.

E-textile material problems. During the process of creating the sets of seven bugs, each bug was tested. The set of bugs were completed two months prior to the F2F meeting. The week before the F2F meeting the bugs were all tested again. Some of the connections appeared to have degraded slightly over the time and needed to be fixed.

Debug activity materials. The materials needed for the debug exercises were well organized. The sets of bugs, and the Livescribe pens and notebooks were labeled to make collection of the data easier. The programs for the bugs were available for the students to download through their course shells. Design scenarios were printed for the groups, and a corrected version of the six bugs was available for demonstration.

Support persons. As the materials were passed out to the students, questions started, and some of the planned instructions were not given to the whole group. Three people assisting the nine groups was not quite enough; the review of the recordings revealed that there were times that students had questions that did not get answered.

Workshop. One evening during the F2F, after their class meetings, the EDLT740 students and professor had an unplanned and optional meeting where all those who attended

worked on their projects. Equipment was shared among the group, and students, professor, and teacher assistant pooled knowledge and assisted each other with debugging their projects. The atmosphere was congenial and collaborative; several students mentioned this workshop time in their reflection papers. Student F0201 wrote, “I found the time spent work-shopping our projects (during the October F2F) to be invaluable and really enjoyable.” Another student, M0901, wrote “The workshop time at our first F2F was critical for my project as <cadre member name> and I sat down and measured voltages with a multimeter that I didn’t have at home, leading to the final breakthrough for my initial design.” A third student M0301 also commented, “During our October Arduino workshop, soldering irons were shared, glue was given, and lots of advice flowed back and forth to everyone’s projects. Martinez and Stager’s advice on building a common workspace is well worth considering in this regard.” The same workshop feeling occurred with the students in the pilot group as they worked on their e-textile sample projects. The students wanted to keep working on the projects rather than do some additional activities that the researcher had planned. One student even came back for extra time after he put on his suit to wait for an awards ceremony to begin. The students chatted about their plans, laughed, and sewed their e-textiles.

Audio recordings. The Livescribe recordings and notebooks captured the pair debugging conversations well, although there were a few times that it was hard to distinguish what was being said above the sounds of the other groups. In addition, there were times when it would have been very useful to see the versions of the programs as they students made changes.

Partner work. The pairs worked well together and verbalized their thoughts as they worked together to find the errors. This interchange between student M0301 and student M0302 shows how they worked together:

M0302: Oh, you know what? Our board is not correctly fixed.
M0301: Oh, is that not?
M0301: Oh, and there it is.
M0301: Good catch <name>.
M0301: All right. It was a snap issue.
(Group 3, Bug 5 Transcript)

Often, the partners encouraged each other if they were frustrated or discouraged. For example, student F0501 said to her partner, as they worked on Bug 7:

F0501: It's OK.
M0502: I'm trying to wake up.
F0501: We'll get it.
M0502: Yeah.
(Group 5, Bug 7 Transcript)

Another example of the students encouraging each other is in the following conversational snippet.

M0101: I'm sorry <name>.
F0102: No, no. I think it's fine. We'll figure it out.
F0102: It's not a problem.
(Group 1, Bug 5 Transcript)

Summary: Evaluation of the Design

In the preceding discussion of the design results, the data from the Student Evaluation of Learning Module (Appendix F), Debug Exercise Transcripts, Reflection Papers, Forum Posts, and researcher observations were used as input for the evaluation of the design. Taken together, they provide context for the students' experiences during the learning module. As each new source of feedback was examined, it confirmed what was seen in the previous source, but also added additional insight on other aspects of the design.

Bugs. The e-textile bugs allowed the students to gain experience with engineering and coding that they did not have to create first, as suggested in the use-modify-create model (Lee et al., 2011). The nature of e-textiles meant that students had to consider additional factors such as

keeping the fabric flat, not letting thread ‘tails’ touch each other, and loose stitches, as they isolated and corrected the bugs. Even though the topology of all the bugs was essentially the same, the variety of the textile backgrounds and different blinking patterns were enough to keep the students from feeling like they were doing the same thing over and over.

Clearly, having multiple errors in the same e-textile project raised the difficulty and frustration level as students tried to find the bugs, as did having simultaneous actions and the fmod function part of the code. The manner of turning on the lights by sending signals to the negative end of the LED, rather than the positive end, also caused the students to spend considerably longer to debug.

Partnerships. Students supported each other as they worked in their partnerships, sharing what they learned, encouraging each other, and celebrating their successful bug fixes. Competitiveness with other groups lessened the overall emphasis on learning, experimentation, and camaraderie that was the desired goal.

The partnership work proved to be so important that the original plan of having students work individually on Bug 7 at the second F2F meeting was changed to have the students work on Bug 7 with their original partners.

Project. The students’ course projects were extremely varied and complex. Chapter 5 presents a list and photos of the projects. The debug exercises were designed to be a controlled environment where the students could experiment and ask questions to help them gain skills and understanding that would be needed for their individual projects. The end of course reflection papers illustrate that choosing and creating their own projects was challenging and rewarding for the students. The Group Debrief comments showed that the students did connect their projects with things that they learned and experienced in the debug exercises.

Sequence of learning module activities. Students strongly indicated that they did not have sufficient introduction to coding prior to participating in the debug exercises. The debug activities themselves could have been used as a way to introduce coding, but not in a competitive atmosphere where the emphasis seemed to be on finishing quickly. By the end of the sixth bug, students were tired and just wanted to be done.

Chapter Summary

This chapter has presented the results of a design-based research study that created a learning module using e-textile debugging exercises to introduce non-computer science majors in a Learning Technology doctoral program to electronics and coding, and to provide an environment where computational thinking could be manifested.

Desired Outcomes. The key outcomes sought for this learning module were:

- Evidence of computational thinking concepts, practices and perspectives
- Increased student confidence in computing abilities
- Appeal to non-computer science majors

Examples illustrating computational thinking were selected from the transcripts of student pairs working on the debugging exercises, from the student forum posts, from the group debriefing comments, and from the students' reflection papers. In addition to the original computational thinking concepts, practices, and perspectives proposed by Brennan and Resnick (2012), additional computational thinking practices and perspectives emerged during the analysis of the sources listed above. These new practices and perspectives have been presented and illustrated in context.

Impact on confidence was determined by again examining the debugging exercise transcripts and by analyzing responses to the E-Textile Self-Efficacy Questionnaire, which was given before and after the learning module took place. These sources show that students' self-efficacy was affected by participation in the learning module, but for some students the effect was a decline in self-efficacy rather than an increase. This impact is an important consideration for design changes that will be discussed in Chapter 5.

The third key outcome desired for the design was that the learning module would appeal to non-computer science majors, therefore giving a place that a more diverse group of people could experience computational thinking. Assessment of the appeal of the module was made by examining student statements in the debugging exercises, debug forums, reflection papers, group debrief, and Learning Module Evaluation questionnaire, and their responses to the Computational Attitudes Survey. Student suggestions for changes to the learning module will be incorporated to increase its appeal.

Discussion of the impact and wider implications of the results of this chapter are found in Chapter 5, where changes to the design for future iterations are also presented.

Chapter 5: Discussion

This chapter discusses the implications of the design-based study that was conducted during Pepperdine's EDLT740 (Applied Seminar in Learning Technology) course, taught in the Fall Semester of 2013. Three major types of implications are considered: implications that inform design changes in the next iteration of the learning module, implications that relate to the research design, and implications that have wider application to the field of computational thinking. Suggestions for future research are also offered.

Design Changes Based on Feedback

The first full version of the learning module contained the following activities:

1. Design scenarios for three e-textile engineering challenge bugs that the students worked on in pairs. Each design scenario contained a single bug.
2. Design scenarios for three e-textile coding bugs that the students worked on in pairs. Each bug was designed to have a single bug.
3. An Arduino project of the student's choice presented through a student-made video.
4. A design scenario for an e-textile project containing a combination of the six previous bugs. Students worked on the design scenario in pairs.

Feedback from students and observations of the researcher led to the following changes for the learning module design.

Design scenarios engineering bugs. Students began their debugging exercises with design scenarios for the first three e-textile engineering challenge bugs. Engineering bugs could be also be considered as bugs related to circuit topology. Students worked on the bugs in pairs.

More initial instruction. The transcripts revealed that some of the difficulties experienced with Bug 1were due to lack of familiarity with the materials and uncertainty about

the directions. In addition to the written instructions, more time should have been spent initially discussing the basics of how to connect the battery, possible uses of the alligator clips, and how the LilyTiny worked.

Remove randomness from controller. While theoretically possible to reprogram the LilyTiny, it was difficult to do, due to the need for special connectors and programmer. Thus, the LilyTiny controllers were kept with their original programming. The original programming had defined behavior assigned to specific pins, but the signals to the pins were sent randomly. This made it more difficult for the students to determine the behavior that their changes occasioned. In the next iteration of the learning module, controllers should have steady signals, not ones that occur randomly.

Partner work. It was evident that the students felt that working with partners was an important aspect of the debugging exercises. Students were allowed to pick their own partners, rather than being assigned partners based on prior experience with computing or electronics. Because these students were part of a cadre that had been working together for over a year, it was felt that there might be strong preferences about with whom they would prefer to work. Having the students work with partners allowed and encouraged them to verbalize their thoughts.

Individual bug per e-textile. The original Fields et al. (2012) study had all six bugs together in one e-textile. This study, however, separated each bug into a separate e-textile object. As a learning exercise, it appeared that having the bugs separated allowed the students to learn the individual concepts with less distraction.

Design scenarios for coding bugs. Another section of the debugging exercises was three design scenarios for e-textile coding bugs on which the students worked in pairs. As with the engineering bugs, each coding bug was designed on a separate e-textile object.

One e-textile for three bugs. In the first iteration of the design, each coding bug was a separate e-textile object. While the placement of the LED's and the backgrounds of the e-textiles made the objects appear differently, the stitching topology was the same for all three bugs. To lower the cost of the controllers, rather than having a separate controller for each e-textile, one removable controller was used for all three of the coding bugs. See Chapter 4 for a picture and description of the removable controller that was used. Different programs were downloaded for each bug.

The removable controller added some unreliability to the circuits because of all the additional stitching that was required to make the connections. To avoid the extra time required to sew the removable controllers and to lessen the unreliability that they added to the e-textiles, one possibility would be to have the same e-textile base for all three of the coding bugs. Then the controller could be sewn directly to the e-textile base. Another option would be to use the LilyPad Protosnap board for the coding bugs. A picture of the Protosnap board is shown in *Figure 69*.

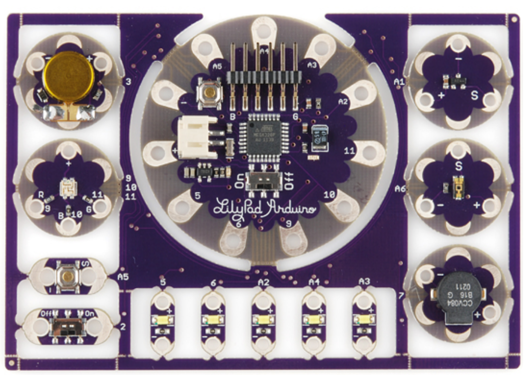


Figure 69. LilyPad Protosnap board.

The June 2015 cost of the Protosnap boards is about $57.00, so for a class of 20 students, requiring ten boards for the student groups, the cost would be approximately $1140.00, plus tax

and shipping. The boards could be reused for several years if students treated the boards gently. Use of the Protoboard would save the time of creating the e-textile projects and the connections would be more reliable, especially over time.

While in the next iteration of the learning module one textile object or Protoboard would be used for the three coding bugs, the bugs would not be combined into the same program. They would still be dealt with as separate bugs.

Simultaneous flashing using fmod function. Students experienced difficulty debugging the program that contained the fmod function that was used to control different, simultaneous flashing behavior on different LED's. If this function is included in e-textiles bugs in future iterations of the course, then it needs to be introduced to the students before, so they can experiment with and understand the function.

Control of LED's using positive terminals. As described in Chapter 4, students found it confusing to use the negative terminal to control turning on and off the LED's. If the Protosnap board is used in the next iteration of the learning module, this circuit configuration would not be possible, due to the fixed wiring pattern that is used to chain the negative terminals. If an e-textile object is used for the coding bugs, the wiring configuration will be changed so that students use the positive terminal of the LED to control on and off behavior.

Individual Arduino project. One of the requirements for the EDLT740 course was that each student design and construct an Arduino project. The students shared their projects as they put finishing touches on them during the course F2F meetings and made videos to demonstrate their projects. Students could choose e-textiles or more traditional Arduino projects. Student projects ranged widely, highlighting their diverse interests. Student projects are listed below to demonstrate the range:

- M0101: Quadratic Formula Calculating Tie

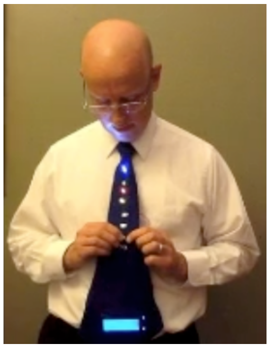


Figure 70. M0101's quadratic formula calculating tie.

- F0102: Personalized Sports Bling Bag

Figure 71. F0102's sports bling bag.

- F0201: Night Runner Belt and Leash, with different flashing patterns

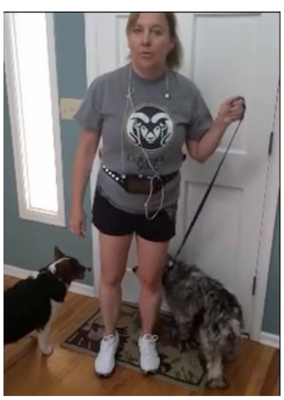


Figure 72. F0201's night runner belt and leash.

- F0202: Step Counter Shoes

Figure 73. F0202's step counter shoes.

- M0301: Water Pistol Water Level Indicator

Figure 74. M0301's water pistol level indicator.

- M0302: Building Model with Lights and Sound

Figure 75. M0302's building model.

- F0401: CurlyGirl Hair Brooch with Humidity Sensor

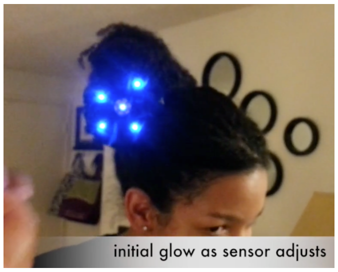


Figure 76. F0401's hair brooch.

- F0402: Stuffed Animal with Pressure Activated Music

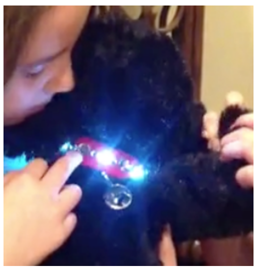


Figure 77. F0402's pressure activated music toy.

- F0501: Make-up Melt Saver with Temperature Sensor

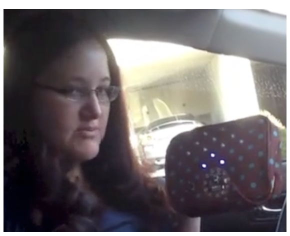


Figure 78. F0501's make-up melt saver bag.

- M0502: Night Kite with Accelerometer Controlled LED's

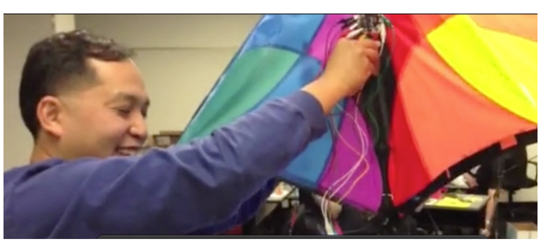


Figure 79. M0502's night kite.

- F0601: Interactive Toy for a Learning-Disabled Student

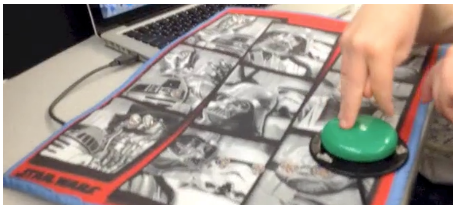


Figure 80. F0601's interactive toy.

- F0602: Paper Constellations

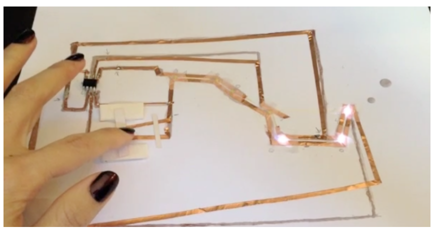


Figure 81. F0602's paper constellation.

- M0701: Town Play Mat with Traffic Lights

Figure 82. M0701's town play mat.

- F0702: Classroom Lunch-Choice Counter

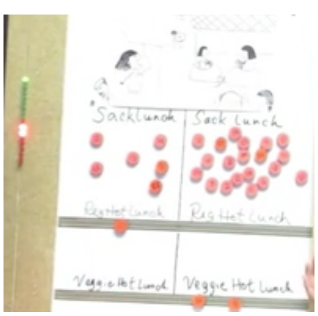


Figure 83. F0702's classroom lunch-choice counter.

- F0801: Lord of the Rings Watercolor Painting with Motion Sensor LED’s

Figure 84. F0801's hand-painted motion sensor watercolor.

- F0802: Newscast Creation Pop-up Book

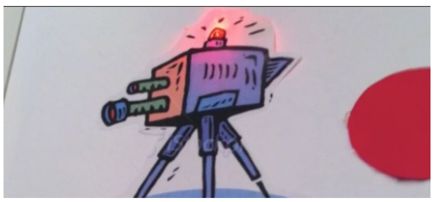


Figure 85. F0802's newscast creation pop-up book.

- M0901: Forgotten Child Detector

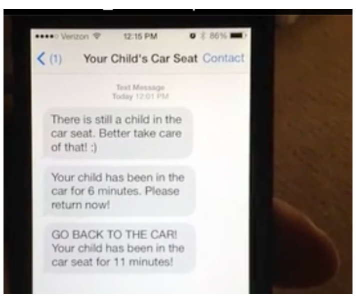


Figure 86. Message from M0901's forgotten child detector car seat.

- M0902: Temperature Sensing Onesie

Figure 87. M0902's temperature sensing onesie.

One project per semester. While they created their projects, the students did the most computational thinking and became or expanded their skills as makers. The debug exercises were designed to be a warm-up for the students to learn about and question computational ideas. The projects were an essential part of the class. During the first iteration of the learning module the students experienced two different technologies and had only half a semester for their projects in each technology. In future iterations, the students will only be required to have one major project, rather than splitting the semester with two technologies. This will give them more time to develop the complex projects that they envision.

Design scenario for multi-bug. As a post-project debug experience, students were given a design scenario for an e-textile project containing a combination of the six previous bugs at their second F2F meeting.

Partnerships. Originally Bug 7 was designed to be done by individual students, but observations of the importance of, and reliance on, partners during the first six design scenarios, led to changing Bug 7 to being done with the same partnerships as the original six bugs. In the next iteration of the learning module, Bug 7 will continue to be done with partners.

E-Textile. Because Bug 7 contains both engineering and coding bugs, it will need to be an e-textile project; the Protosnap board would not be feasible for this bug. The short circuit that was created by sewing the positive and negative terminals of the LED together on the back side of the e-textile was very difficult for the students. A hint about checking both top and bottom might lessen the students' frustration with that bug.

Final debrief. After the first six debugging exercises were completed a group debrief discussion was conducted. Interesting insights to the students' experience and thinking were shared. A debrief discussion should also be held after the students complete Bug 7.

Additional instruction. As discussed in Chapter 4, in the researcher observations section, the learning module was impacted by the scheduling of the time with the students, both in when the first F2F meeting took place and how the time at the F2F was scheduled. The planned instruction was adapted, and some was presented online during the synchronous sessions. However, there wasn't sufficient time for the students to experiment and ask all their questions.

Feeling nervous about their lack of understanding of programming seemed to be one of the main reasons that students experienced diminished confidence, so additional instruction prior to the debugging exercises and work on the project could address this issue as well.

Programming. One of the most significant and repeated feedback comments from the students was regarding their desire for more instruction about programming. In the next iteration of the learning module, preliminary instruction will be planned to help the students understand and experiment with the Arduino platform, and the C coding language.

Electrical circuits. Circuits were discussed during the online meetings, but some students indicated that they would like more instruction about parallel and serial circuits. Several students also suggested using the LilyPad Lightening Bug (see Chapter 4) as a required, common first experience with simple e-textiles.

In the next iteration of the learning module, the LilyPad Lightening bug will be used as an introductory project. Since student's rate of completion of the project will vary, it will be the last activity done before the day's activities end. During the pilot, it became evident that the students didn't like to leave even a simple e-textile project unfinished, so planning anything after starting the project is difficult.

How to use the materials. Some of the materials used in the debugging exercises were unfamiliar to the students. In the next iteration of the learning module, more time will be taken explaining the different parts of the e-textile kit and how to use them. This will make the beginning of the exercises go more smoothly, and students will be more aware of the tools that they have available.

Timing of the activities. The following is the recommended schedule for the activities in the learning module. See the guidebook in Appendix A for the full description of the activities in the list. This schedule allows time for experimenting with materials and additional instruction, as needed

- F2F1 Day 1 (2 hours):
  - Activity 1: Polarity
  - Activity 2: Conductivity
  - Activity 3: Series and Parallels Circuits
  - Activity 4: Sew E-Textile Project
- F2F1 Day 2 (2 hours):
  - Activity 5: Debug Exercise 1
  - Activity 6: Debug Exercise 2
  - Activity 7: Debug Exercise 3
- F2F1 Day 3 (4 hours)
  - Activity 8: Protosnap Program 1
  - Activity 9: Protosnap Program 2
  - Activity 10: Protosnap Program 3
  - Activity 11: Debug Exercise 4

    - Activity 12: Debug Exercise 5
    - Activity 13: Debug Exercise 6
- 4 to 5 Weeks
    - Project
- F2F2 Day 2 (2 hours)
    - Activity 15: Debug Exercise 7
    - Workshop time to work on project

Other considerations. Several other considerations regarding the learning module design are discussed below.

Learning module guidebook. A learning module guidebook is provided in Appendix A; the handbook contains materials lists, circuit layouts, design scenarios, and sample programs for the learning module activities. The purpose of the handbook is to assist someone conducting the workshop. Copies of the buggy and corrected programs were added to the second iteration of the learning module guidebook to be helpful to the workshop leader.

Design scenarios. The design scenarios document in Appendix J is intended to be used as a handout for the students working on the debug exercises. There is also a student handout for the introduction to programming in Appendix L. The design scenarios were written in a light-hearted tone, using the names of the course professors and teaching assistant. They could be used as they are, but they could also be personalized to the group participating in the learning module. The debug exercises were created on a variety of fabrics with a sea theme, because the Pepperdine University mascot is a wave. Since the group of students was of mixed gender, an attempt was made to choose fabric patterns that were not gender specific. A different theme could be chosen, or hand created designs could be used

Sample of correct behavior. As the students worked on the debug exercises during the first F2F, it became evident that there were still some questions about the desired end behavior for the projects. For Bug 7, a short video was created to demonstrate the end behavior, to address this problem. Demonstration videos would be helpful for the students for the earlier bugs, especially where the desired behavior is more complex. A point to be considered, however, is whether the videos will impact the students debugging process. With the Bug 7 video, it became evident that the students were using the sample videos for clues about the circuit stitching as well as the behavior.

Wiki. The Sakai platform that was used for the EDLT740 course had a built-in Wiki option. However, students needed to be prompted to contribute to the Wiki, and it was not highly utilized. On the Learning Module Evaluation, the agreement score for the question about the Wiki's usefulness was 3.6 out of 6 (with 1 representing disagreement and 6 representing agreement). One factor that may have contributed to the non-effectiveness of the Wiki was the difficulty in formatting the C code that was added to the Wiki. It was not possible to simply copy and paste code from the Arduino platform to the Wiki. Each line needed to be altered so that the code would be formatted in a readable way.

Student M0901, one of the experienced coders, created a 'Coding Crash Course' for his cadre mates early in the semester, but rather than use the Wiki, he shared it as a Google Doc. Later a link was added to the Wiki, but more as an afterthought. Another student, M0701, found and shared a web resource called Codebender, which had a useful method of sharing code.

For the Wiki to be a useful tool for sharing code, adding code snippets needs to be simplified. Perhaps an alternative Wiki site could be explored, or another technology could serve the purpose of collecting shared code and Arduino examples.

Project notebook. Students were told that a project notebook was required for the research study. Most students created their project notebook using the Sakai Blogging tool, but others used various other technologies, such as google docs and outside blogs. The students rated the project notebooks usefulness at an average of 3.2 of 6 (using the same 1 to 6 scoring), but for the course professor and researcher the project notebooks were a quick way to check in on project progress. Example entries from two blogs are shown in Figure 88 and Figure 89.

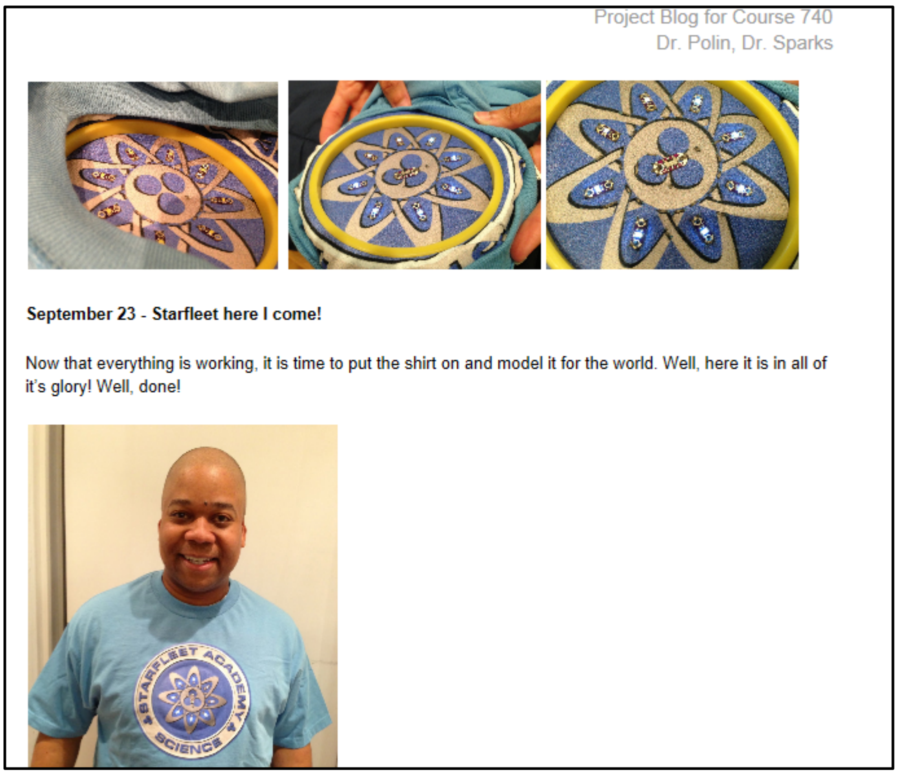
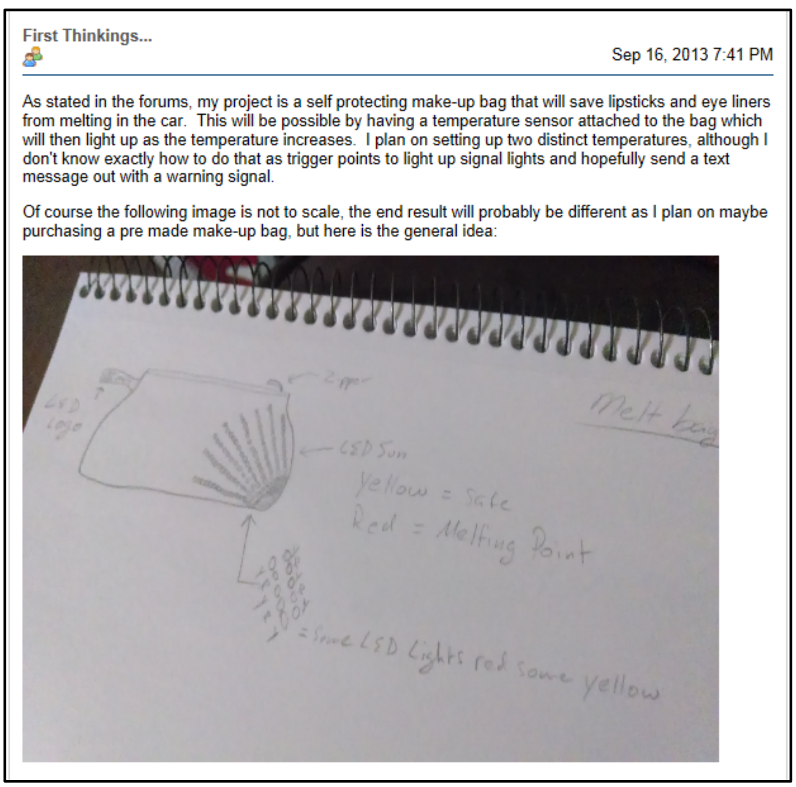


Figure 88. M0302's project blog.

Since the project notebooks were useful to the course professors, but not perceived to be very useful by the students, their use needs to be considered and perhaps adjusted in some way.

Other channels. Within the course the students had multiple ways to communicate with and support each other's project work. There were multiple forum discussions, some created by the Professors and Teacher Assistant, and some created by the students themselves. The students

could also choose to make their Project Notebooks (Blogs) readable and allow comments. During synchronous on-line course meetings, students spoke about their projects and challenges they were facing. Students posted their reflection papers in a forum, so they were able to read each other's reflections on the course experience. In addition, some students posted code and samples in the course Wiki. Interestingly, students not only used the in-course methods for communicating about their projects, but they used other channels as well. Student M0301 commented, "For our cadre, advice flowed not only in sync sessions and Sakai, but on Facebook, email, and Twitter."


First Thinkings...

Sep 16, 2013 7:41 PM

As stated in the forums, my project is a self protecting make-up bag that will save lipsticks and eye liners from melting in the car. This will be possible by having a temperature sensor attached to the bag which will then light up as the temperature increases. I plan on setting up two distinct temperatures, although I don't know exactly how to do that as trigger points to light up signal lights and hopefully send a text message out with a warning signal.

Of course the following image is not to scale, the end result will probably be different as I plan on maybe purchasing a pre made make-up bag, but here is the general idea:

Figure 89. F0501's project notebook.

Suggestions Regarding the Design of the Study

Because this is a design-based study, there are two foci. One focus is the learning module, which is the product that is being designed. The second focus is the design of the study

that is capturing data about the learning module. This section contains suggestions regarding the second focus – the design of the study.

Recordings. The data collected in this study was obtained from several sources. Livescribe pens recorded the pair work conversations as the students worked on the debugging exercises. A bit more distance was needed between the pairs as it was difficult to transcribe certain parts of the dialogues due to overlapping conversations.

Another aspect of the recordings was that the groups needed to be reminded about restarting the Livescribe recordings after any break. Some group work was not captured because the students forgot to restart the recordings.

Materials. Considerable time is required to make the sets of debugs for the groups. There is also quite a bit of expense required. As mentioned before, some of the measures taken in the first iteration to lower the cost of the materials actually affected the debug exercises. Using the smaller LilyTiny's for the three engineering bugs was less expensive, but because the signals were random it made it more difficult for the students to tell what effect their changes had. If LilyTiny's are used, they should be reprogrammed.

The use of the removable LilyPad for Bugs 4, 5, and 6 greatly increased the sewing time for the bugs, due to the number of snaps that needed to be sewn. If something similar is done, it is important that the snaps all be the same size so that the connections are not erratic. Other options would be to use the Protosnap board as mentioned before, purchase the pre-built removable LilyPad's, or purchase a separate LilyPad for each textile.

A different material, each with a sea theme, was selected for each of the six bugs. The different material was selected to add variety, but it also allowed the groups to easily compare their progress to the other groups. The sea theme was selected to give an underlying continuity

for the design scenarios. Bug 7 was created using a stencil and fabric paint, but it added additional construction time as compared to using pre-stamped material.

Because of the number of bugs and groups, it took quite some time to prepare all the e-textiles for the debugging exercises. The thread in some of the bugs seemed to have lost some of its conductivity by the time the bugs were used. Research into which conductive thread has the best longevity needs to be considered.

Participants. This study was conducted with non-computer science graduate students in education. The computer experience survey completed at the beginning of the study showed that six of the 17 students who responded to the survey had some programming experience, but even including these six students, the overall average agreement score for the statement “I learn new programming languages easily” was 2.6 out of 6 (1 represents complete disagreement, 6 represents complete agreement.) The students did have high agreement scores about the importance of computing and technology, as discussed in Chapter 4, in the section called ‘Overall attitudes toward computing, electronics, e-textiles, and technology.’ These non-computer science graduate students were able to debug e-textile projects and create complex Arduino projects that expressed their interests. As graduate students in Learning Technology, these participants were able to verbalize their debugging strategies as they worked on the debugging exercises and reflect meta-cognitively on how their personal learning experiences fit with the learning theories that they were studying. This level of insight and reflection is probably not representative of a general group of students. The diverse projects created by the students does show the breadth of the possibilities that e-textiles and traditional Arduino technology can provide.

Self-efficacy. One of the desired outcomes for this study was increased computational self-efficacy and confidence. The data analysis in Chapter 4 shows that there were both increases and decreases in self-efficacy, and that there were many expressions of lack of self-confidence as well as expressions of confidence. This points to a need for further research and care in the design of the learning module with regard to students' self-efficacy. Students indicated that more up-front instruction would make them feel more comfortable with the material and that feedback has been incorporated into the modifications for the next iteration of the learning module. In addition, direct instruction about the dispositions that support computational thinking, described by Barr and Stephenson (2011), could be helpful.

Versions. During the transcribing of the debug exercises there were times that it would have been helpful to see the code the students were working on, at the time they were speaking. The only code available, however, was the group's final code for the bug. This problem could be addressed by having the groups save the programs with different version numbers as they made changes.

Gathering survey data. Students were invited to complete six quick surveys over the course of the semester. Five of the surveys collected data using Qualtrics online survey software, but students responded to the Learning Module Evaluation Questionnaire (LME) on an anonymous paper survey completed at the end of the second F2F meeting. The online surveys took 2 to 5 minutes each to complete, and the LME took 5 to 7 minutes to complete. Invitations to complete the online surveys were emailed to the students, and follow-up reminders were also sent, but the response rates were disappointing (Computer Experience: 94%, Computer Attitudes, completed both pre and post: 67%, Self-efficacy, completed both pre and post: 44%). The surveys were administered separately to make them quick, easy, and convenient to respond to,

but the response rates might have been greater if the surveys that were conducted at the same time were combined. The LME questionnaire that was collected in person had the highest response rate. Since the LME survey was designed to be an anonymous survey, one of the students collected the completed surveys. All students completed the LME.

E-Textile or Arduino. Some students in the study designed e-textile projects and others chose to design more traditional Arduino projects. Both types of projects elicited computational thinking, so it seems that allowing the students a choice of either type of project allows them the best opportunity to use the computational thinking perspective of expressing.

Implications for the Study of Computational Thinking

Previous sections of this chapter have explored the implications that the findings of this study have for the development of the next iteration of the learning module and also provided suggestions for future research studies based upon this study's design. This section will consider the wider implications that this study has for the study of computational thinking.

Current definition of computational thinking. As stated in the literature review in Chapter 2, there is still discussion and disagreement about the definition of Computational Thinking. This study was based on the Brennan & Resnick (2012) definition that considers computational thinking in terms of concepts, practices, and perspectives. The data gathered in this study's debugging exercise transcripts, group debrief, forum posts, and reflection papers supports the computational thinking concepts, practices, and perspectives identified by Brennan & Resnick (2012). Through the process of data analysis in this study, additional computational thinking concepts, practices, and perspectives have emerged:

Concepts. The concepts that emerged were illustrated in Chapter 4, and are related to engineering principles used with e-Textiles. They are:

- Power, Positive & Negative
- Physical setup determines code requirements
- Control LED lights with high/low signals
- Crossed threads cause short circuits
- If connected, same polarity
- Independent behavior needs different pins
- Need delay to see LED behavior

Practices. The practices that emerged, also illustrated in Chapter 4, are:

- Document
- Keep Code Clean

The Brennan and Resnick (2012) computational thinking practice of Testing and Debugging was further refined to have the following sub-practices:

- Benchmark
- Use a debugging strategy
- Establish circuit correctness
- Predict expected behavior
- Refer to specifications
- Use the visibility of the behavior
- Verify correct end
- Walk through code
- Versions

Perspectives. The perspectives that emerged, also illustrated in Chapter 4, are:

- Adapt to Others' Code

- Many Ways to Solve
- Failure is just a step forward

Having emerging elements does not invalidate Brennan and Resnick's (2012) definition of computational thinking; rather, it strengthens the conceptual premise that computational thinking is composed of concepts, practices, and perspectives. Considering the gathered data through the lens of Brennan and Resnick's (2012) definition helped identify elements that might otherwise have been overlooked.

Activity type influences demonstrated computational thinking. Different types of activities evoke different types of computational thinking. As the students in this study participated in the debugging exercises, they did not discuss the individual computational concepts much, but really engaged in computational thinking practices. This is likely due to the fact that in the debugging exercises they were given pre-written programs to correct rather having to create a program from scratch. Working on their project programs required students to match computational concepts with project needs, but this work was not captured in the study.

The debugging exercises were intended to be an easy entry into coding, since they would allow students to manipulate existing code to see what it did. Different exercises could be created that would bring out computational thinking concepts. An example could be a matching game where students could choose among existing code chunks to accomplish given tasks. A study looking to examine particular aspects of computational thinking needs to be designed with activities that are likely to elicit those aspects.

Appeal to non-computer science majors. Although the students in the study were not computer science majors, they were able to make sophisticated Arduino projects that expressed their interests and concerns. Several students expressed the desire to continue learning more

about making with Arduino and e-textiles. This confirms that e-textiles and Arduinos are able to engage a diverse group of people and can lead them to experiences with computational thinking.

Suggestions for Future Research

In the course of conducting this study, opportunities for future research have become apparent. Some of these are discussed below.

Case study, focusing on self-efficacy. As the pair interactions during the debug exercises were transcribed it became very noticeable that certain students were experiencing great anxiety with regard to their ability to debug and program. These students also seemed to be hesitant to ask questions, perhaps because they did not want to expose their lack of understanding. It would be enlightening to do a case study of one of these students to understand what methods could be used to allow this type of student to learn and experiment with computation without the anxiety.

Assessment. Assessment of computational thinking is an area that requires further research. Many people and organizations are advocating bringing more emphasis on computational thinking into the public school system. This will bring an even greater need for non-complex ways to measure computational thinking. Constructing meaningful projects allows the development of computational thinking concepts, practices, and perspectives, but teachers need an effective and time efficient way to assess their development.

During the debug exercises certain computational thinking practices and perspectives seemed to be associated with those who were more experienced with programming. For example, the perspective of being able to adapt to others' code seemed to be more evident among those students who had more programming experience. The perspectives that there are many ways to solve a problem and that failure is just a step forward also seemed to be more evident in

the more advanced students. Examples of practices that seemed to be more advanced were benchmarking and being iterative. Further study could indicate if there is a hierarchy among the use of the computational thinking concepts, practices, and perspectives. If a hierarchy can be established, then it could be used a tool in assessment.

Additional CT practices. There were some indications in the transcriptions of the coding challenges of a debugging practice that might be described as pinching in. This pinching in is with reference to what the students were given to start with, and what the end goal of the exercise was. The students would start with what they were given and work through the program for a while, and then they would switch to considering what the end goal of the program was and what type of code was needed to get them there. Then, they would take that perspective and go back to look at the code they were given again. They would do this switching of debugging direction several times as they narrowed in on the coding bug. This is analogous to the student who used the alligator clips and an LED to walk down the circuit line from both directions to find the area where there was a break. It is also reminiscent of a practice sometimes used to write geometrical proofs. Further study of this computational thinking practice could provide insight into the debugging process.

Summary

This chapter discussed implications of the results of the design-based study that developed an e-textile learning module to increase computational thinking among graduate education students. The discussion touched upon recommended changes for the next iteration of the learning module, considerations that relate to the design of the study, and implications that have wider application to the field of computational thinking. Suggestions for future research were also offered.

Results of the study show that students engaged in computational thinking as they worked with their partners to complete the debugging exercises. Their experiences with the exercises led to insight about their personal Arduino projects. The debugging exercises were seen to have an impact on student confidence and self-efficacy; the traumatic experiences of several students lend emphasis to the need to give proper support for the students as they become familiar with computing.

During the analysis of the pair work, new computational thinking concepts, practices, and perspectives emerged. Examples of these emergent codes were illustrated as well as the original concepts, practices, and perspectives in Brennan & Resnick (2012).

The study helps contradict the shrinking pipeline metaphor, showing that it is possible to encourage computational thinking and interest in computation with university-aged students and not just middle-school ones. In addition, the study provided evidence that even a relatively short e-textile module could change attitudes and engage students in computational thinking.

As the results and implications of this study are considered, it is important to remember that the e-textile debugging module was part of an applied educational seminar whose larger purpose was to provide context for understanding constructionism, maker culture, and communities of practice. The students' creation of a personal Arduino project was an essential part of their learning experience.

REFERENCES

Aho, A. (2012). Computation and computational thinking. *The Computer Journal, 55*(7), 832-835. doi:10.1093/comjnl/bxs074

Bandura, A. (1977). *Social Learning Theory.* Englewood Cliffs, NJ: Prentice Hall. doi:10.4135/9781412959193.n17

Barab, S. & Squire, K. (2004). Design-based research: Putting a stake in the ground. *The Journal of the Learning Sciences, 13*(1), 1-14. doi:10.1207/s15327809jls1301_1

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48-54. doi:10.1145/1929887.1929905

Beckwith, L., Kissinger, C., Burnett, M., Wiedenbeck, S., Lawrance, J., Blackwell, A. & Cook, C. (2006). Tinkering and gender in end-user programmers' debugging. *Proceedings of the ACM Conference on Human Factors in Computing Systems* (pp. 231-240). New York, NY: ACM Press. doi:10.1145/1124772.1124808

Beede, D., Julian, T., Langdon, D., McKittrick, G., Khan, B. & Doms, M. (2011). *Women in STEM: A gender gap to innovation*. Washington, DC: U.S. Department of Commerce. doi:10.2139/ssrn.1964782

Blackwell, A., Church, L., & Green, T. (2008). The abstract is 'an enemy': Alternative perspectives to computational thinking. Proceedings PPIG'08, 20th Annual Workshop of the Psychology of Programming Interest Group, 34-43. Retrieved from https://www.researchgate.net/publication/242727102_The_Abstract_is_'an_Enemy'_Alternative_Perspectives_to_Computational_Thinking

Brennan, K., & Resnick, M. (2012). *Using artifact-based interviews to study the development of computation thinking in interactive media design.* Paper presented at the 2012 Annual Meeting of the American Educational Research Association (Vancouver, BC, Canada,).

Bruner, J. S. (1990). *Acts of meaning*. Cambridge, MA: Harvard University Press. doi:10.1017/s0033291700030555

Buechley, L. (2010). *LilyPad Arduino: How an open source hardware kit is sparking new engineering and design communities.* Retrieved from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.168.5521&rep=rep1&type=pdf doi:10.1.1.168.5521

Buechley, L., Eisenberg, M., Catchen, J., & Crockett, A. (2008). The LilyPad Arduino: Using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 423-432). doi:10.1145/1357054.1357123

Camp, T. (1997). The incredible shrinking pipeline. *Communications of the ACM, 40*(10), 103–110. doi:10.1145/543812.543846

Cech, E., Rubineau, B., Silbey, S., & Seron, C. (2011). Professional role confidence and gendered persistence in engineering, *American Sociological Review, 76*(5), 641-666. doi:10/1177/0003122411420815

Cox, T. H., & Blake, S. (1991). Managing cultural diversity: Implications for organizational competitiveness. *Academy of Management Perspectives*, *5*(3), 45-56. doi:10.5465/ame.1991.4274465

Eagly, A. H., & Carli, L. L. (2007). Women and the labyrinth of leadership. *Harvard Business Review, 85*(9), 62. doi:10.1108/hrmid.2008.04416aad.004

Einhorn, S. (2011). MicroWorlds, computational thinking, and 21st century learning. *LCSI*. Retrieved from: http://www.microworlds.com/support/files/lcsi-computational-thinking.pdf

Fields, D., Kafai, Y., Searle, K. A., & Min, H. S. (2012, March). *Debugg'ems to assess student learning in e-textiles*. Poster presented at the annual meeting of the ACM Special Interest Group on Computer Science Education (SIGCSE), Raleigh, NC. doi:10.1145/2157136.2157367

Fisher, A., Margolis, J., & Miller, F. (1997). Undergraduate women in computer science: Experience, motivation and culture. *ACM SIGCSE Bulletin, 29(*1), 106-110. doi:10.1145/268085.268127

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher, 42*(1), 38-43. doi:10.3102/0013189x12463051

Hamner, E., Lauwers, T., Bernstein, D., Nourbakhsh, I., & DiSalvo, C. (2008). Robot diaries: Broadening participation in the computer science pipeline through social technical exploration. *Proceedings of the AAAI Spring Symposium on Using AI to Motivate Greater Participation in Computer Science, 2008*(1)*,* 48-43.

Hill, C., Corbett, C., & St. Rose, A. (2010). *Why so few? Women in science, technology, engineering, mathematics*. Washington, DC: American Association of University Women.

Jesse, J. (2006). The poverty of the pipeline metaphor: The AAAS/CPST study of nontraditional pathways into IT/CS education and the workforce. In J. M. Cohoon & W. Aspray (Eds.), *Women and Information Technology*, (pp. 239-278). doi:10.7551/mitpress/9780262033459.003.008

Johansson, F. (2006). *The Medici effect: What elephants & epidemics can teach us about innovation.* Boston, MA: Harvard Business School Press.

Kafai, Y. B., & Resnick, M. (1996). Introduction. In Y. B. Kafai & M. Resnick (Eds.), *Constructionism in Practice: Designing, thinking, and learning in a digital world,* (pp. 1-8). Mahwah, NJ: Lawrence Erlbaum Associates. doi:10.4324/9780203053492

Kramer, J. (2007). Is abstraction the key to computing? *Communications of the ACM, 50*(4), 37-42. doi:10/1145/1232743.1232745

Kwahk, K.Y., & Ahn, H. (2010). Computer Self-Efficacy Measure—Revised. Retrieved from *PsycTESTS.* doi:10.1037/t13162-000.

Lacey, T. A., & Wright, B. (2009). Occupational employment projections to 2018. *Monthly Labor Review*, *132*(11), 82–123.

Lapidot, T., & Hazzan, O. (2005). Song debugging: Merging content and pedagogy. *Inroads, The SIGCSE Bulleting*, *37*(4), 79–83. doi:10.1145/1113847.1113882

Lave, J. & Wenger, E. (1991). *Situated learning: Legitimate peripheral participation*. New York, NY: Cambridge University Press. doi:10.1017/cbo9780511815355.003

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., . . . Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads, 2*(1), 32 -37. doi:10.1145/1929887.1929902

Lu, J., & Fletcher, G. (2009). Thinking about computational thinking. In *Proceedings of the 40th ACM technical symposium on Computer Science Education (SIGCSE'09),* (pp. 260-264). Chattanooga, TN: ACM. doi:10.1145/1508865.1508959

Margolis, J., Fisher, A., & Miller, F. (1999-2000). Caring about connections: Gender and computing. *IEEE Technology and Society Magazine, 18*(4), 13-20. doi:10.1109/44.808844

Margolis, J., & Fisher, A. (2006). Geek mythology and attracting undergraduate women to computer science. *Women in Engineering ProActive Network.* doi:10.1177/0270467602239766

McLuhan, M. (1964). *Understanding media: The extensions of man*. New York, NY: McGraw Hill. doi:10.2307/2711172

National Science Board. (2012). *Science and Engineering Indicators 2012*. Arlington, VA: National Science Foundation (NSF).

National Science Foundation (2013). *Awards Simple Search*. Retrieved from: http://www.nsf.gov/awardsearch/simpleSearchResult?queryText=e-textile&ActiveAwards=true

Page, S. E. (2008). *The difference: How the power of diversity creates better groups, firms, schools, and societies*. Princeton, NJ: Princeton University Press. doi:10.1017/s1537592708082054

Papert, S. (1980). *Mindstorms. Children, computers and powerful ideas.* New York, NY: Basic Books. doi:10.1007/978-3-0348-5357-6

Papert, S. (1993). *Mindstorms: Children, Computers, and Powerful Ideas*, New York, NY: Harper Collins.

Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism*, 1-11. Norwood, NJ: Ablex Publishing. doi:10.1.1.691.4506

Pearl, A., Pollack, M. E., Riskin, E., Thomas, B., Wolf, E., & Wu, A. (2002). Becoming a computer scientist. *ACM SIGCSE Bulletin, 34*(2), 135-143. doi:10.1145/543812.543847

Phillips, P. (2008). Computational thinking: A problem-solving tool for every classroom. *CSTA & Microsoft*. Retrieved from: http://www/csta.acm.org/resources/sub/resourcefiles/computationalthinking.pdf

Polin, L. (2012). *Developing and assessing computational thinking: There's an app for that.* Culver City, CA: Pepperdine University IT Grant.

Resnick, M. (2007). Sowing the seeds for a more creative society. *Learning and Leading with Technology, 35*(4), 18. doi:10.1145/1518701.2167142

Repenning, A., Webb, D., & Ioannidou, A. (2010). Scalable game design and the development of a checklist for getting computational thinking into public schools. *Proceedings of the 41st ACM technical symposium on Computer Science education*, 265-269. doi:10.1145/1734263.1734357

Siegel, E. V. (1999). Why do fools fall into infinite loops: Singing to your computer science class. *ACM SIGCSE Bulletin*, *31*(3), 167-170. doi:10.1145/384267.305909

Simard, D. (n.d.). What if women were equally represented in innovation? *Anita Borg Institute for Women and Technology, News.* Retrieved from: http://anitaborg.org/news/archive/what-if-women-were-equally-represented-in-innovation/

Thom, M. (2001). *Balancing the equation: Where are the women and girls in science, engineering and technology?* New York, NY: The National Council for Research on Women.

Tyler-Wood, T., Knezek, G., & Christensen, R. (2010). Instruments for assessing interest in STEM content and careers. *Journal of Technology and Teacher Education, 18*(2), 341-363.

Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*. Cambridge, MA: Harvard University Press. doi:10.2307/1421493

Wang, F., & Hannafin, M. J. (2005). Design-based research and technology-enhanced learning environments. *Educational Technology Research & Development, 53*(4), 5-23. doi:10.1007/BF02504682

Wilensky, U. (1991). Abstract meditations on the concrete and concrete implications for mathematics education. *Constructionism.* 193-203. Norwood, NJ: Ablex Publishing.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35. doi:10.1145/1118178.1118215

Zweben, S. (2012). Computing degree and enrollment trends. Computing Research Association, Preliminary results of the 2010-2011 Taulbee Survey. Retrieved from https://cra.org/govaffairs/wp-content/uploads/sites/6/2013/03/CRA_ Taulbee_CS_Degrees_and_Enrollment_2011-12.pdf

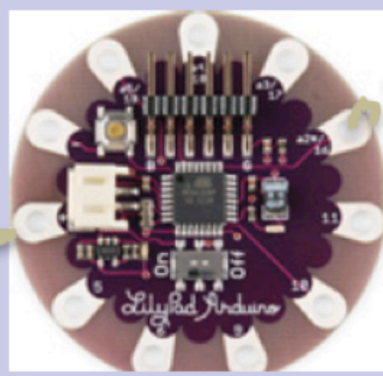
APPENDIX A

Learning Module Guidebook

Computational Thinking

Through

E-Textiles



Learning Module Guide

Activity 1: Polarity

Learning Goal: Batteries have a positive and negative terminal. LED's have a positive and negative side.

Tools & Materials:
Coin Cell Battery
Double ended Alligator Clips, Red and Black
LED with curled ends
LilyPad LED
Two conductive magnets
Circuit Diagram

Vocabulary:

Electricity:	Needs a power supply and a closed circuit to flow. Flows from negative and positive.
Circuit:	A closed loop that allows electricity to flow.
Current:	Flow of electricity.
Polarity:	The condition of being positive or negative.
Short Circuit:	Faulty or accidental connection between two points of different potential in an electrical circuit.

Description:

Use alligator clips to connect battery to LED. Use a conductive magnet on each side of the magnet to attach alligator clip to battery. Use red alligator clip to connect positive of battery and LED. Use black alligator clip to connect negative of battery and LED.

Try with standard LED.

Try with LilyPad LED.

Try switching positive and negative of standard LED.

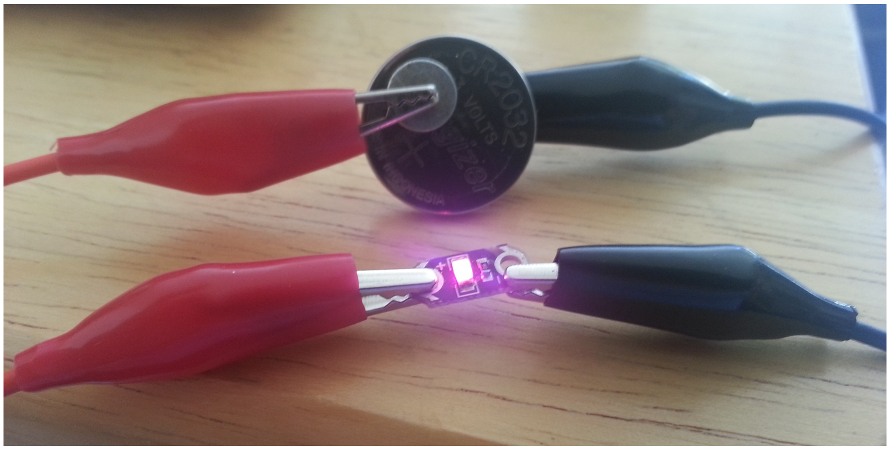


Figure A90. Alligator clip circuit lighting an LED.

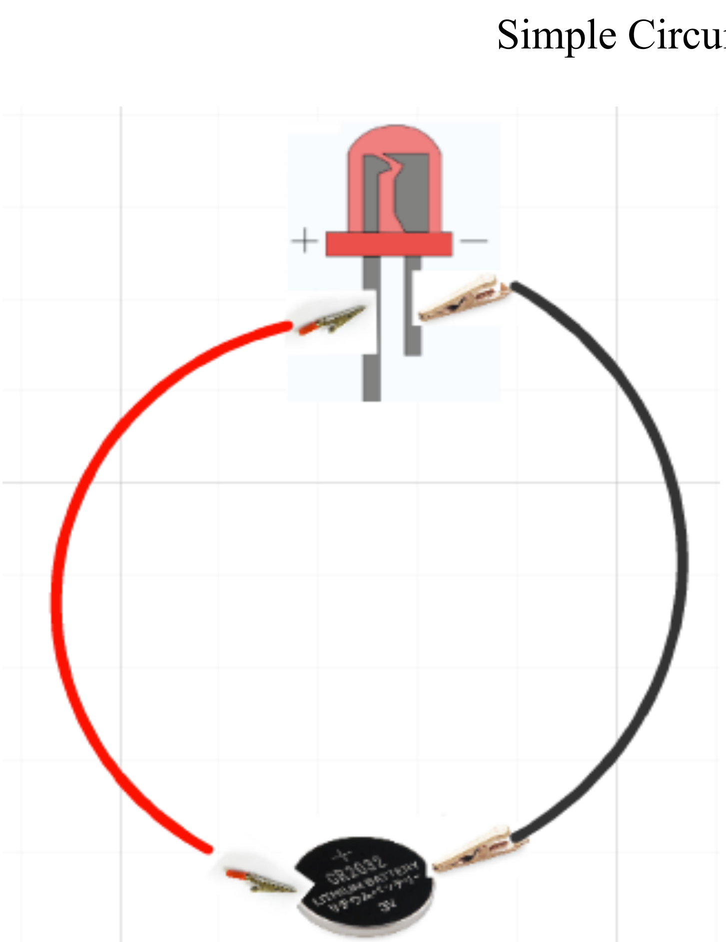


Figure A91. Schematic for alligator clip circuit.

Activity 2: Conductivity

Learning Goal: Some objects conduct electricity; others do not.

Tools & Materials:
Coin Cell Battery
Double-ended alligator clips, Red x 1, and Black x 2
Standard LED with curled ends
Two conductive magnets
Circuit Diagram
Samples to test conductivity

Vocabulary:
Conductive: Allows electricity to flow

Description:

Create a simple circuit as in Activity 1. Add an extra black alligator clip. Try different materials between two black alligator clips. If the LED lights up, the material is conductive.

Circuit to Test Conductivity

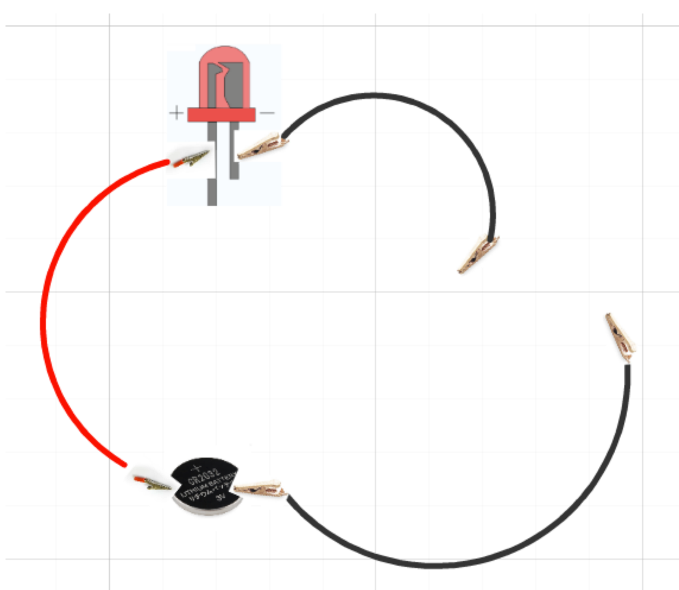


Figure A92. Alligator circuit to test conductivity.

Activity 3: Series and Parallel Circuits

Learning Goal: There are two basic types of circuits, series and parallel. Parallel circuits are most common in e-textiles.

Tools & Materials:
Coin Cell Battery
6 Double-ended alligator clips, Red x 3 and Black x 3
3 LilyPad LED s
Two conductive magnets
Circuit Diagrams

Vocabulary:
Parallel Circuit
Series Circuit

Description:

Create a parallel circuit. Use red alligator clips to connect all positive ends of the LED's to the positive of the battery. Use black alligator clips to all the negative ends of the LED's to the negative of the battery. Verify that all LED's light. Switch one LED. No LED's light.

Parallel Circuit

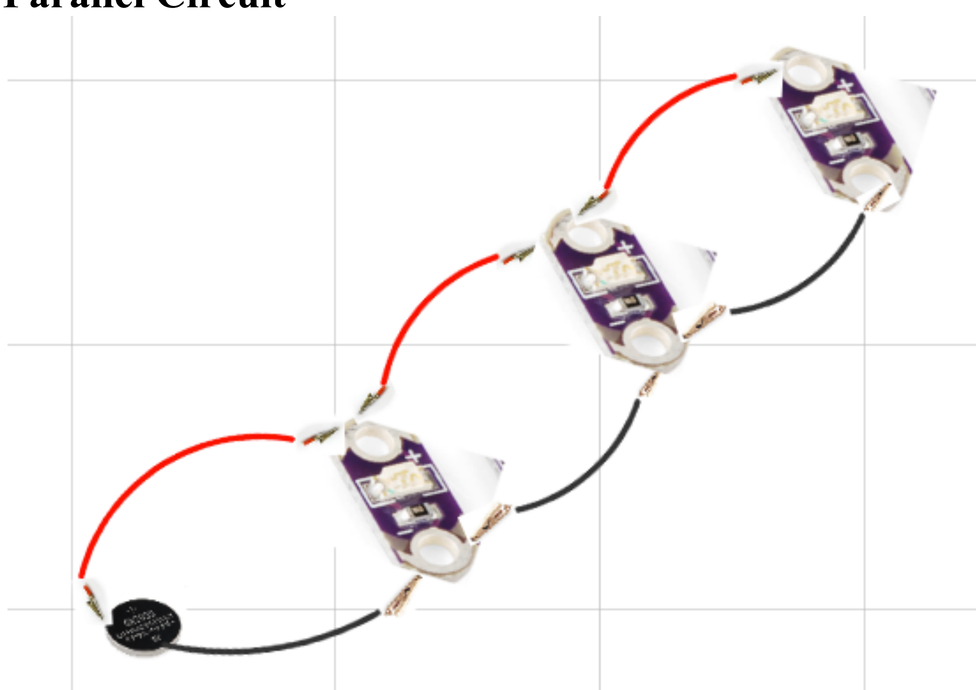


Figure A93. Alligator clip example of parallel circuit.

Activity 4: Sew E-textile Project

Learning Goal: To create a working e-textile project.

Tools & Materials:
Protosnap Sewing Kit (Fabric patch with diagram, conductive thread, battery holder, battery, LED's, button, switch, needles).
Circuit Diagrams

Vocabulary:
Knots
Running Stitch
Needle Threader
Bobbin

Notes:
The circuits are parallel. The direction of the LED's matter!

Description:
Examine the diagram on the fabric patch. Trace the circuits. Identify why it they are parallel circuits.

Separate the parts in the Protosnap sewing kit. Use a running stitch to create the LED's and battery. Make sure that the LED's are oriented correctly. Also, make sure that the threads on the back do not touch or a short circuit will be created. Knots can be secured using glue.

Test to see the effects of the switch and the button.

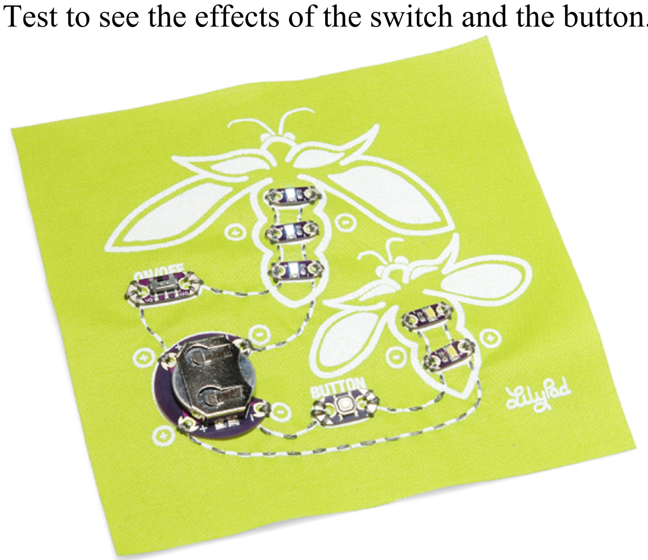


Figure A94. E-textile project included in LilyPad kit.

Activity 5: Debug 1

Learning Goal: Locate and correct short circuit.

Tools & Materials:
Debug 1 project, one for each pair
Design Scenario 1
Conductive thread
Coin cell battery
Scissors
Needle
Needle threader
Tool for Circuit Diagrams (paper & pencil or Prezi)

Vocabulary:
Bug
Short Circuit

Notes:
Have participants work in pairs
(Permission to record,
Record pair collaboration,
Circulate and interview teams) for research
If using LilyTiny, make sure that the participants know that the signals are random.
Take photo of bug fix.

Description:
Participants will work to find and correct the bug in the project.

Design Scenario:
For the tropical drinks e-textile, Vicky decides that she wants the LED at Point A to have a certain behavior, the LED's at Points B & C to behave the same as each other, but different from the LED at Point A, and the LED's at Points D & E to behave the same as each other, but different from all the others.

Please locate and correct Vicky's bug. You may need to cut and re-sew some of the soft circuits. You can draw on the diagrams, if you wish.

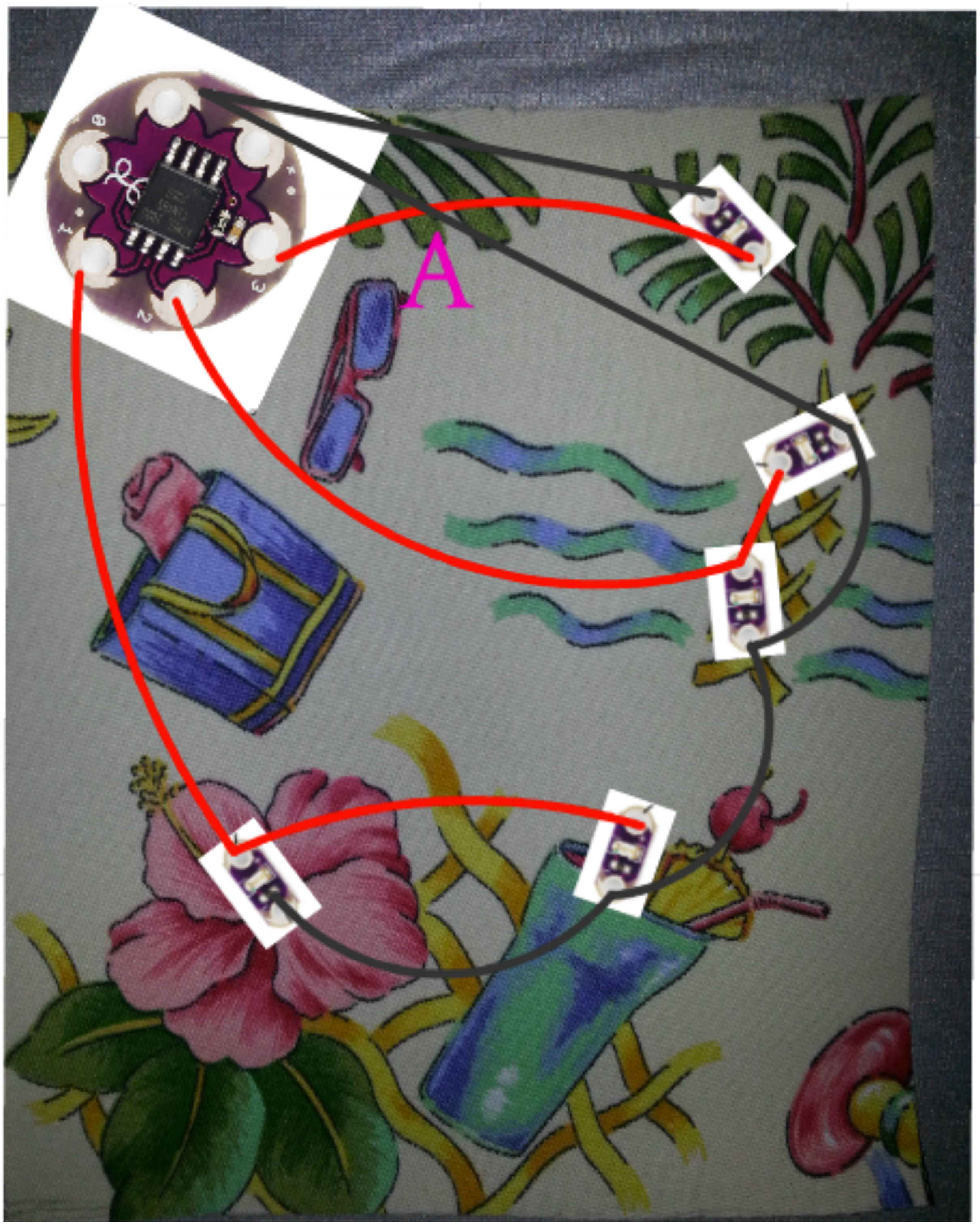


Figure A95. Debug activity 1, short circuit error.

Possible correction for Bug 1.

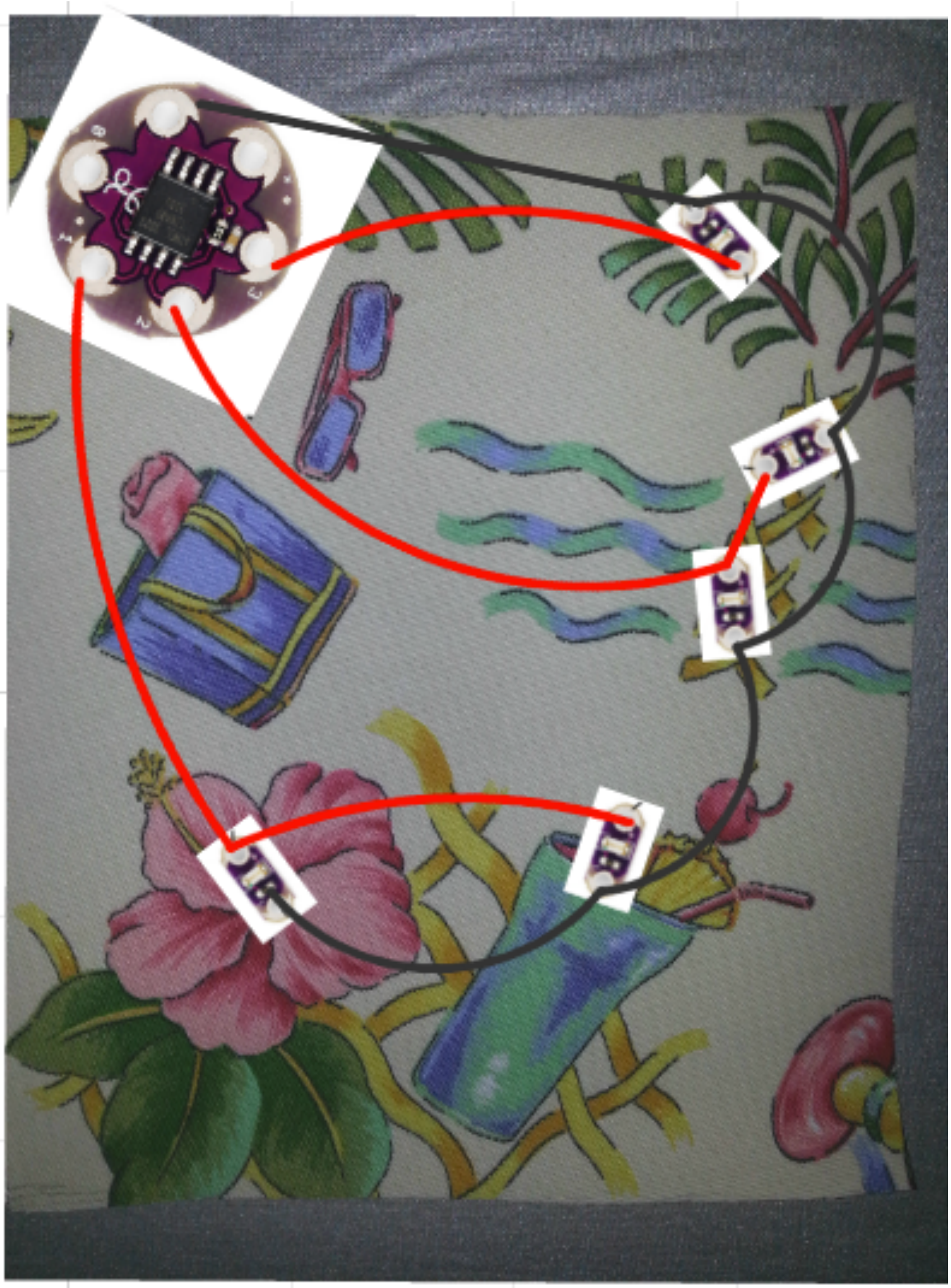


Figure A96. Possible correction for Bug 1.

Activity 6: Debug 2

Learning Goal: Recognize circuits, make independent circuits for LED's requiring independent actions.

Tools & Materials:
Debug 2, one for each pair
Design Scenario 2
Conductive thread
Coin cell battery
Scissors
Needle
Needle threader
Tool for Circuit Diagrams (paper & pencil or Prezi)

Notes:
Participants work in pairs.
Take photo of bug fix.

Description:
Participants will work to find and correct the bug in the project.

Design Scenario:
Vicky's next e-textile project is a colorful flip-flops scene. Vicky has decided that she wants the LED at Point A to have a certain behavior, the LED's at Points B & C to behave the same as each other, but different from the LED at Point A, and the LED's at Points D & E to behave the same as each other, but different from all the others.

Please locate and correct Vicky's bug. You may need to cut and re-sew some of the soft circuits. You can draw on the diagrams, if you wish.

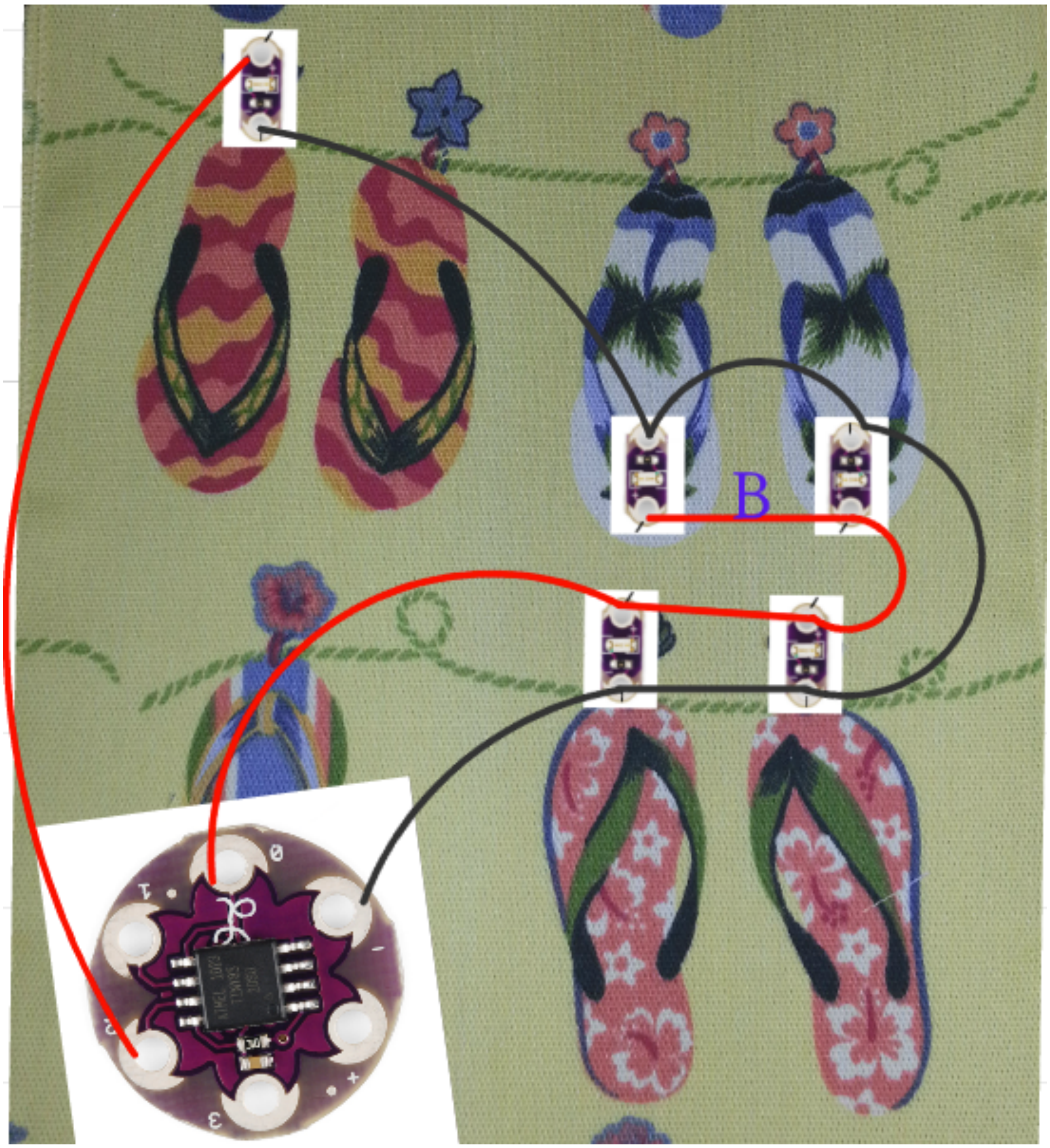


Figure A97. Debug activity 2, electronic topology error.

Possible Correction for Bug 2.

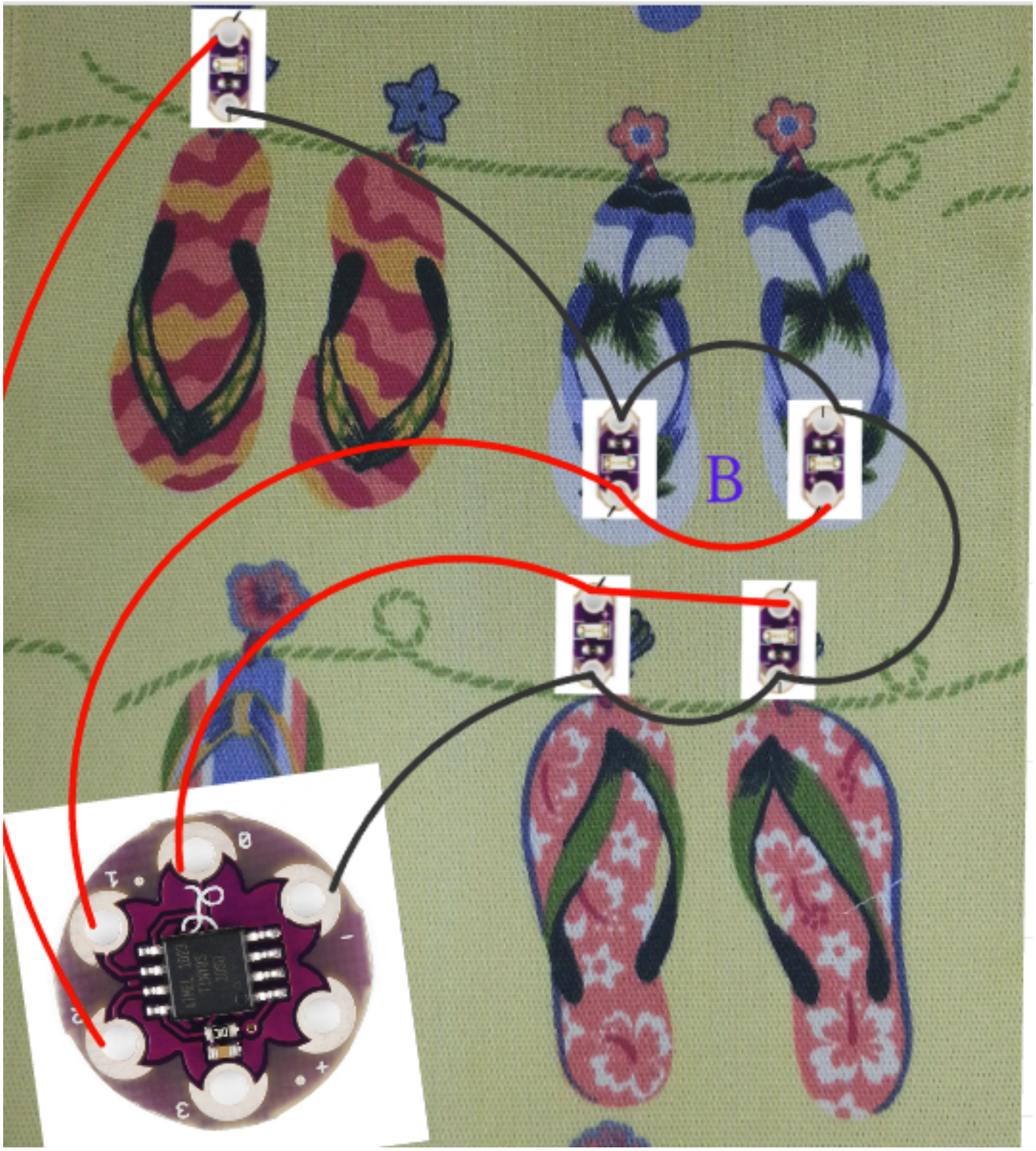


Figure A98. Possible correction for Bug 2.

Activity 7: Debug 3

Learning Goal: Have code and circuitry correctly use the polarity of each LED.

Tools & Materials:
Debug 3, one for each pair
Design Scenario 3
Conductive thread
Coin cell battery
Scissors
Needle
Needle threader
Tool for Circuit Diagrams (paper & pencil or prezi)

Vocabulary:
Polarity

Notes:
Participants work in pairs.
Take photo of bug fix.

Description:
Participants will work to find and correct the bug in the project.

Design Scenario:
Vicky's third e-textile is a blue sailboat. She wants the LED's on the sail, the LED's at Points A, B, and C, to behave the same way, and the LED at Point D to behave independently.

Vicky's wall hanging is not lighting up as she wants. Please locate and correct her bug. You may need to cut and re-sew some of the soft circuits. You can draw on the diagrams, if you wish.

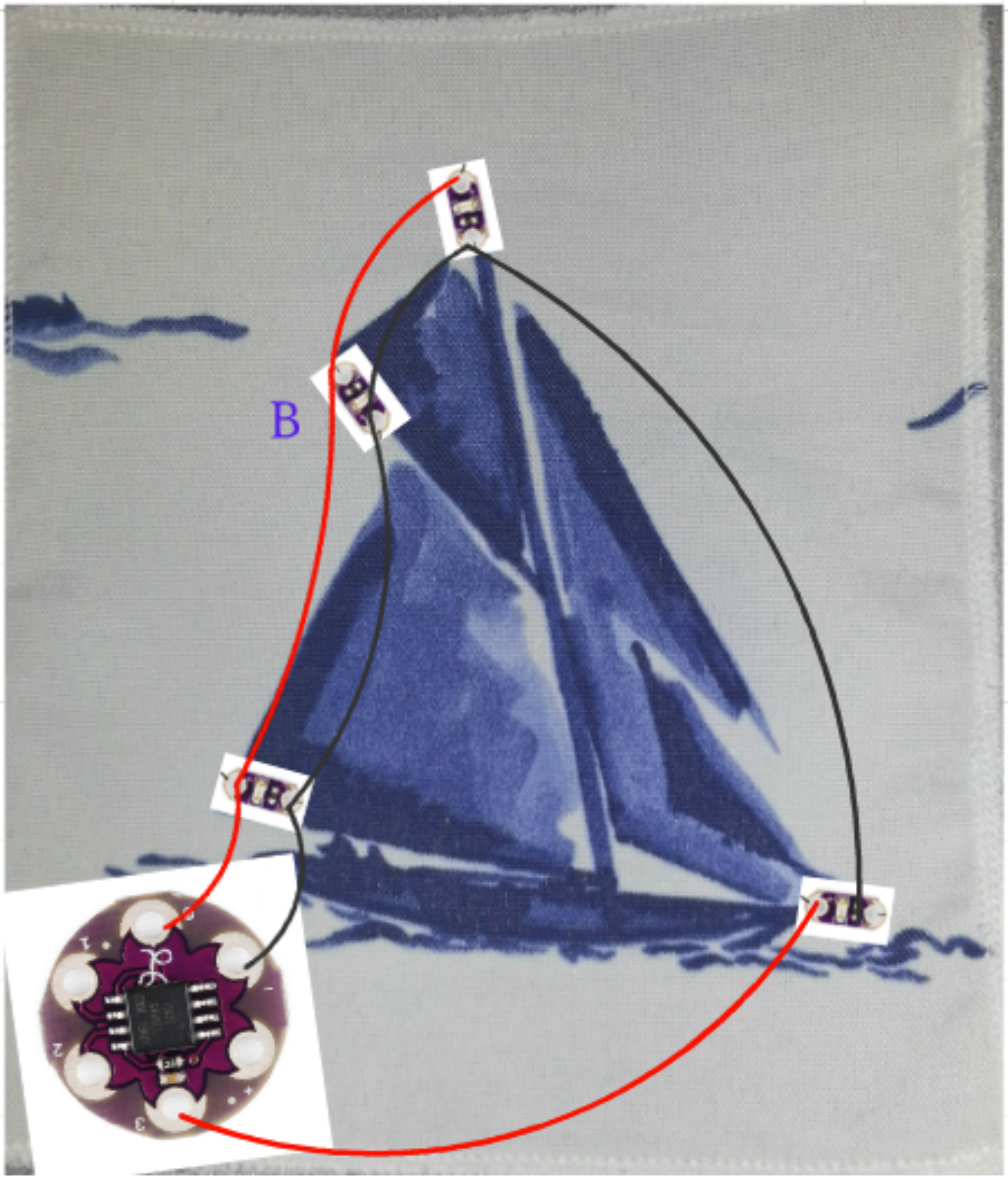


Figure A99. Debug Activity 3, polarization error.

Possible correction for Bug 3.

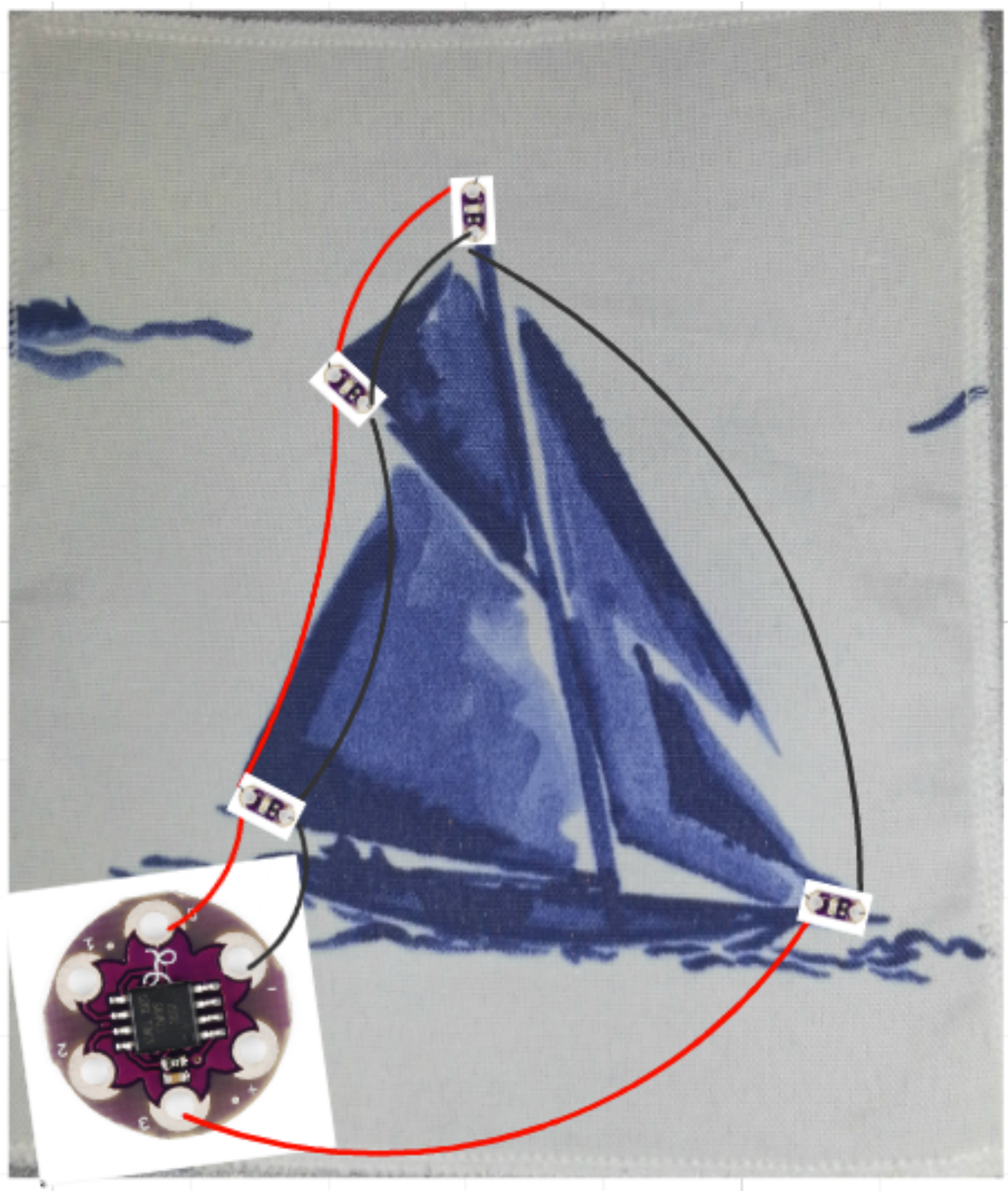


Figure A100. Possible correction for Bug 3.

Activity 9: Protosnap Program 1

Learning Goals: Experience the Arduino programming environment. Upload programs. Run programs. Learn how to place comments in code. Learn how to declare input or output type. Explore Arduino sample programs.

Tools & Materials:
Protosnap Kit (Protosnap board, FTDI connector, USB mini cord, Computer, Arduino software)

Notes:
Demonstrate uploading programs
Experiment with inputs and outputs
Set Up
    LED's, High and Low
Show the sample programs

Description:
Begin with the Blink Program
Alternate Blinking LEDs

Figure A101. Tools for programming the LilyPad Protosnap board.

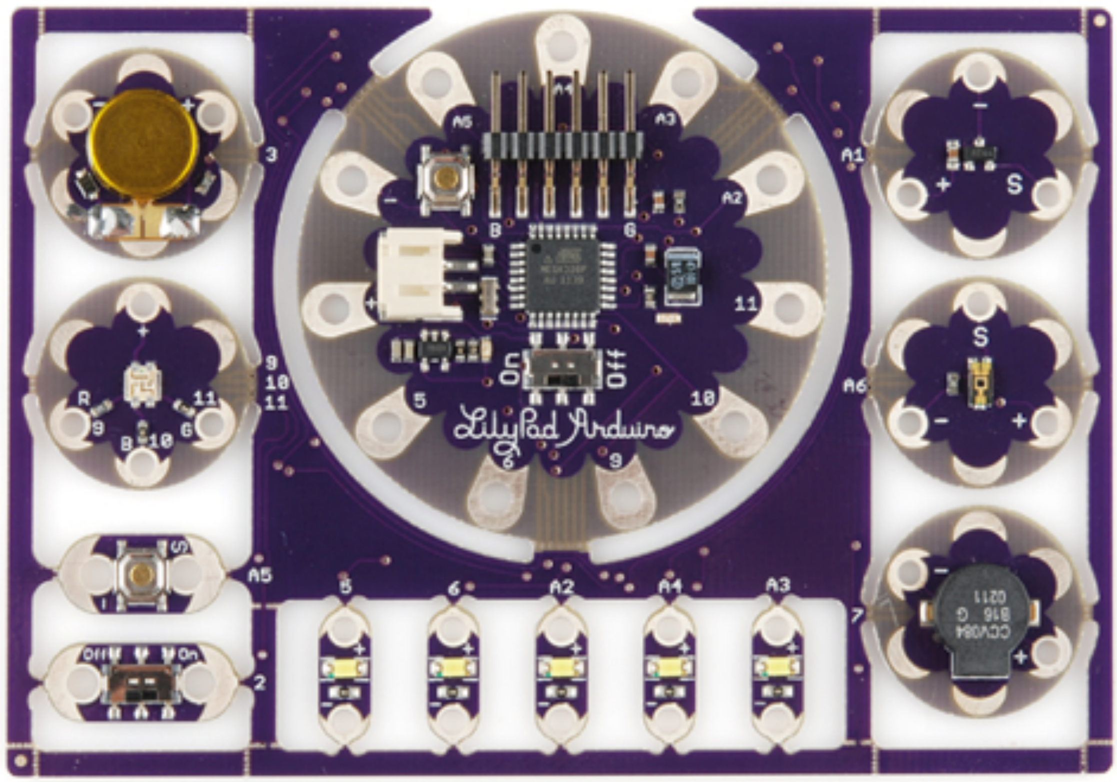


Figure A102. Close-up of the LilyPad Protosnap board.

The code in the Blink program below is an Arduino example, in the Basics folder. There are two ways to make comments. For one-line comments you can use //. For longer comments enclose the block between /* and */

```
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.
This example code is in the public domain.
*/
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;
// the setup routine runs once when you press reset:
void setup() {
// initialize the digital pin as an output.
pinMode(led, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(1000);               // wait for a second
digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
delay(1000);               // wait for a second
}
```

Figure A103. Code for Protosnap Program 1.

The code above in Blink makes the LED ON the LilyPad blink. We will change the code to make the LED attached to pin 5 blink and make the blink twice as fast. Changes are highlighted in yellow.

```
/*
Blink –Altered

Turns on an LED on for one second, then off for one second, repeatedly.
This example code is in the public domain.
*/
// Blink LED 5
//
int led = 5;
// the setup routine runs once when you press reset:
void setup() {
// initialize the digital pin as an output.
pinMode(led, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(1000/2);             // wait for half a second
digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
delay(1000/2);             // wait for half a second
}
```

Figure A104. Blink altered.

We will change the code to make the LEDs attached to pin 5 and 6 alternate blinks. Changes are highlighted in yellow.

```
/*
 Blink –Altered-2
 Turns on an LED on for for half a second, then off for half a second, repeatedly.
 LED6 alternates with LED5.
 */
// Blink LEDs 5 & 6
//
int led5 = 5;
int led6 = 6;
// the setup routine runs once when you press reset:
void setup() {
// initialize the digital pins as an output.
pinMode(led5, OUTPUT);
pinMode(led6, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
digitalWrite(led5, HIGH);    // turn the LED on (HIGH is the voltage level)
digitalWrite(led6, LOW);     // turn the LED off (LOW is the voltage level)
delay(1000/2);               // wait for half a second
digitalWrite(led5, LOW);     // turn the LED off by making the voltage LOW
digitalWrite(led6, HIGH);    // turn the LED on by making the voltage HIGH
delay(1000/2);               // wait for half a second
}
```

Figure A105. Blink altered 2.

Open the Fade Example in the Arduino-Examples-Basics. Change the LED to 5.

```
/*
  Fade

  This example shows how to fade an LED on pin 5
  using the analogWrite() function.

  This example code is in the public domain.
  */

int led = 5;          // the pin that the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup()  {
  // declare pin 5 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop()  {
  // set the brightness of pin:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount ;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

Figure A106. Fade altered.

Activity 10: Protosnap Program 2

Learning Goal: Understand the behavior of a conditional statement in the program. Add conditional to a program to produce desired behavior. Understand the behavior of a loop in the program. Alter the behavior of the loop.

Tools & Materials:
Protosnap Kit (Protosnap board, FTDI connector, USB mini cord, Computer, Arduino software)

Notes:
If statement
Loops

Description:

Alter program. Insert IF statement, Loop

Figure A107. Tools for programming the LilyPad Protosnap board.

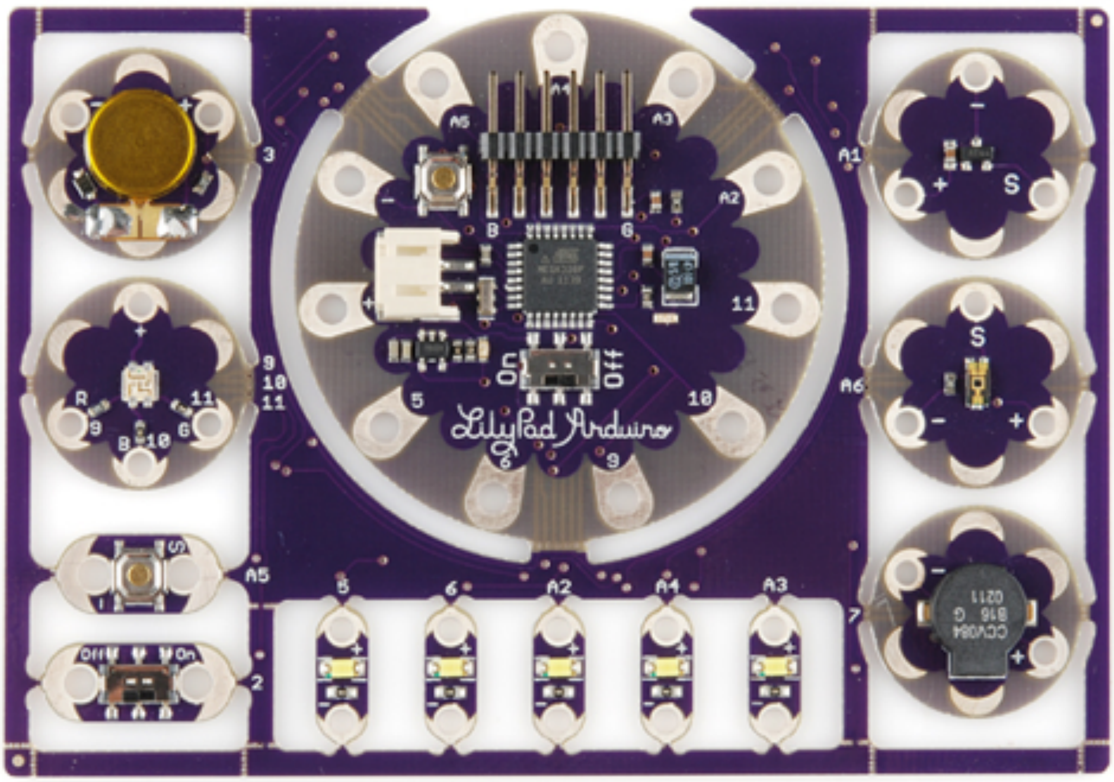


Figure A108. Close-up of the LilyPad Protosnap board.

Run Conditional Example. Note the use of == in the if statement.

```
/*
Conditional Example
If it is the first time through the loop, turn on LED 5,
otherwise turn on LED6.
*/
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int ledB = 13;
int led5 = 5;
int led6 = 6;
int startCount;
// the setup routine runs once when you press reset:
void setup() {
// initialize the digital pin as an output.
pinMode(ledB, OUTPUT);
pinMode(led5, OUTPUT);
pinMode(led6, OUTPUT);
startCount = 0;
}
// the loop routine runs over and over again forever:
void loop() {
if (startCount == 0)
{
digitalWrite(led5,HIGH);
}
else
{
digitalWrite(led6,HIGH);
}
digitalWrite(ledB, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(1000);            // wait for a second
digitalWrite(ledB, LOW);    // turn the LED off by making the voltage LOW
digitalWrite(led5,LOW);
digitalWrite(led6,LOW);
delay(1000);            // wait for a second
startCount = startCount + 1;
}
```

Figure A109. Conditional example.

Activity 11: Protosnap Program 3

Learning Goal: Use the serial port to display program progress.

Tools & Materials:
Protosnap Kit (Protosnap board, FTDI connector, USB mini cord, Computer, Arduino software)

Notes:
Serial Output

Description:

Alter loop program to show serial output

Figure A110. Tools for programming the LilyPad Protosnap board.

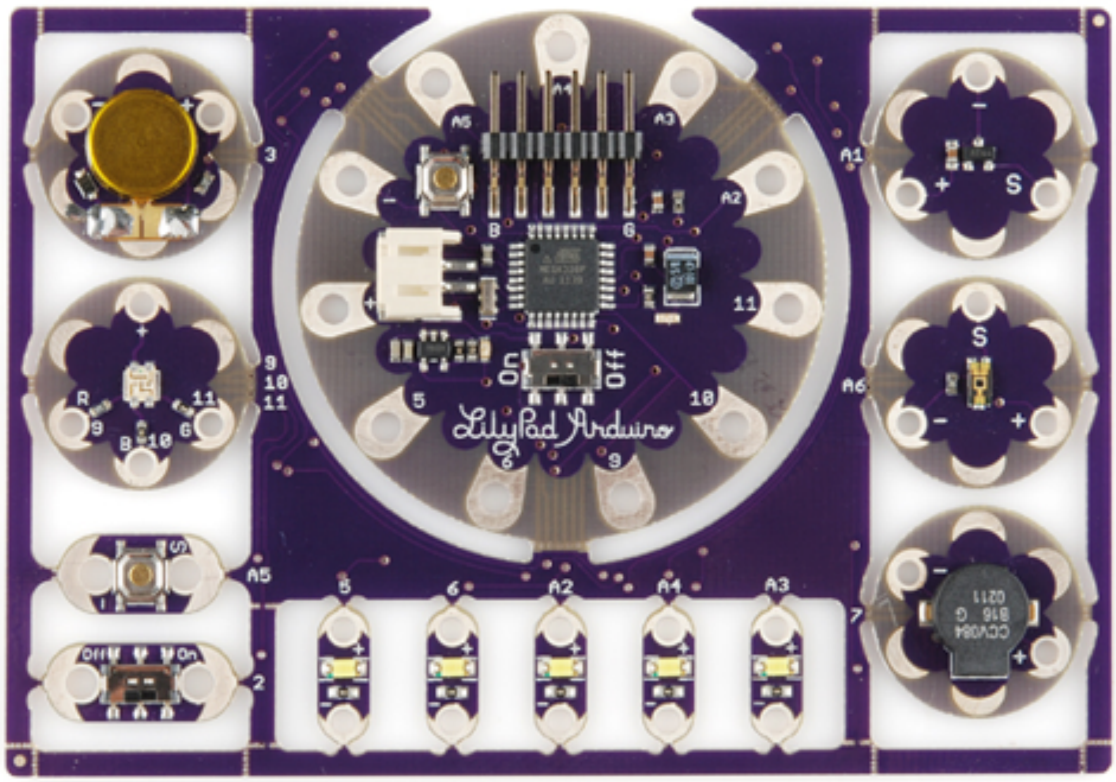


Figure A111. Close-up of the LilyPad Protosnap board.

We will change the Conditional example to output to the monitor. Save as ConditionalSerial. Note Serial.print vs. Serial.println. Changes are highlighted in yellow.

```
/*
Conditional Serial Example
If it is the first time through the loop, turn on LED 5,
otherwise turn on LED6.
Display output on serial monitor
*/
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int ledB = 13;
int led5 = 5;
int led6 = 6;
int startCount;
 // the setup routine runs once when you press reset:
void setup() {
// initialize the digital pins as an output.
pinMode(ledB, OUTPUT);
pinMode(led5, OUTPUT);
pinMode(led6, OUTPUT);
Serial.begin(9600);      // prepare to use serial monitor
startCount = 0;
}
// the loop routine runs over and over again forever:
void loop() {
Serial.print("startCount is  "); // note this is print not println
Serial.print(startCount);
if (startCount == 0)
{
digitalWrite(led5,HIGH);
Serial.println(", Blink Led 5");
}
else
{
digitalWrite(led6,HIGH);
Serial.println(", Blink Led 6");
}
digitalWrite(ledB, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(1000);             // wait for a second
digitalWrite(ledB, LOW);    // turn the LED off by making the voltage LOW
digitalWrite(led5,LOW);
digitalWrite(led6,LOW);
delay(1000);             // wait for a second
startCount = startCount + 1;
}
```

Figure A112. Sample serial print.

Activity 12: Debug 4

Learning Goal: Find and correct the code error so that LED's function correctly. Code Challenge: Constant vs. Variable Pin on LilyPad.

Tools & Materials:
Bug 4, one for each pair
Design Scenario 4
Program for Bug4
LilyPad Simple
Connectors for LilyPad
Scissors
Thread
Needles
Tool for Circuit Diagrams (paper & pencil or Prezi)
Computer with Arduino environment installed
FTDI and USB cable

Notes:
Be sure students are able to connect the LilyPad processor to the debug project correctly.

Description:
The first programming challenge bug.

Design Scenario:
Vicky's fourth wall-hanging is more sophisticated. She has used a computer to control the behavior of the LED's. Vicky wants to start with the spiked shell, and turn shell lights on and off, individually, in a counter clockwise direction.

The program is not working correctly, however. Please locate and correct Vicky's bug. You can draw on the diagrams, if you wish.

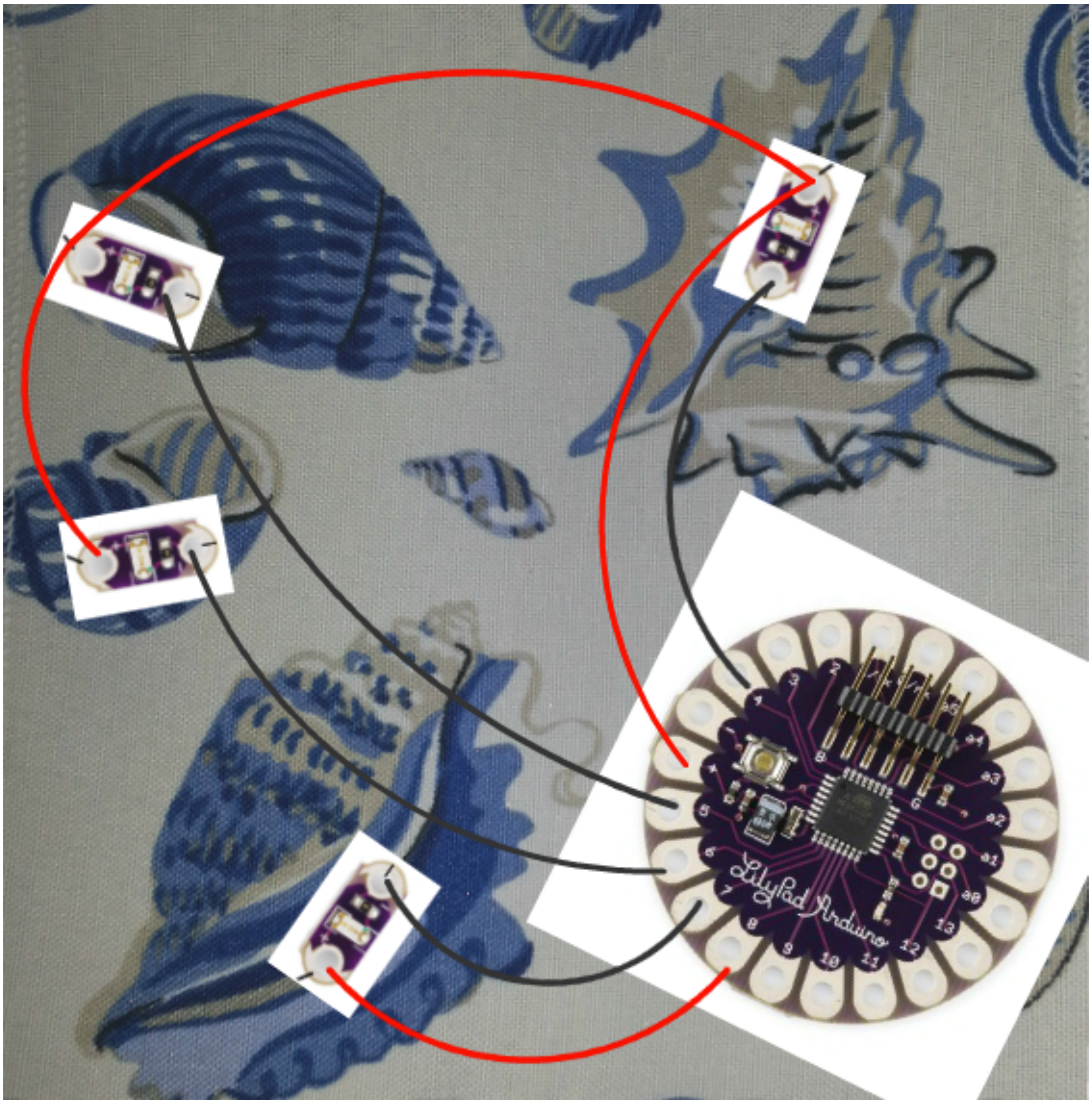


Figure A113. Debug activity 4, variable pin error.

```
/*
 BUG 4, Constant versus Variable Pins   BUGGY
 Victoria Kim Dissertation

 Desired Action: Starting with spiked shell, turn shell lights on and off,
 individually, in a counter clockwise direction.
 */

// Name the led pins on the Arduino.
int led19 = 19 ;   //   spikey shell
int led5 = 5 ;     //   smooth shell
int led6 = 6 ;     //   little shell
int led9 = 9 ;     //   rough shell
int led10 = 10 ;   //   rough shell 2nd conn
int waitTime = 1000;

// setup runs once
void setup() {
  // initialize the digital pins as an output.
  pinMode(led19, OUTPUT);
  pinMode(led5, OUTPUT);
  pinMode(led6, OUTPUT);
  pinMode(led9, OUTPUT);
  pinMode(led10, OUTPUT);
  digitalWrite(led10, LOW);    //set pin to negative

  digitalWrite(led19, HIGH);   //start with pin off
  digitalWrite(led5, HIGH);    //start with pin off
  digitalWrite(led6, HIGH);    //start with pin off
  digitalWrite(led9, HIGH);    //start with pin off

}

// Loop routine (continually repeats)
void loop() {
  digitalWrite(led19, LOW);   // turn spikey on
  delay(2*waitTime);          // wait 2 second
  digitalWrite(led19, HIGH);  // turn spikey off
  delay(2*waitTime);          // wait 2 second

  digitalWrite(led5, LOW);    // turn smoothy on
  delay(2*waitTime);          // wait 2 second
  digitalWrite(led5, HIGH);   // turn smoothy off
  delay(2*waitTime);          // wait 2 second

  digitalWrite(led6, LOW);    // turn baby on
  delay(2*waitTime);          // wait 2 second
  digitalWrite(led6, HIGH);   // turn baby off
```

```
    delay(2*waitTime);          // wait 2 second

    digitalWrite(led9, LOW);    // turn roughy on
    delay(2*waitTime);          // wait 2 second
    digitalWrite(led9, HIGH);   // turn roughy off
    delay(2*waitTime);          // wait 2 second

}

```

```
 /*
 BUG 4 CORRECTED, Constant versus Variable Pins
 Victoria Kim Dissertation

 Desired Action: Starting with spiked shell, turn shell lights on and off,
 individually, in a counter clockwise direction.
 */

// Name the led pins on the Arduino.
int led19 = 19 ;   //   spikey shell
int led5 = 5 ;     //   smooth shell
int led6 = 6 ;     //   little shell
int led9 = 9 ;     //   rough shell
int led10 = 10;     // create a negative pin
int waitTime = 1000;

// setup runs once
void setup() {
  // initialize the digital pins as an output.
  pinMode(led19, OUTPUT);
  pinMode(led5, OUTPUT);
  pinMode(led6, OUTPUT);
  pinMode(led9, OUTPUT);
  pinMode(led10, OUTPUT);

  digitalWrite(led10, HIGH);   //set pin to positive   ** Correction

  digitalWrite(led19, HIGH);   //start with pin off
  digitalWrite(led5, HIGH);    //start with pin off
  digitalWrite(led6, HIGH);    //start with pin off
  digitalWrite(led9, HIGH);    //start with pin off

}

// Loop routine (continually repeats)
void loop() {
  digitalWrite(led19, LOW);   // turn spikey on
  delay(2*waitTime);          // wait 2 second
  digitalWrite(led19, HIGH);  // turn spikey off
  delay(2*waitTime);          // wait 2 second

  digitalWrite(led5, LOW);    // turn smoothy on
  delay(2*waitTime);          // wait 2 second
  digitalWrite(led5, HIGH);   // turn smoothy off
  delay(2*waitTime);          // wait 2 second

  digitalWrite(led6, LOW);    // turn baby on
  delay(2*waitTime);          // wait 2 second
```

```
digitalWrite(led6, HIGH);   // turn baby off
delay(2*waitTime);          // wait 2 second

digitalWrite(led9, LOW);    // turn roughy on
delay(2*waitTime);          // wait 2 second
digitalWrite(led9, HIGH);   // turn roughy off
delay(2*waitTime);          // wait 2 second

}
```

Activity 13: Debug 5

Learning Goal: Find and correct the code error so that LED's function correctly. Code challenge: control flow.

Tools & Materials:
Bug 5, one for each pair
Design Scenario 5
Program for Bug 5
LilyPad Simple
Connectors for LilyPad
Scissors
Thread
Tool for Circuit Diagrams (paper & pencil or Prezi)
Needles
Computer with Arduino environment installed
FTDI and USB cable

Notes:
Be sure students are able to connect the LilyPad processor to the debug project correctly.
Explain fmod function.
Explain !=

Description:
The second programming challenge.

Design Scenario:
Vicky's fifth wall-hanging is also the more sophisticated kind. She has again used the computer to control the behavior of her LED's. Vicky wants the bottom flag (not on boat) to stay on, the middle flag (not on boat) to blink quickly, and the top flag (not on boat) to blink slowly. Vicky wants the light on the mast to blink twice quickly, then pause, then repeat.

The program is not working correctly, however. Please locate and correct Vicky's bug. You can draw on the diagrams, if you wish.

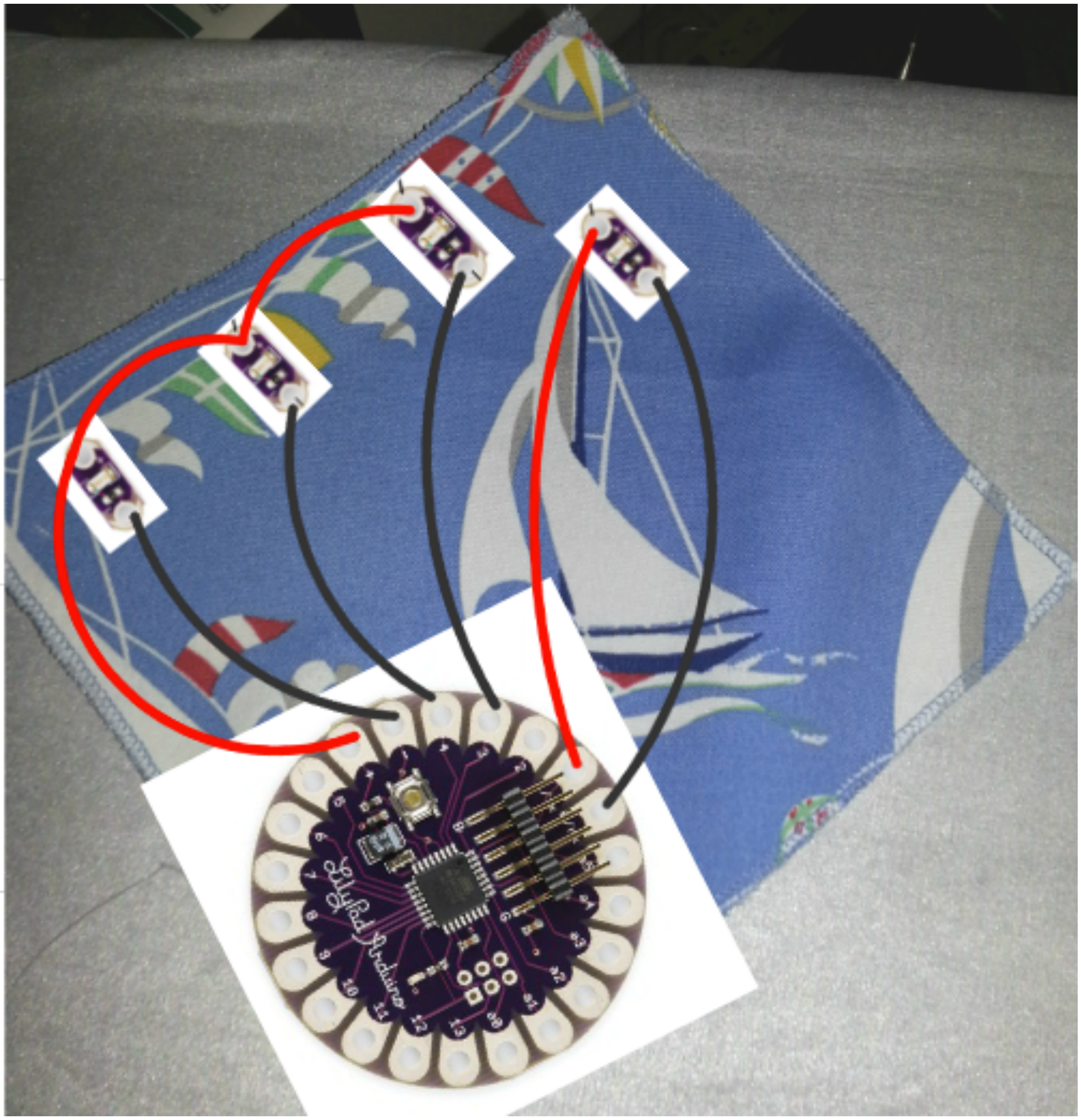


Figure A114. Debug activity 5, control flow error.

```
/*
BUG 5, Control Flow                BUGGY
Victoria Kim Dissertation

Desired Actions: Bottom flag (not on boat) stays on,
             Middle flag (not on boat) blinks quickly,
             Top flag (not on boat) blinks slowly,
             Light on mast blinks twice quickly, then pauses, then repeats.
*/

// Name the led pins on the Arduino.
int led11 = 11 ;
int led16 = 16 ;
int led17 = 17 ;
int led19 = 19 ;
int waitTime = 1000;
int cycleCnt = 0;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pins as output.
  pinMode(led11, OUTPUT);  //mast
  pinMode(led16, OUTPUT);  //mast
  pinMode(led17, OUTPUT);  //top
  pinMode(led19, OUTPUT);  //middle

  digitalWrite(led17, LOW);   // turn the LED on
  digitalWrite(led19, LOW);   // turn the LED on
  digitalWrite(led11, LOW);   // turn the LED on

  delay(waitTime/2);         // wait 1/2 second

}

// loop routine
void loop() {

  digitalWrite(led19, HIGH);  // turn the LED off
  digitalWrite(led11, HIGH);  // turn the LED off

  // led17 is on slower cycle
  // fmod divides by the second number and looks at the remainder
  if (fmod(cycleCnt,2) == 0)
  {
    digitalWrite(led17, HIGH);   // turn the LED off
```

```
}

delay(waitTime/2);

digitalWrite(led19, LOW);  // turn the LED on


// slower blink
if (fmod(cycleCnt,2) == 0)
{
digitalWrite(led17, LOW);   // turn the LED on
}

//Every third time, the mast LED will not come on
if (fmod(cycleCnt,3) != 0)
{
  digitalWrite(led11, LOW);
}

delay(waitTime/2);

cycleCnt = cycleCnt + 1;


}
```

```
/*
 BUG 5, Control Flow                          CORRECTED
 Victoria Kim Dissertation

 Desired Actions: Bottom flag (not on boat) stays on,
                  Middle flag (not on boat) blinks quickly,
                  Top flag (not on boat) blinks slowly,
                  Light on mast blinks twice quickly, then pauses, then repeats.
 */

// Name the led pins on the Arduino.
int led11 = 11 ;
int led16 = 16 ;
int led17 = 17 ;
int led19 = 19 ;
int waitTime = 1000;
int cycleCnt = 0;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pins as output.
  pinMode(led11, OUTPUT);  //mast
  pinMode(led16, OUTPUT);  //mast
  pinMode(led17, OUTPUT);  //top
  pinMode(led19, OUTPUT);  //middle
  digitalWrite(led16, HIGH);        //                    **  Corrected

  digitalWrite(led17, LOW);   // turn the LED on
  digitalWrite(led19, LOW);   // turn the LED on
  digitalWrite(led11, LOW);   // turn the LED on

  delay(waitTime/2);         // wait 1/2 second

}

// loop routine
void loop() {

  digitalWrite(led19, HIGH);  // turn the LED off
  digitalWrite(led11, HIGH);  // turn the LED off

  // led17 is on slower cycle
  // fmod divides by the second number and looks at the remainder
  if (fmod(cycleCnt,2) == 0)
  {
    digitalWrite(led17, HIGH);   // turn the LED off
  }

```

```
delay(waitTime/2);

digitalWrite(led19, LOW);  // turn the LED on

// slower blink
if (fmod(cycleCnt,2) == 0)
{
digitalWrite(led17, LOW);   // turn the LED on
}

//Every third time, the mast LED will not come on
if (fmod(cycleCnt,3) != 0)
{
  digitalWrite(led11, LOW);
}

delay(waitTime/2);

cycleCnt = cycleCnt + 1;
 }
```

Activity 14: Debug 6

Learning Goal: Find and correct the code error so that LED's function correctly. Code challenge: end state definition.

Tools & Materials:
Bug 6, one for each pair
Design Scenario 6
Program for Bug 6
LilyPad Simple
Connectors for LilyPad
Scissors
Thread
Needles
Tool for Circuit Diagrams (paper & pencil or Prezi)
Computer with Arduino environment installed
FTDI and USB cable

Notes:
Be sure students are able to connect the LilyPad processor to the debug project correctly.

Description:
The third programming challenge.

Design Scenario:
Vicky's final wall-hanging is the sophisticated type again, and she has used a computer to control the behavior of the LED's. Vicky wants all the corners to light together, flashing on and off.

The program is not working correctly, however. Please locate and correct Vicky's bug. You can draw on the diagrams, if you wish.

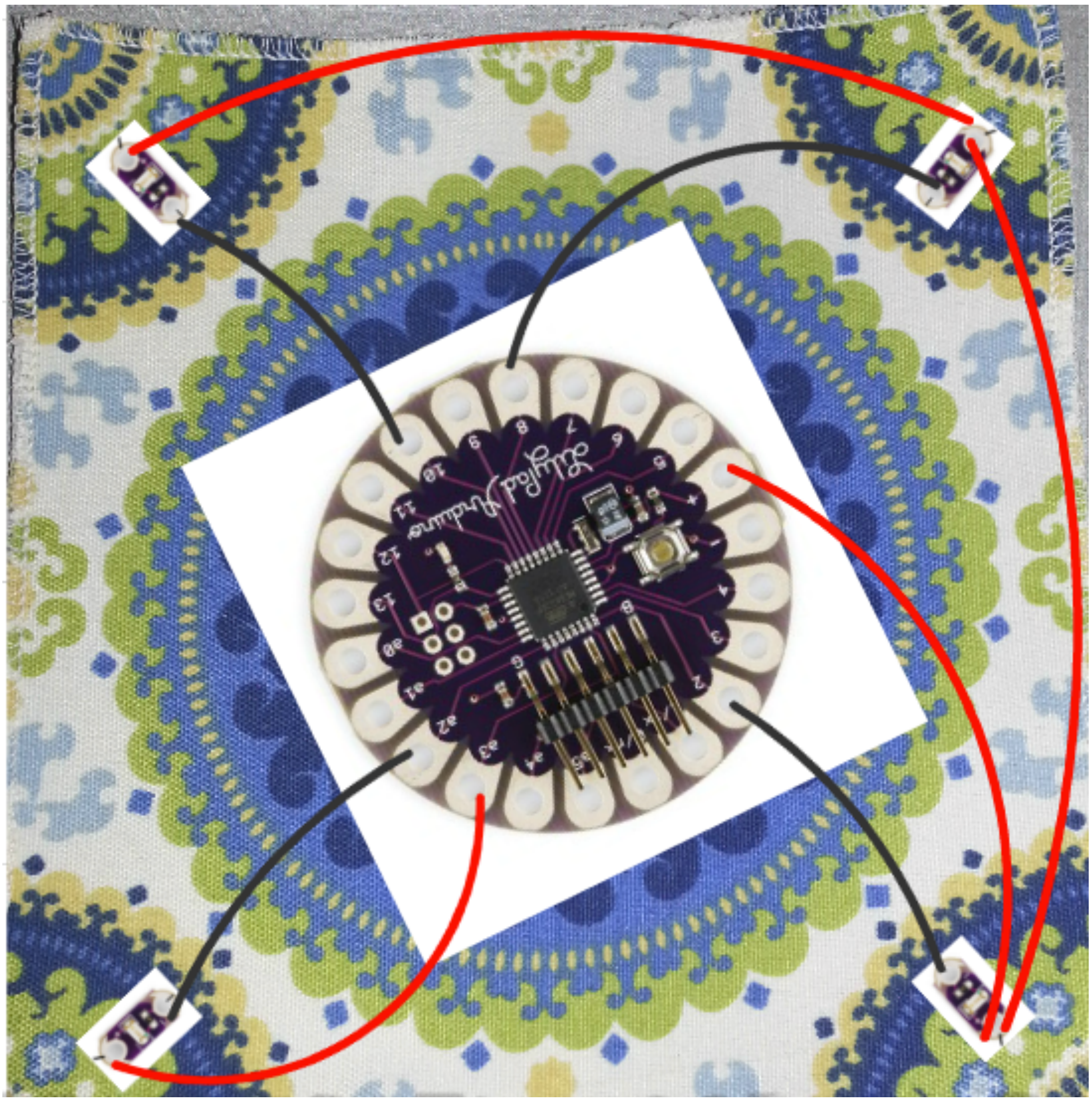


Figure A115. Debug activity 6, end state error.

```
 /*
 BUG 6, End State Definition,                BUGGY
 Victoria Kim Dissertation

 Desired behavior:  All corners light together, flash on and off
 */

// Name the led pins on the Arduino.
int led6 = 6 ;
int led10 = 10 ;
int led17 = 17 ;
int led16 = 16 ;
int led19 = 19 ;
int waitTime = 1000;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pins as output.
  pinMode(led6, OUTPUT);
  pinMode(led10, OUTPUT);
  pinMode(led17, OUTPUT);
  pinMode(led16, OUTPUT);
  pinMode(led19, OUTPUT);

  digitalWrite(led16, HIGH);   // Set as positive, Keep pin positive
}

// loop routine
void loop() {
  digitalWrite(led10, LOW);   // turn the LED on
  digitalWrite(led19, LOW);   // turn the LED on
  digitalWrite(led16, LOW);   // turn the LED on
  digitalWrite(led6,  LOW);   // turn the LED on

  delay(waitTime);           // wait 1 second

  digitalWrite(led10, HIGH);  // turn the LED off
  digitalWrite(led19, HIGH);  // turn the LED off
  digitalWrite(led16, HIGH);   // turn the LED on
  digitalWrite(led6,  HIGH);   // turn the LED on
  }
```

```
/*
BUG 6, End State Definition,            CORRECTED
Victoria Kim Dissertation
Desired behavior:  All corners light together, flash on and off
*/
// Name the led pins on the Arduino.
int led6  = 6 ;
int led10 = 10 ;
int led17 = 17 ;
int led16 = 16 ;
int led19 = 19 ;
int waitTime = 1000;
// the setup routine runs once when you press reset:
void setup() {
// initialize the digital pins as output.
pinMode(led6, OUTPUT);
pinMode(led10, OUTPUT);
pinMode(led17, OUTPUT);
pinMode(led16, OUTPUT);
pinMode(led19, OUTPUT);
digitalWrite(led17, HIGH);   // Set as positive, Keep pin positive
}
// loop routine
void loop() {
digitalWrite(led10, LOW);   // turn the LED on
digitalWrite(led19, LOW);   // turn the LED on
digitalWrite(led16, LOW);   // turn the LED on
digitalWrite(led6,  LOW);   // turn the LED on
delay(waitTime);            // wait 1 second
digitalWrite(led10, HIGH);  // turn the LED off
digitalWrite(led19, HIGH);  // turn the LED off
digitalWrite(led16, HIGH);   // turn the LED on
digitalWrite(led6,  HIGH);   // turn the LED on
delay(waitTime);            // wait 1 second                         ** Correction

}
```

Activity 15: Compound Debug (Bug 7)

Learning Goal: Find and correct the coding and circuitry errors so that LED's function as described in the scenario. Combines all six engineering and coding challenges.

Tools & Materials:
Bug 7, one for each student
Design Scenario 7
Program for Bug 7
LilyPad Simple
Connectors for LilyPad
Scissors
Thread
Needles
Tool for Circuit Diagrams (paper & pencil or Prezi)
Computer with Arduino environment installed
FTDI and USB cable

Notes:
Do this bug at the second F2F meeting.

Description:
Final Debug Scenario. The project contains 6 bugs of the types identified by Fields, Searle, Kafai & Min.

Design Scenario:
At the end of the EDLT740, Paul decides he wants an e-textile wall-hanging with an Android on it, to integrate both parts of the course. Vicky rushes to create the wall-hangings, and makes multiple errors. Vicky has used a computer to control the behavior of the LEDs', and she has made both coding and circuit errors. Vicky wants both legs to flash on at the same time and then turn off. When the leg LED's turn off, the arm LED's should turn on. The Android robot's leg (B & C) and arm (D & A) LED's should alternate, flashing on and off.

Please locate and correct Vicky's bugs. You will need to cut and re-sew some of the soft circuits.

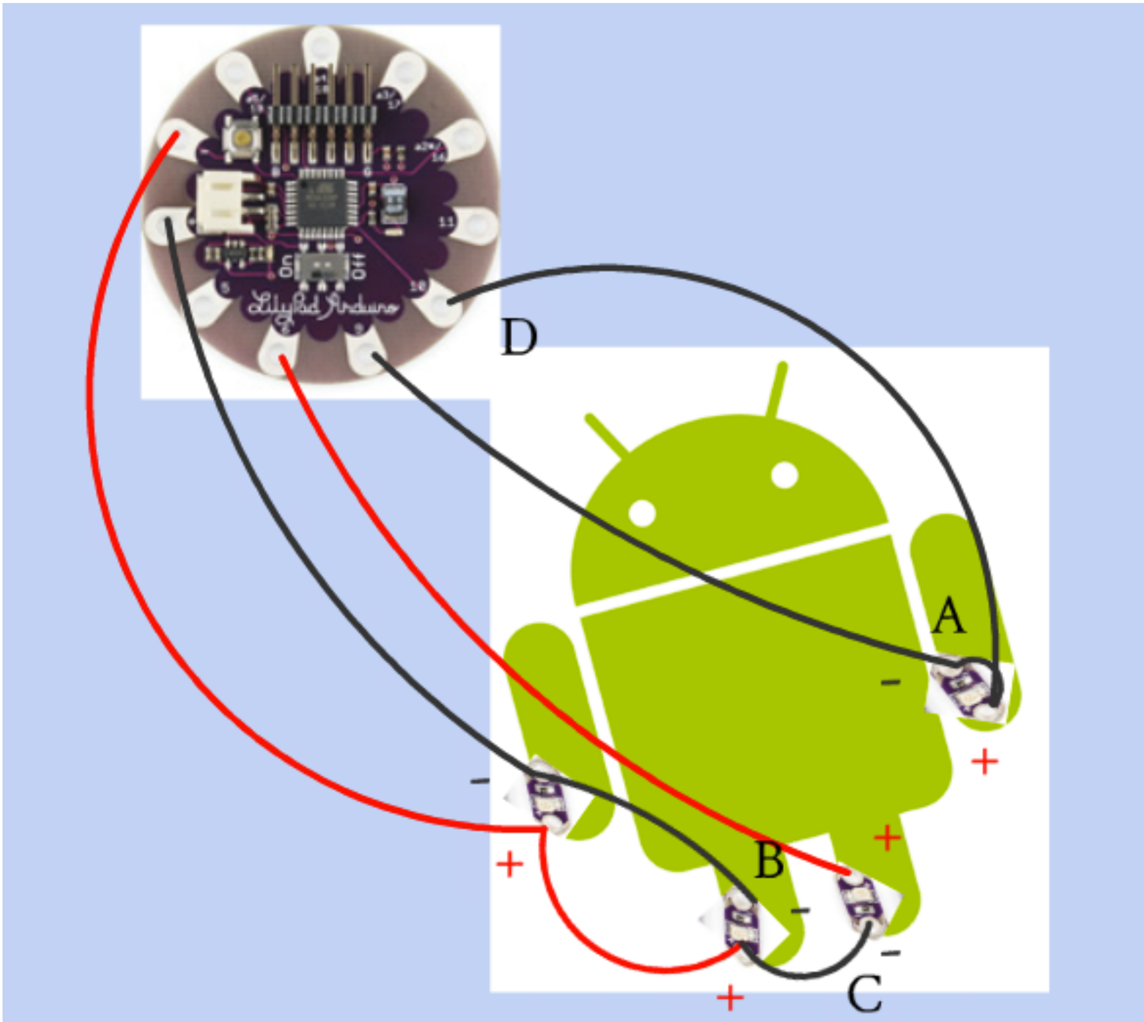


Figure A116. Schematic for final debug scenario, multiple bugs.

Possible correction for engineering errors of Bug 7.

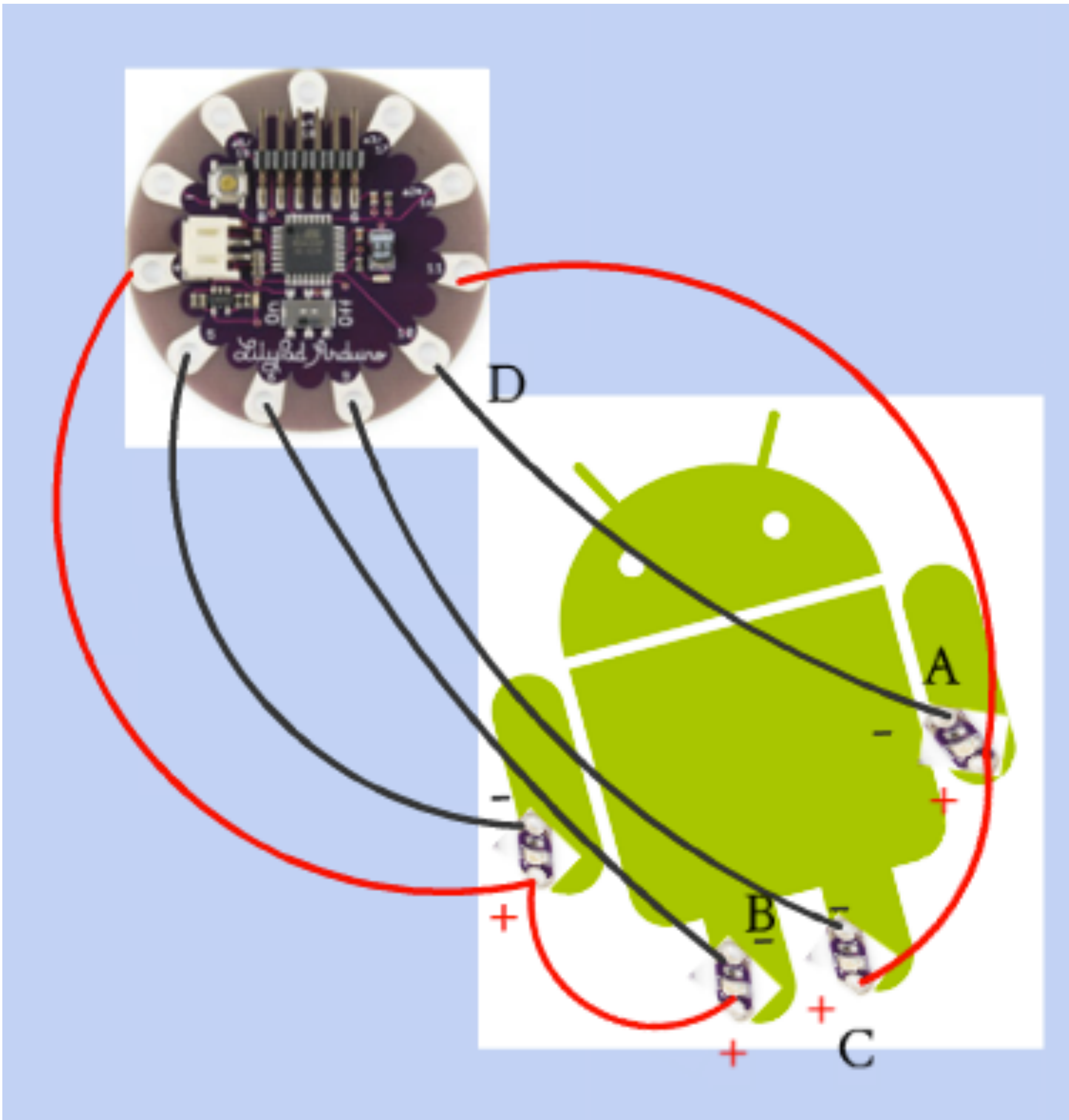


Figure A117. Possible correction of engineering errors in Bug 7.

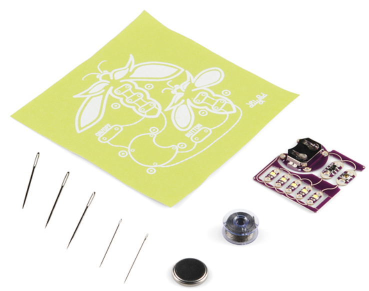
```
/*
BUG 7       BUGGY,       Multibug
Victoria Kim Dissertation

Desired Action: The Robot's arms and legs alternate, in pairs.
*/

// Name the led pins on the Arduino.
int led5 = 5 ;     //   Robot's right arm
int led6 = 6 ;     //   Robot's right leg
int led9 = 9 ;     //   Robot's left leg
int led10 = 10;     //  Robot's left arm

int waitTime = 1000;

// setup runs once
void setup() {
  // initialize the digital pins as an output.
  pinMode(led5, OUTPUT);
  pinMode(led6, OUTPUT);
  pinMode(led9, OUTPUT);
  pinMode(led10, OUTPUT);

  digitalWrite(led5, HIGH);   //start with pin off
  digitalWrite(led6, HIGH);    //start with pin off
  digitalWrite(led9, HIGH);    //start with pin off
  digitalWrite(led10, HIGH);    //start with pin off

}

// Loop routine (continually repeats)
void loop() {
  digitalWrite(led5, LOW);    // turn arms on
  digitalWrite(led10, LOW);   // turn arms on

  digitalWrite(led6, HIGH);   // turn legs off
  digitalWrite(led9, HIGH);   // turn legs off

  delay(waitTime);         // wait 1 seconds

  digitalWrite(led5, HIGH);   // turn arms off
  digitalWrite(led10, HIGH);  // turn arms off

  digitalWrite(led6, LOW);    // turn legs on
  digitalWrite(led9, LOW);    // turn legs on


}
```

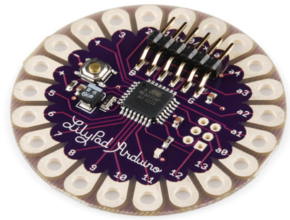
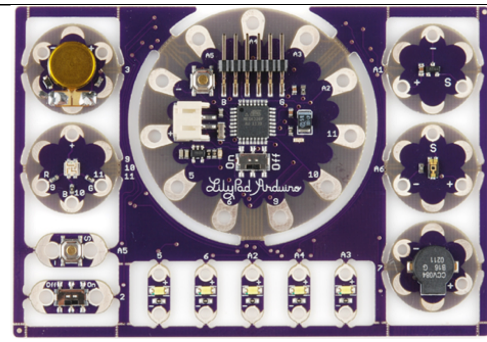
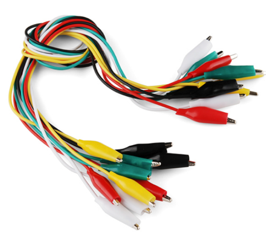
```
/*
 New Bug 7 CORRECTED, Multibug
 Victoria Kim Dissertation

 Desired Action: The Robot's arms and legs alternate, in pairs.
 */

// Name the led pins on the Arduino.
int led5 = 5 ;     //   Robot's right arm
int led6 = 6 ;     //   Robot's right leg
int led9 = 9 ;     //   Robot's left leg
int led10 = 10;     //  Robot's left arm  (neg)
int led11 = 11;     //  Robot's left arm  (pos)                 ***  bug fix

int waitTime = 1000;

// setup runs once
void setup() {
  // initialize the digital pins as an output.
  pinMode(led5, OUTPUT);
  pinMode(led6, OUTPUT);
  pinMode(led9, OUTPUT);
  pinMode(led10, OUTPUT);
  pinMode(led11, OUTPUT);              //                           ***  bug fix

  digitalWrite(led11, HIGH);   //set pin to positive              ***  bug fix

  digitalWrite(led5, HIGH);   //start with pin off
  digitalWrite(led6, HIGH);    //start with pin off               *** bug fix (sewing)
  digitalWrite(led9, HIGH);    //start with pin off
  digitalWrite(led10, HIGH);    //start with pin off

}

// Loop routine (continually repeats)
void loop() {
  digitalWrite(led5, LOW);    // turn arms on
  digitalWrite(led10, LOW);   // turn arms on

  digitalWrite(led6, HIGH);   // turn legs off
  digitalWrite(led9, HIGH);   // turn legs off

  delay(waitTime);          // wait 1 seconds

  digitalWrite(led5, HIGH);   // turn arms off
  digitalWrite(led10, HIGH);  // turn arms off

  digitalWrite(led6, LOW);    // turn legs on
```

```
digitalWrite(led9, LOW);    // turn legs on

delay(waitTime);         // wait 1 seconds                    *** bug fix

}
```

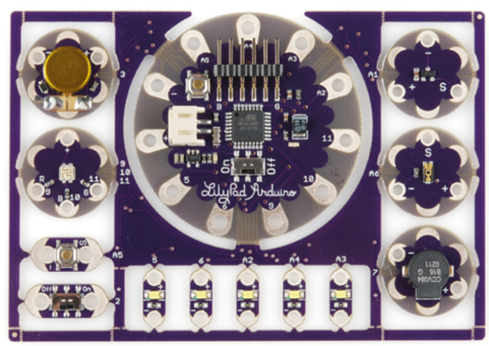
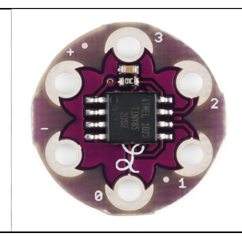
Recommended Parts Lists, Student and Presenter

The following table shows the recommended parts for the course presenter and for the individual students. The part numbers are from SparkFun.

Student Kit **www.SPARKFUN.com** **Wishlist:** **PepperdineEDLT740StudentKit**	**Image**	**SparkFun** **Part Number**
Conductive Thread bobbin X 2		DEV-10867
Starter Kit		RTL-11051
CR2032 batteries, X 2		PRT-00338
Lilypad LEDs (5 pack X 2)		DEV-10081

Lilypad		DEV-09266
LilyPad Protosnap Board		DEV-11262
Alligator Clips		PRT-11037
2 Conductive Magnets		COM-08914
Approximate Total, Without Shipping, Before Discount		*$135.00
In Addition, Students Need:		
Needles		
Needle Threader		
Scissors		
USB cord, mini		

* Students will most likely need some additional components for their individual projects.

Presenter Kit www.SPARKFUN.com Wishlist: PepperdineEDLT740StudentKit		**Part Number**
5 Student kits	(see above)	
Spool of Thread		DEV-08544
Protosnap boards x 5		RTL-11261
Lilypad Twinkles x 5		DEV-11364
Approximate Total, Without Shipping		*755.00
In Addition, Presenter Needs:		
Needles		
Needle Threader		
Scissors		
USB cords, mini connection on one end		

APPENDIX B

Computer Experience

Name: Gender:

I choose not to participate in this study ☐

Please circle the number that represents how you feel about the topic. The words on the end represent the two ends of a spectrum.

	Statement								
1	I have never programmed a computer.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
2	I have never sewn.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
3	I have never completed an e-textile project.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
4	I have never made an electronics project.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
5	I have tinkered with HTML.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
6	I am comfortable programming in several languages.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
7	I learn new programming languages easily.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
8	I have an artistic side to my personality.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
9	I am comfortable installing new software on my computer.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
10	I am comfortable solving problems that arise on my computer.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
11	I enjoy helping others find their errors.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
12	I am good at debugging.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree

APPENDIX C

E-Textile Self-Efficacy Questionnaire

Name: Gender:

I choose not to participate in this study ☐

Please circle the number that represents your level of agreement with the statement on the left. 1 represents strong disagreement and 6 represents strong agreement.

	Statement								
1	I could complete an e-textile project using the LilyPad Arduino environment if there were **no one around** to tell me what to do.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
3	I could complete an e-textile project using the LilyPad Arduino environment if I had only **manuals for reference**.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
4	I could complete an e-textile project using the LilyPad Arduino environment if I had **seen someone else using it** before trying it myself.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
5	I could complete an e-textile project using the LilyPad Arduino environment if I could **call on someone for help** if I got stuck.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
6	I could complete an e-textile project using the LilyPad Arduino environment if **someone else would help me get started**.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
7	I could complete an e-textile project using the LilyPad Arduino environment if I had **a lot time to** complete the project.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
8	I could complete an e-textile project using the LilyPad Arduino environment if I had just the **built-in help facility** for assistance.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
9	I could complete an e-textile project using the LilyPad Arduino environment if **someone showed me how** to do it first.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
10	I could complete an e-textile project using the LilyPad Arduino environment if I had **used similar programming environments** like it before to do other projects.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
11	I could complete an e-textile project using the LilyPad Arduino environment if I **could combine several of the tools** listed above.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree

APPENDIX D

Computational Attitudes

Name: Gender:

I choose not to participate in this study ☐

Please circle the number that represents how you feel about the topic. The words on the end represent the two ends of a spectrum. For example, if you think computing is fascinating, you would circle 5 or 6, depending on your level of fascination.

To me, COMPUTING is

1.	mundane	1	2	3	4	5	6	fascinating
2.	unappealing	1	2	3	4	5	6	appealing
3.	hard	1	2	3	4	5	6	easy
4.	unimportant	1	2	3	4	5	6	important
5.	boring	1	2	3	4	5	6	exciting

To me, ELECTRONICS are

1.	mundane	1	2	3	4	5	6	fascinating
2.	unappealing	1	2	3	4	5	6	appealing
3.	hard	1	2	3	4	5	6	easy
4.	unimportant	1	2	3	4	5	6	important
5.	boring	1	2	3	4	5	6	exciting

To me, E-TEXTILES are

1.	mundane	1	2	3	4	5	6	fascinating
2.	unappealing	1	2	3	4	5	6	appealing
3.	hard	1	2	3	4	5	6	easy
4.	unimportant	1	2	3	4	5	6	important
5.	boring	1	2	3	4	5	6	exciting

To me, TECHNOLOGY is

1.	mundane	1	2	3	4	5	6	fascinating
2.	unappealing	1	2	3	4	5	6	appealing
3.	hard	1	2	3	4	5	6	easy
4.	unimportant	1	2	3	4	5	6	important
5.	boring	1	2	3	4	5	6	exciting

APPENDIX E

The Computational Thinking Rubric for Examining Students' Project Work

(Revised, Grayed Font)
(Informed by Brennan & Resnick, 2012)

Participant:

Partner:

Data Item:
Debugging Pair Work,
Debug Forum,
Wiki,
Project Notebook,
Multi-bug Task,
Final e-textile Project

Computational Thinking Concept, Practice, or Perspective (Brennan & Resnick)	**Description** Location in project, photo, etc., distinguishing characteristics	Error Free **(EF),** Error with Attempt to Correct **(AC),** Error Corrected **(EC)**	**Notes** Debugging Technique Used, Etc.
Concept: Sequences			
Concept: Loops			
Concept: Events			
Concept: Parallelism			
Concept: Conditionals			
Concept: Operators			
Concept: Data			
Elect. Concept: Power, Pos., Neg			
Elect. Concept: Phys. Setup Affects Code			

Elect. Concept: Control LED's			
Elect. Concept: Short Circuits			
Elect. Concept: Connected, Polarity			
Elect. Concept: Indep, Diff Pins			
Elect. Concept: Delay Req to See			
Practice: Being Incremental and Iterative			
Practice: Testing and Debugging			**Subpractices:** • Benchmark • Use a debugging strategy • Establish circuit correctness • Predict expected behavior • Refer to specifications • Use the visibility of the behavior • Verify correct end • Walk through code • Versions
Practice: Reusing and Remixing			
Practice:			

Abstracting and Modularizing			
Practice: Document			
Practice: Keep Code Clean			
Perspective: Expressing			
Perspective: Connecting			
Perspective: Questioning			
Perspective: Adapt to Others' Code			
Perspective: Many Ways to solve			
Perspective: Failure is just a step forward			
Attitudinal Element	**Present**	**Indication of Increase or Decrease**	**Notes**
Confidence: General			**Increase or Decrease?**
Confidence: e-textiles			**Increase or Decrease?**
Confidence: Programming			**Increase or Decrease?**
Interest: e-textiles			
Interest: Programming			
Gender Focus ?			

APPENDIX F

Student Evaluation of Learning Module

(This evaluation will be completed anonymously.)

About this evaluation:

Thank you for your participation in the research for Victoria Kim's doctoral dissertation entitled Development of an E-Textile Learning Module to Increase Computational Thinking Skills in Graduate Education Students.

Your participation in the study continues to be voluntary; you may still withdraw. You can choose to skip any question in this evaluation at your own discretion. Your responses are anonymous. They will be used only to evaluate and improve the design of the e-textile learning module.

Neither your class grade nor your standing in the Doctor of Learning Technologies program will be affected by your evaluation of the learning module.

The questionnaire that you are being asked to complete contains 16 questions that can be answered by selecting a number from 1 to 6 that corresponds to your level of agreement. There are an additional 6 questions that are open ended. The questionnaire should take approximately 15 to 20 minutes to complete.

Thank you again for your time and participation.

Gender:

Please circle the number that represents your level of agreement with the statement on the left. 1 represents strong disagreement and 6 represents strong agreement.

	Statement								
1	The introduction to circuits helped me understand how circuits work.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
2	We should have spent more time exploring how e-textiles work.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
3	We spent too much time going over e-textile basics.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
4	I had all the materials I needed to learn about e-textiles.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
5	We should have spent more time on the programming part of e-textiles.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
6	We spent too much time going over programming basics.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
7	The background fabric of the bugs detracted from the task.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
8	I really liked being able to experiment with e-textiles.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
9	I would have been more interested in Arduino that use soldering instead of sewing.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
10	The debug forum was useful and informative.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
11	The class Wiki was useful and informative.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
12	The project notebook was useful for me.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree

13	The design scenario bugs were too easy.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
14	The programming part of e-textile projects is harder than the circuit sewing part.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
15	The circuit sewing part of e-textile projects is harder than the programming part.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree
16	E-textiles make good projects for both males and females.	Strongly Disagree	1	2	3	4	5	6	Strongly Agree

Please answer the open-ended questions.

17. What suggestions do you have to improve this e-textile learning module for future students?

18. What part of the e-textile learning module did you find most enjoyable?

19. Did your attitude towards computational thinking change?

20. What outside sources did you use to help you with your e-textile project?

21. What resource did you turn to most often?

22. Do you have anything else you would like to share?

APPENDIX G

Invitation to Participate in Study

Title of Study: **Development of an E-Textile Debugging Module to Increase Computational Thinking among Graduate Education Students.**

My name is Victoria Kim, and I am a doctoral candidate in Pepperdine University's Doctor of Education in Learning Technologies program. I would like to invite you to participate in a research study. This research study will focus on designing an e-textile learning module to develop computational thinking concepts, practices, and perspectives. The results of this study will contribute to the partial fulfillment of the dissertation requirement for my doctoral degree and will hopefully advance understanding of how computational thinking can be developed by people from diverse cultures and genders.

Taking part in this study is voluntary and will not directly help nor hurt your grade in EDLT740. You may stop taking part in this study at any time. Your identity will not be revealed in any part of the study publication. Your personal information will be kept confidential and will not be shared with anyone else.

There will be several stages to the research.

Stage 1: Read the Research Information Sheet and consent to participate in the study.

Stage 2: Fill out questionnaires about Computer Experience and Computational Attitudes.

Stage 3: As a course activity, you will work as part of a pair to find and correct six bugs in e-textile projects. With your permission, you and your partner's collaboration will be audio recorded. These debugging activities will take place at your first F2F meeting for the semester.

Stage 4: At the end of the first F2F meeting, fill out the questionnaire regarding E-Textile Self-Efficacy.

Stage 5: During the following 4 weeks, maintain a weekly project notebook describing your challenges and showing the progress you are making with your course e-textile project.

Stage 6: At the second F2F meeting, as part of the course activities, you will individually work on a multi-bug design scenario.

Stage 7: At the end of the second F2F meeting, fill out questionnaires regarding Computer Attitudes and E-Textile Self-Efficacy.

Stage 8: At the end of the second F2F, participate in a group interview about the multi-bug design scenario. This interview will be audio recorded.

Stage 9: Anonymously complete the Student Evaluation of the Learning Module. Invitations to complete the evaluation will be sent to the students' school email by the principal researcher and will be completed by the students outside of the course time. Because this is a design-based research study your ideas about how to improve the learning module are especially valuable.

The chart below identifies the activities that are course requirements and those that are required for the research study.

Activity Requirements for Course and Research

Activity	Required for EDLT740	Required for Research Study	Audio Recording (optional)	Images Captured (optional)
Introduction to E-Textiles	X			
6 Debugging Design Scenarios at F2F 1	X		X	X
Self-Efficacy Questionnaire		X		
Computer Experience Questionnaire		X		
Computational Attitude Questionnaire		X		
Student Evaluation of Learning Module		X		
Group Interview		X	X	
Debug Forum	X			
Class E-textile Wiki	X			
Project Notebook		X		
E-Textile Project	X			X
Multi-bug Design Scenario at F2F 2	X			X

If you would like to participate in this study, please read and sign the attached Informed Consent for Participation in Research Activities.

Thank you for considering being part of this research.

Victoria Kim
vkim@pepperdine.edu, 510-673-8275

APPENDIX H

Research Information Sheet

Title of Study: **Development of an E-Textile Debugging Module to Increase Computational Thinking among Graduate Education Students.**
Principal Researcher (PR): Victoria Kim, (510) 673-8275

Purpose:
You are being asked to be a participant in a design-based research study of the development of an e-textile learning module whose aim is to provide a content-rich environment in which computational concepts, practices, and perspectives can be constructed. You are being invited to participate because you are enrolled in Pepperdine University's Fall 2013 EDLT740 course. This research is in partial fulfillment of the dissertation requirements for the principal researcher's doctoral degree.

Study Procedures:
If you choose to participate in the study, you will be asked to do the following:

- Give permission to be recorded as you work with your partner on six debugging design scenarios.
- Fill out questionnaires that describe your previous computer experience, your e-textile self-efficacy, your computational attitude, and respond to a survey where you evaluate the learning module.
- Keep a project notebook.
- Participate in a group interview at the end of the multi-bug design scenario.
- Give permission to have photos taken while working on debugging design scenarios.

The following chart shows a comparison of EDLT740 course activities and related research study activities.

Activity Requirements for Course and Research

Activity	Required for EDLT740	Required for Research Study	Audio Recording (optional)	Images Captured (optional)
Introduction to E-Textiles	X			
6 Debugging Design Scenarios at F2F 1	X		X	X
Self-Efficacy Questionnaire		X		
Computer Experience Questionnaire		X		
Computational Attitude Questionnaire		X		

Student Evaluation of Learning Module		X		
Group Interview		X	X	
Debug Forum	X			
Class E-textile Wiki	X			
Project Notebook		X		
E-Textile Project	X			X
Multi-bug Design Scenario at F2F 2	X			X

Benefits:

Possible benefits of this research study are:

- Learning about and creating e-textile projects.
- Gaining a deeper understanding of computational thinking.
- Developing greater confidence in computational skills.
- Having an opportunity to construct knowledge about e-textiles and computational thinking.
- Development of an alternative entry point into computer science.
- Development of a computing environment that appeals to a diverse group of people.

Risks:

There are certain minimal risks and discomforts that might be associated with this research. These risks include:

- Because e-textile circuits are sewn with conductive thread, you will use needles, scissors, and handle small parts.
- Additional time will be required to respond to the questionnaires, group interview, and evaluation. (Approximate total of about 120 minutes.) Time will also be spent maintaining a project notebook. (Approximate total of 20 minutes per week for 4 weeks.)
- Possible frustration may result if you are unable to debug the e-textile projects.
- Frustration or boredom could result from responding to multiple questionnaires.

Costs:

- There will be no additional costs to you for participation in this research study. You will use required course e-textile components.

Compensation: You will not be paid for taking part in this study.

Confidentiality: You will be identified in the research records by a code number. When the results of this research are published or discussed in conferences, no information will be included that would reveal your identity, unless you give permission for your image to be used. Your choice to participate, or not, in the research study will be kept confidential from the course professors.

Voluntary Participation/Withdrawal: Taking part in this study is voluntary. You are free to not answer any questions or withdraw at any time. Your decision will not change any present or future relationships with Pepperdine University nor affect your grade in EDLT740. You will still be responsible for graded course projects required for EDLT740. See the preceding Activity Requirements Chart to see the course requirements.

Questions: If you have any questions about this study, now or in the future, you may contact Victoria Kim at (510) 673-8275, vherbstkim@gmail.com. If you have questions or concerns about your rights as a research participant, you may contact the chairperson of the Graduate and Professional School IRB, Pepperdine University, Doug Leigh, 310-568-2389, dleigh@pepperdine.edu

Participation: By signing the consent form you are agreeing to participate in this study.

________________________________, Participant Signature ________, Date

APPENDIX I

Informed Consent for Participation

Participant: ________________________________ Principal Researcher:Victoria Kim

Title of Study: **Development of an E-Textile Debugging Module to Increase Computational Thinking Among Graduate Education Students.**

1. I ________________________________ , agree to participate in the research study being conducted by Victoria Kim under the direction of Dr. Linda Polin.
2. The overall purpose of this research is:

   To develop a debugging learning module that will use e-textile projects and the LilyPad Arduino programming environment to provide graduate education students with the opportunity to construct computational thinking concepts, practices, and perspectives and increase their interest and confidence in their computing abilities.

Specifically, this study has the following research objectives:

A. Identify e-textile learning module characteristics and procedures that contribute to developing computational thinking concepts, practices, and perspectives, based upon theory and student experience.
B. Design an e-textile debugging learning module, based on the identified characteristics and procedures, that
   1 Contributes to the development of computational thinking concepts, practices, and perspectives.
   2 Increases confidence in computing abilities.
   3 Stimulates interest in computer science and other computer fields.
   4 Appeals to non-computer science majors.
C. Identify assessment characteristics and procedures to show the development of computational thinking concepts, practices, and perspectives.

3. My participation will involve the following:

   a. Filling out 4 questionnaires (E-Textile Self-Efficacy Questionnaire, Computational Attitude Questionnaire, Computer Experience Questionnaire, Student Evaluation of Learning Module). The total expected time for all questionnaires is 45 to 60 minutes.
   b. Maintenance of a weekly project notebook. The expected time for the notebook entries is 20 minutes per week.
   c. Participation in a group interview. The total expected time for the group interview is 60 minutes or less.

4. My participation in the study will take place during the Fall 2013 semester. The study shall be conducted at Pepperdine University in West Los Angeles in conjunction with the EDLT740 course.

5. I understand that the possible benefits to myself or society from this research are:

- Learning about and creating e-textile projects.
- Deeper understanding of computational thinking.
- Greater confidence in computational skills.
- Opportunity to construct knowledge about e-textiles and computational thinking.
- Development of an alternative entry point into computer science.
- Development of a computing environment that appeals to a diverse group of people.

6. I understand that there are minimal risks and discomforts that might be associated with this research. These risks include:

    - E-textile circuits are sewn with conductive thread. You will use needles, scissors, and handle small parts.
    - Additional time will be required responding to the questionnaires and evaluation.
    - Possible frustration may result if you are unable to debug the e-textile projects.
    - Frustration or boredom could result from responding to multiple questionnaires.

7. I understand that I may choose not to participate in this research at any point. I will still be held accountable for the elements of EDLT740 that are not parts of the research. I understand that the activities for the course will result in data for the study. I may opt out of having my data used in the study, but I may not opt out of the course activities.

Activity Requirements for Course and Research

Activity	Required for EDLT740	Required for Research Study	Audio Recording (optional)	Images Captured (optional)
Introduction to E-Textiles	X			
6 Debugging Design Scenarios at F2F 1	X		X	X
Self-Efficacy Questionnaire		X		
Computer Experience Questionnaire		X		
Computational Attitude Questionnaire		X		
Student Evaluation of Learning Module		X		
Group Interview		X	X	
Debug Forum	X			
Class E-textile Wiki	X			
Project Notebook		X		
E-Textile Project	X			X
Multi-bug Design Scenario at F2F 2	X			X

8. I understand that my participation is voluntary and that I may refuse to participate and/or withdraw my consent and discontinue participation in the project or activity at any time without penalty or loss of benefits to which I am otherwise entitled. My course grade will not be affected by non-participation in the research study. There is no remuneration for my participation in the study.

9. I understand that the researcher will take all reasonable measures to protect the confidentiality of my records and my identity will not be revealed in any publication that may result from this project. The confidentiality of my records will be maintained in accordance with applicable state and federal laws.

10. I understand that the principal researcher, Victoria Kim, is willing to answer any inquiries I may have concerning the research herein described. I understand that I may contact Victoria Kim at 510-673-8275 or vherbstkim@gmail.com if I have other questions or concerns about this research. If I have questions about my rights as a research participant, I understand that I can contact Dr. Doug Leigh, Chairperson of the Graduate and Professional School IRB, Pepperdine University, 310-568-2389, dleigh@pepperdine.edu.

11. I will be informed of any significant new findings developed during the course of my participation in this research, which may have a bearing on my willingness to continue in the study.

12. I understand to my satisfaction the information regarding participation in the research project. All my questions have been answered to my satisfaction. I have received a copy of this informed consent form, which I have read and understand. I hereby consent to participate in the research described above.

___________________________________, Participant Signature _________, Date

By checking this box, I give permission to Victoria Kim to use images of my e-textile projects and me in the publication of this research. ☐

By checking this box, I give permission to Victoria Kim to make audio recordings of my participation in the debugging scenarios and group interview. ☐

I have explained and defined in detail the research procedure in which the subject has consented to participate. Having explained this and answered any questions, I am cosigning this form and accepting this person's consent.

___________________________________, Researcher Signature _________, Date

APPENDIX J

Design Scenarios for E-Textile Debugging Activities

Design Scenarios for

E-Textile Debugging Activities

by
Victoria Kim

Background

Paul and Linda have been teaching a hands-on e-textile class and want to have an end-of-course celebration. To increase the party atmosphere of the classroom, they decide that it would be appropriate to have e-textile wall hangings. They are very busy, however, so they assign the task to Vicky, the TA.

Vicky decides that since the school mascot is waves that water/beach related wall hangings are a good idea. She has rushed, however, and there are bugs in the e-textiles. Please help Vicky correct the problems.

Design Scenario 1

Design Scenario:
For the tropical drinks e-textile, Vicky decides that she wants the LED at Point A to have a certain behavior, the LED's at Points B & C to behave the same as each other, but different from the LED at Point A, and the LED's at Points D & E to behave the same as each other, but different from all the others.

Please locate and correct Vicky's bug. You may need to cut and re-sew some of the soft circuits. You can draw on the diagrams, if you wish.

Tools & Materials:
Debug 1
Conductive thread
Coin cell battery
Scissors
Needle
Needle threader
Alligator Clips

Notes:
Record your interactions with your partner
Work as Partners
Discuss your thoughts about what you are doing to debug
- What is happening incorrectly?
- What might cause that behavior?
- How can you fix it?

Image for Design Scenario 1

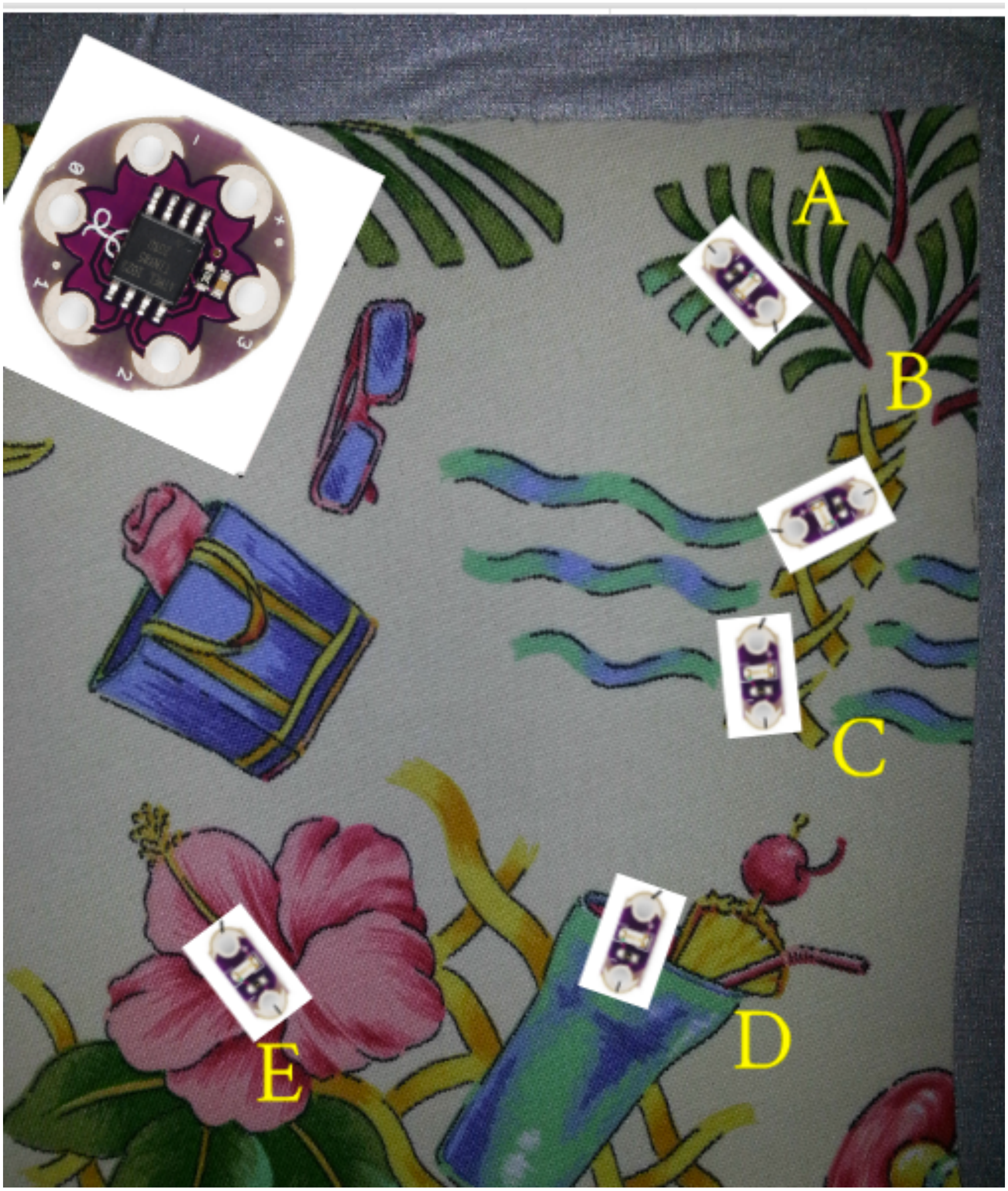


Figure J118. Design scenario, Bug 1.

Design Scenario 2

Design Scenario:
Vicky's next e-textile project is a colorful flip-flops scene. Vicky has decided that she wants the LED at Point A to have a certain behavior, the LED's at Points B & C to behave the same as each other, but different from the LED at Point A, and the LED's at Points D & E to behave the same as each other, but different from all the others.

Please locate and correct Vicky's bug. You may need to cut and re-sew some of the soft circuits. You can draw on the diagrams, if you wish.

Tools & Materials:
Debug 2
Conductive thread
Coin cell battery
Scissors
Needle
Needle threader
Alligator Clips

Notes:
Record your interactions with your partner
Work as Partners
Discuss your thoughts about what you are doing to debug

What is happening incorrectly?
What might cause that behavior?
How can you fix it?

Image for Design Scenario 2

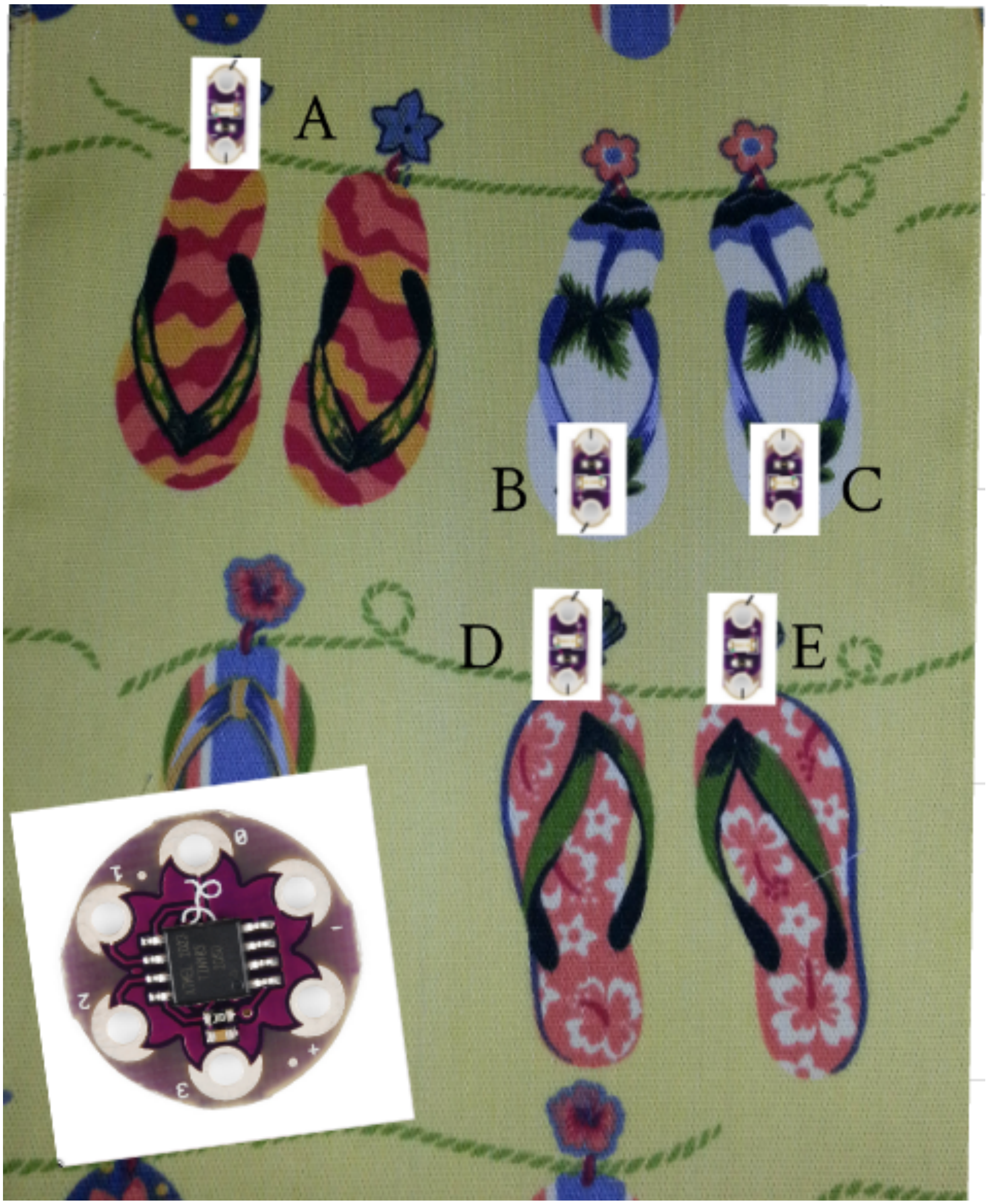


Figure J119. Design scenario, Bug 2.

Design Scenario 3

Design Scenario:
Vicky's third e-textile is a blue sailboat. She wants the LED's on the sail, the LED's at Points A, B, and C, to behave the same way, and the LED at Point D to behave independently.

Vicky's wall hanging is not lighting up as she wants. Please locate and correct her bug. You may need to cut and re-sew some of the soft circuits. You can draw on the diagrams, if you wish.

Tools & Materials:
Debug 3
Conductive thread
Coin cell battery
Scissors
Needle
Needle threader
Alligator Clips

Notes:
Record your interactions with your partner
Work as Partners
Discuss your thoughts about what you are doing to debug
- What is happening incorrectly?
- What might cause that behavior?
- How can you fix it?

Image for Design Scenario 3

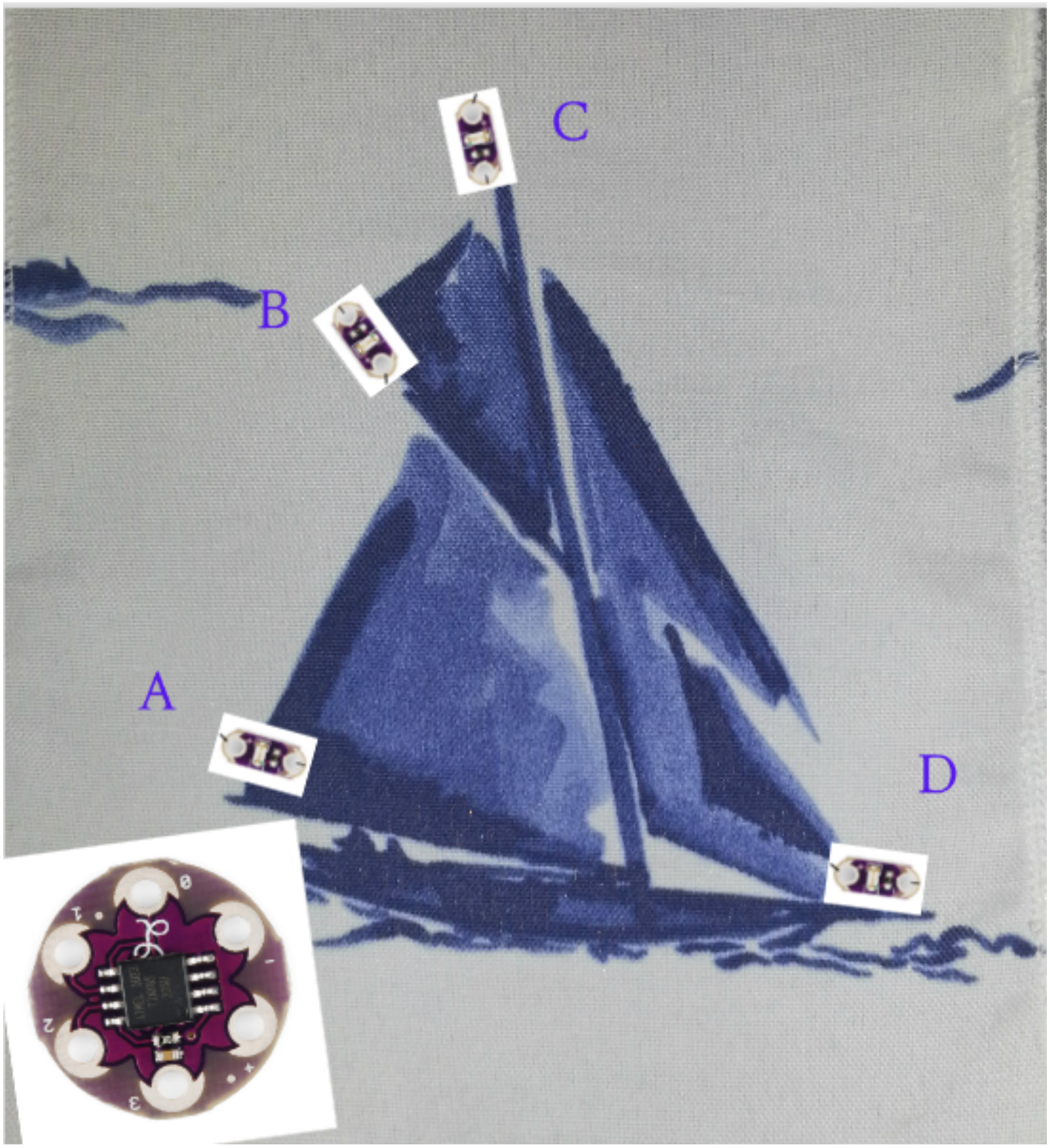


Figure J120. Design scenario, Bug 3.

Design Scenario 4

Design Scenario:
Vicky's fourth wall-hanging is more sophisticated. She has used a computer to control the behavior of the LED's. Vicky wants to start with the spiked shell, and turn shell lights on and off, individually, in a counter clockwise direction.

The program is not working correctly, however. Please locate and correct Vicky's bug. You can draw on the diagrams, if you wish.

Tools & Materials:
Debug 4
Computer with Arduino environment installed
Lilypad with connectors
FTDI Connector
USB mini cord
Bugged Program 4
Alligator Clips
Coin Cell Battery

Notes:
Record your interactions with your partner
Work as Partners
Discuss your thoughts about what you are doing to debug
    What is happening incorrectly?
    What might cause that behavior?
    How can you fix it?

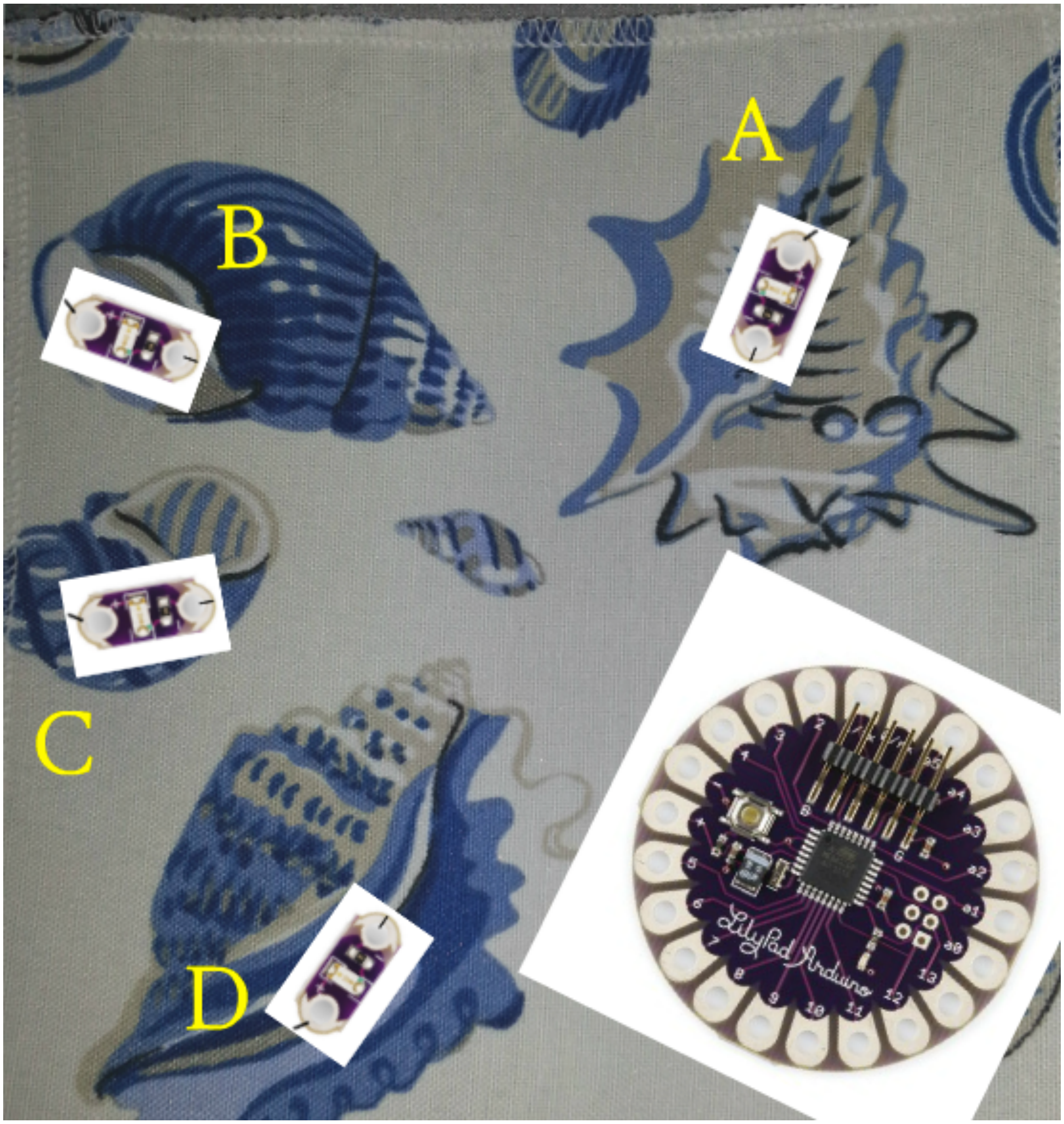


Figure J121. Design scenario, Bug 4.

Design Scenario 5

Design Scenario:
Vicky's fifth wall-hanging is also the more sophisticated kind. She has again used the computer to control the behavior of her LED's. Vicky wants the bottom flag (not on boat) to stay on, the middle flag (not on boat) to blink quickly, and the top flag (not on boat) to blink slowly. Vicky wants the light on the mast to blink twice quickly, then pause, then repeat.

The program is not working correctly, however. Please locate and correct Vicky's bug. You can draw on the diagrams, if you wish.

Tools & Materials:
Debug 5
Computer with Arduino environment installed
LilyPad with connectors
FTDI Connector
USB mini cord
Bugged Program 5
Alligator Clips
Coin Cell Battery

Notes:
Record your interactions with your partner
Work as Partners
Discuss your thoughts about what you are doing to debug
- What is happening incorrectly?
- What might cause that behavior?
- How can you fix it?

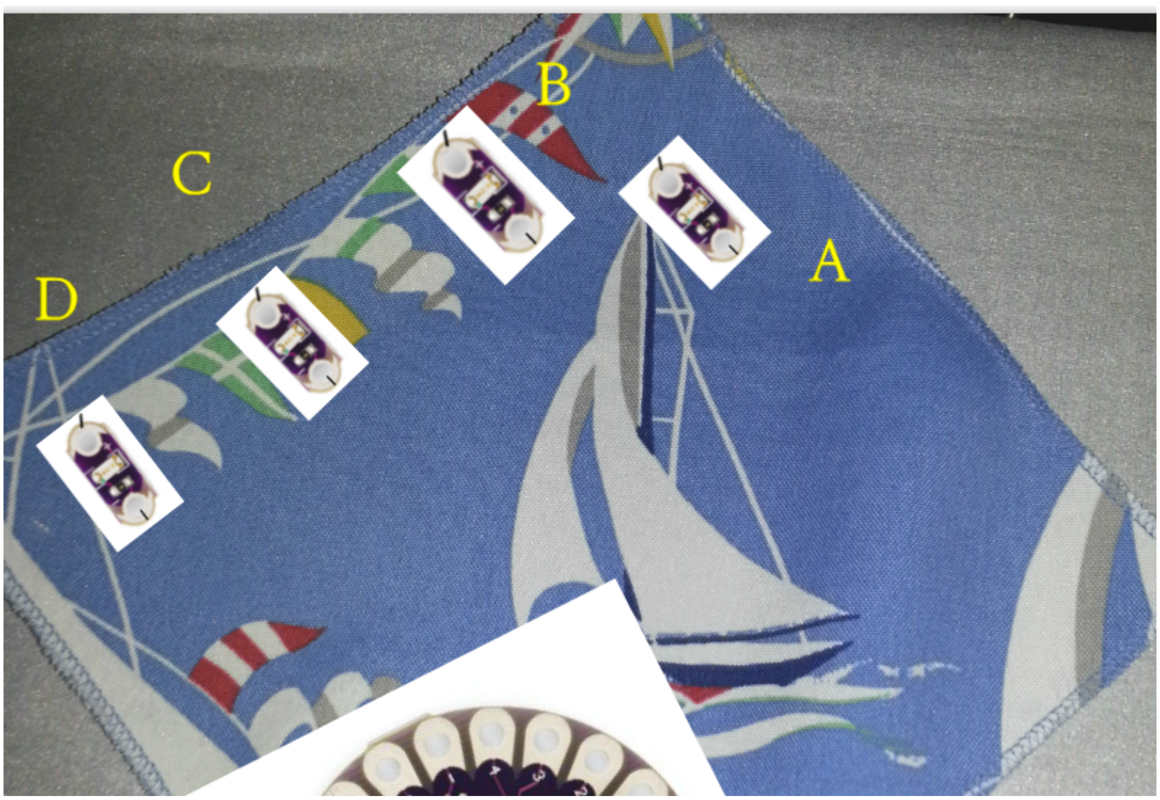


Figure J122. Design scenario, Bug 5.

Design Scenario 6

Design Scenario:
Vicky's final wall-hanging is the sophisticated type again, and she has used a computer to control the behavior of the LED's. Vicky wants all the corners to light together, flashing on and off.

The program is not working correctly, however. Please locate and correct Vicky's bug. You can draw on the diagrams, if you wish.

Tools & Materials:
Debug 6
Computer with Arduino environment installed
LilyPad with connectors
FTDI Connector
USB mini cord
Bugged Program 6
Alligator Clips
Coin Cell Battery

Notes:
Record your interactions with your partner
Work as Partners
Discuss your thoughts about what you are doing to debug
What is happening incorrectly?
What might cause that behavior?
How can you fix it?

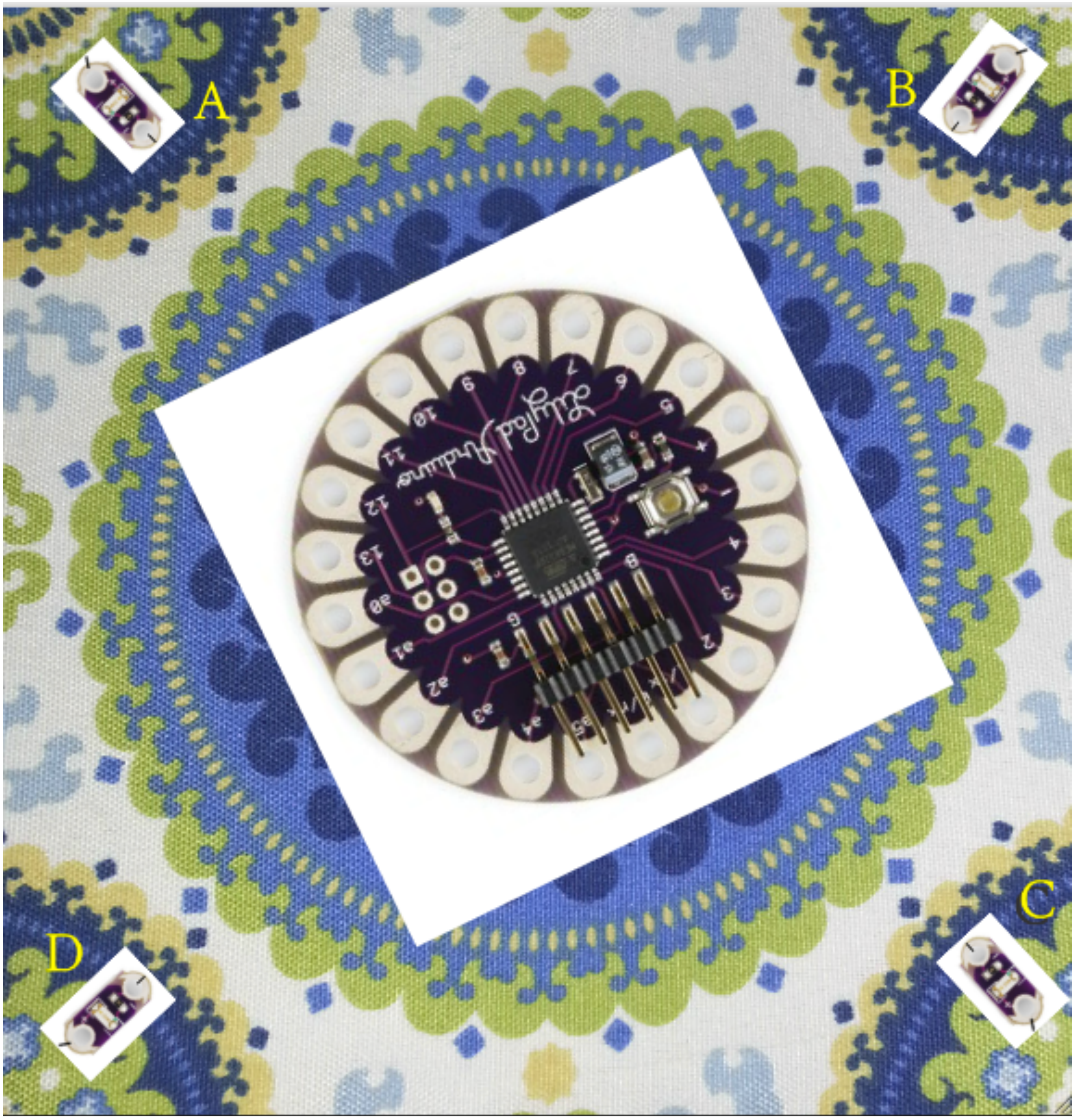


Figure J123. Design scenario, Bug 6.

Thank you for helping Vicky correct the bugs in her wall-hangings. They will add to a great atmosphere for the end-of-course celebration!

Debug Scenario 7

Goal: Find and correct the **coding and circuitry errors** so that LED's function as described in the scenario. Combines all six engineering and coding challenges.

Design Scenario:
At the end of the EDLT740, Paul decides he wants an e-textile wall-hanging with an Android on it, to integrate both parts of the course. Vicky rushes to create the wall-hangings and makes multiple errors. Vicky has used a computer to control the behavior of the LEDs', and she has made both coding and circuit errors. Vicky wants both legs to flash on at the same time and then turn off. When the leg LED's turn off, the arm LED's should turn on. The Android robot's leg (B & C) and arm (D & A) LED's should alternate, flashing on and off.

Please locate and correct Vicky's bugs. You will need to cut and re-sew some of the soft circuits.

Tools & Materials:
Bug 7
Design Scenario 7
Program for Bug 7
LilyPad
Connectors for LilyPad
Scissors
Thread
Needles
Tool for Circuit Diagrams (paper & pencil or Prezi)
Computer with Arduino environment installed
FTDI and USB cable
Alligator Clips

Notes:
Record your interactions with your partner. Please use your Lifescribe to make a note as you locate/fix a bug.
Work as Partners
Discuss your thoughts about what you are doing to debug

- What is happening incorrectly?
- What might cause that behavior?
- Since there are multiple bugs, what should you start on first?
- How can you fix the bugs?

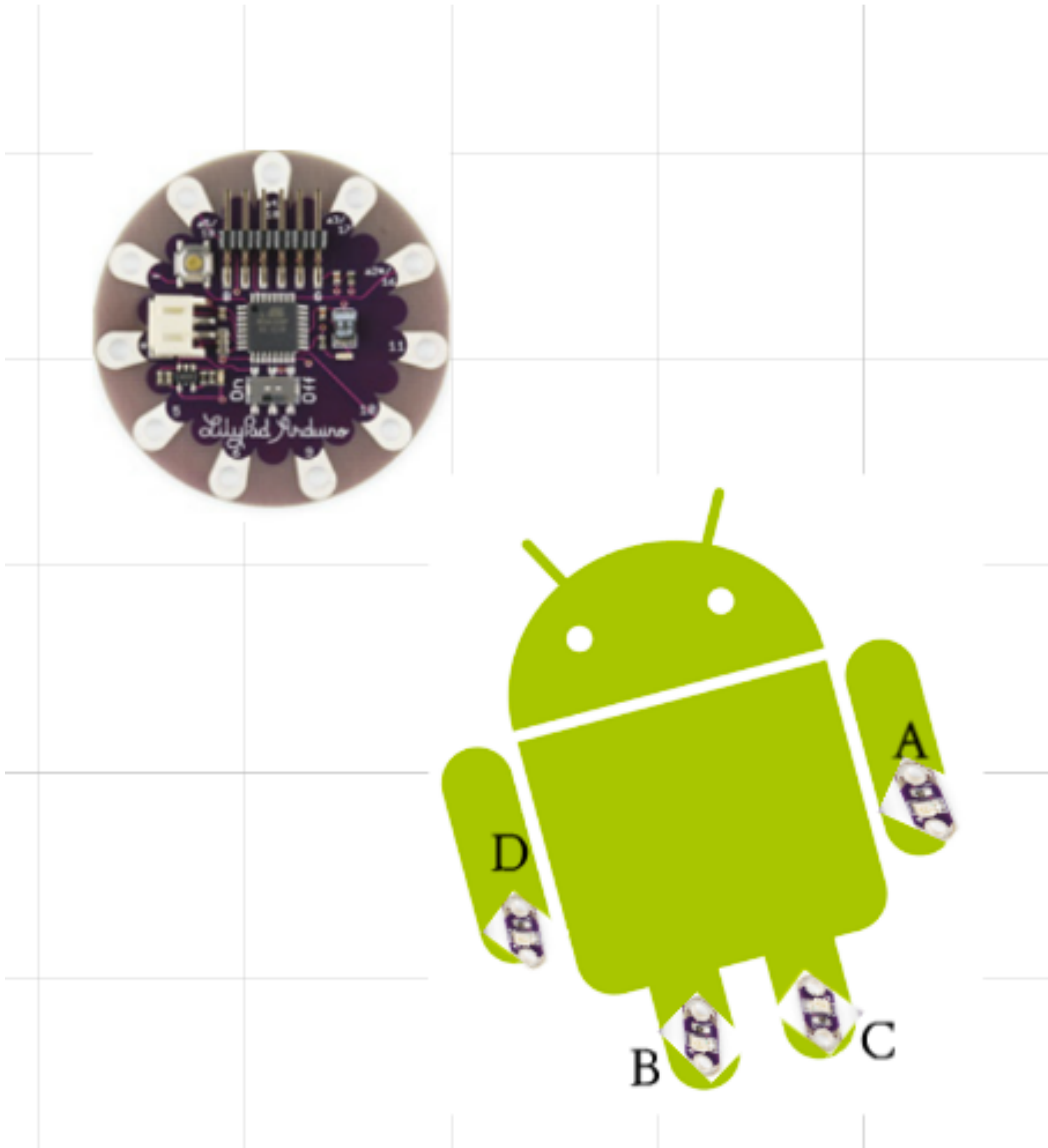


Figure J124. Design scenario, Bug 7.

APPENDIX K

Programs Used in Debugging Exercises

```
/*
BUG 4, Constant versus Variable Pins   BUGGY
Victoria Kim Dissertation

Desired Action: Starting with spiked shell, turn shell lights on and off,
individually, in a counter clockwise direction.
*/

// Name the led pins on the Arduino.
int led19 = 19 ;   //   spikey shell
int led5 = 5 ;     //   smooth shell
int led6 = 6 ;     //   little shell
int led9 = 9 ;     //   rough shell
int led10 = 10 ;   //   rough shell 2nd conn
int waitTime = 1000;

// setup runs once
void setup() {
  // initialize the digital pins as an output.
  pinMode(led19, OUTPUT);
  pinMode(led5, OUTPUT);
  pinMode(led6, OUTPUT);
  pinMode(led9, OUTPUT);
  pinMode(led10, OUTPUT);
  digitalWrite(led10, LOW);    //set pin to negative

  digitalWrite(led19, HIGH);   //start with pin off
  digitalWrite(led5, HIGH);    //start with pin off
  digitalWrite(led6, HIGH);    //start with pin off
  digitalWrite(led9, HIGH);    //start with pin off

}

// Loop routine (continually repeats)
void loop() {
  digitalWrite(led19, LOW);   // turn spikey on
  delay(2*waitTime);          // wait 2 second
  digitalWrite(led19, HIGH);  // turn spikey off
  delay(2*waitTime);          // wait 2 second

  digitalWrite(led5, LOW);    // turn smoothy on
  delay(2*waitTime);          // wait 2 second
  digitalWrite(led5, HIGH);   // turn smoothy off
  delay(2*waitTime);          // wait 2 second

  digitalWrite(led6, LOW);    // turn baby on
  delay(2*waitTime);          // wait 2 second
  digitalWrite(led6, HIGH);   // turn baby off
```

```
delay(2*waitTime);          // wait 2 second

digitalWrite(led9, LOW);    // turn roughy on
delay(2*waitTime);          // wait 2 second
digitalWrite(led9, HIGH);   // turn roughy off
delay(2*waitTime);          // wait 2 second


}
```

```
 /*
BUG 4 CORRECTED, Constant versus Variable Pins
Victoria Kim Dissertation

Desired Action: Starting with spiked shell, turn shell lights on and off,
individually, in a counter clockwise direction.
*/

// Name the led pins on the Arduino.
int led19 = 19 ;  //   spikey shell
int led5 = 5 ;    //   smooth shell
int led6 = 6 ;    //   little shell
int led9 = 9 ;    //   rough shell
int led10 = 10;    // create a negative pin
int waitTime = 1000;

// setup runs once
void setup() {
  // initialize the digital pins as an output.
  pinMode(led19, OUTPUT);
  pinMode(led5, OUTPUT);
  pinMode(led6, OUTPUT);
  pinMode(led9, OUTPUT);
  pinMode(led10, OUTPUT);

  digitalWrite(led10, HIGH);   //set pin to positive   ** Correction

  digitalWrite(led19, HIGH);   //start with pin off
  digitalWrite(led5, HIGH);    //start with pin off
  digitalWrite(led6, HIGH);    //start with pin off
  digitalWrite(led9, HIGH);    //start with pin off

}

// Loop routine (continually repeats)
void loop() {
  digitalWrite(led19, LOW);   // turn spikey on
  delay(2*waitTime);          // wait 2 second
  digitalWrite(led19, HIGH);  // turn spikey off
  delay(2*waitTime);          // wait 2 second

  digitalWrite(led5, LOW);    // turn smoothy on
  delay(2*waitTime);          // wait 2 second
  digitalWrite(led5, HIGH);   // turn smoothy off
  delay(2*waitTime);          // wait 2 second

  digitalWrite(led6, LOW);    // turn baby on
  delay(2*waitTime);          // wait 2 second
```

```
digitalWrite(led6, HIGH);   // turn baby off
delay(2*waitTime);          // wait 2 second

digitalWrite(led9, LOW);    // turn roughy on
delay(2*waitTime);          // wait 2 second
digitalWrite(led9, HIGH);   // turn roughy off
delay(2*waitTime);          // wait 2 second

}
```

```
/*
BUG 5, Control Flow                    BUGGY
Victoria Kim Dissertation

Desired Actions: Bottom flag (not on boat) stays on,
                 Middle flag (not on boat) blinks quickly,
                 Top flag (not on boat) blinks slowly,
                 Light on mast blinks twice quickly, then pauses, then repeats.
*/

// Name the led pins on the Arduino.
int led11 = 11 ;
int led16 = 16 ;
int led17 = 17 ;
int led19 = 19 ;
int waitTime = 1000;
int cycleCnt = 0;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pins as output.
  pinMode(led11, OUTPUT);  //mast
  pinMode(led16, OUTPUT);  //mast
  pinMode(led17, OUTPUT);  //top
  pinMode(led19, OUTPUT);  //middle

  digitalWrite(led17, LOW);   // turn the LED on
  digitalWrite(led19, LOW);   // turn the LED on
  digitalWrite(led11, LOW);   // turn the LED on

  delay(waitTime/2);          // wait 1/2 second

}

// loop routine
void loop() {

  digitalWrite(led19, HIGH);  // turn the LED off
  digitalWrite(led11, HIGH);  // turn the LED off

  // led17 is on slower cycle
  // fmod divides by the second number and looks at the remainder
  if (fmod(cycleCnt,2) == 0)
  {
    digitalWrite(led17, HIGH);   // turn the LED off
```

```
  }

  delay(waitTime/2);

  digitalWrite(led19, LOW);  // turn the LED on


  // slower blink
  if (fmod(cycleCnt,2) == 0)
  {
  digitalWrite(led17, LOW);   // turn the LED on
  }

  //Every third time, the mast LED will not come on
  if (fmod(cycleCnt,3) != 0)
  {
    digitalWrite(led11, LOW);
  }

  delay(waitTime/2);

  cycleCnt = cycleCnt + 1;

}
```

```
/*
 BUG 5, Control Flow                    CORRECTED
 Victoria Kim Dissertation

 Desired Actions: Bottom flag (not on boat) stays on,
                Middle flag (not on boat) blinks quickly,
                Top flag (not on boat) blinks slowly,
                Light on mast blinks twice quickly, then pauses, then repeats.
 */

// Name the led pins on the Arduino.
int led11 = 11 ;
int led16 = 16 ;
int led17 = 17 ;
int led19 = 19 ;
int waitTime = 1000;
int cycleCnt = 0;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pins as output.
  pinMode(led11, OUTPUT);  //mast
  pinMode(led16, OUTPUT);  //mast
  pinMode(led17, OUTPUT);  //top
  pinMode(led19, OUTPUT);  //middle
  digitalWrite(led16, HIGH);        //                  **  Corrected

  digitalWrite(led17, LOW);   // turn the LED on
  digitalWrite(led19, LOW);   // turn the LED on
  digitalWrite(led11, LOW);   // turn the LED on

  delay(waitTime/2);         // wait 1/2 second

}

// loop routine
void loop() {

  digitalWrite(led19, HIGH);  // turn the LED off
  digitalWrite(led11, HIGH);  // turn the LED off

  // led17 is on slower cycle
  // fmod divides by the second number and looks at the remainder
  if (fmod(cycleCnt,2) == 0)
  {
    digitalWrite(led17, HIGH);   // turn the LED off
  }

```

```
delay(waitTime/2);

digitalWrite(led19, LOW);  // turn the LED on

// slower blink
if (fmod(cycleCnt,2) == 0)
{
digitalWrite(led17, LOW);   // turn the LED on
}

//Every third time, the mast LED will not come on
if (fmod(cycleCnt,3) != 0)
{
  digitalWrite(led11, LOW);
}

delay(waitTime/2);

cycleCnt = cycleCnt + 1;
 }
```

```
 /*
 BUG 6, End State Definition,              BUGGY
 Victoria Kim Dissertation

 Desired behavior:  All corners light together, flash on and off
 */

// Name the led pins on the Arduino.
int led6 = 6 ;
int led10 = 10 ;
int led17 = 17 ;
int led16 = 16 ;
int led19 = 19 ;
int waitTime = 1000;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pins as output.
  pinMode(led6, OUTPUT);
  pinMode(led10, OUTPUT);
  pinMode(led17, OUTPUT);
  pinMode(led16, OUTPUT);
  pinMode(led19, OUTPUT);

  digitalWrite(led16, HIGH);   // Set as positive, Keep pin positive
}

// loop routine
void loop() {
  digitalWrite(led10, LOW);   // turn the LED on
  digitalWrite(led19, LOW);   // turn the LED on
  digitalWrite(led16, LOW);   // turn the LED on
  digitalWrite(led6,  LOW);   // turn the LED on

  delay(waitTime);           // wait 1 second

  digitalWrite(led10, HIGH);  // turn the LED off
  digitalWrite(led19, HIGH);  // turn the LED off
  digitalWrite(led16, HIGH);   // turn the LED on
  digitalWrite(led6,  HIGH);   // turn the LED on
  }
```

```
/*
BUG 6, End State Definition,            CORRECTED
Victoria Kim Dissertation
Desired behavior:  All corners light together, flash on and off
*/
// Name the led pins on the Arduino.
int led6  = 6 ;
int led10 = 10 ;
int led17 = 17 ;
int led16 = 16 ;
int led19 = 19 ;
int waitTime = 1000;
// the setup routine runs once when you press reset:
void setup() {
// initialize the digital pins as output.
pinMode(led6, OUTPUT);
pinMode(led10, OUTPUT);
pinMode(led17, OUTPUT);
pinMode(led16, OUTPUT);
pinMode(led19, OUTPUT);
digitalWrite(led17, HIGH);   // Set as positive, Keep pin positive
}
// loop routine
void loop() {
digitalWrite(led10, LOW);   // turn the LED on
digitalWrite(led19, LOW);   // turn the LED on
digitalWrite(led16, LOW);   // turn the LED on
digitalWrite(led6,  LOW);   // turn the LED on
delay(waitTime);            // wait 1 second
digitalWrite(led10, HIGH);  // turn the LED off
digitalWrite(led19, HIGH);  // turn the LED off
digitalWrite(led16, HIGH);   // turn the LED on
digitalWrite(led6,  HIGH);   // turn the LED on
delay(waitTime);            // wait 1 second                ** Correction
 }
```

```
/*
BUG 7       BUGGY,       Multibug
Victoria Kim Dissertation

Desired Action: The Robot's arms and legs alternate, in pairs.
*/

// Name the led pins on the Arduino.
int led5 = 5 ;     //   Robot's right arm
int led6 = 6 ;     //   Robot's right leg
int led9 = 9 ;     //   Robot's left leg
int led10 = 10;     //  Robot's left arm

int waitTime = 1000;

// setup runs once
void setup() {
  // initialize the digital pins as an output.
  pinMode(led5, OUTPUT);
  pinMode(led6, OUTPUT);
  pinMode(led9, OUTPUT);
  pinMode(led10, OUTPUT);

  digitalWrite(led5, HIGH);   //start with pin off
  digitalWrite(led6, HIGH);    //start with pin off
  digitalWrite(led9, HIGH);    //start with pin off
  digitalWrite(led10, HIGH);    //start with pin off

}

// Loop routine (continually repeats)
void loop() {
  digitalWrite(led5, LOW);    // turn arms on
  digitalWrite(led10, LOW);   // turn arms on

  digitalWrite(led6, HIGH);   // turn legs off
  digitalWrite(led9, HIGH);   // turn legs off

  delay(waitTime);          // wait 1 seconds

  digitalWrite(led5, HIGH);   // turn arms off
  digitalWrite(led10, HIGH);  // turn arms off

  digitalWrite(led6, LOW);    // turn legs on
  digitalWrite(led9, LOW);    // turn legs on
  }
```

```
/*
 New Bug 7 CORRECTED, Multibug
 Victoria Kim Dissertation

 Desired Action: The Robot's arms and legs alternate, in pairs.
 */

// Name the led pins on the Arduino.
int led5 = 5 ;     //   Robot's right arm
int led6 = 6 ;     //   Robot's right leg
int led9 = 9 ;     //   Robot's left leg
int led10 = 10;     //  Robot's left arm  (neg)
int led11 = 11;     //  Robot's left arm  (pos)                 ***  bug fix

int waitTime = 1000;

// setup runs once
void setup() {
  // initialize the digital pins as an output.
  pinMode(led5, OUTPUT);
  pinMode(led6, OUTPUT);
  pinMode(led9, OUTPUT);
  pinMode(led10, OUTPUT);
  pinMode(led11, OUTPUT);              //                          ***  bug fix
  digitalWrite(led11, HIGH);   //set pin to positive             ***  bug fix
  digitalWrite(led5, HIGH);   //start with pin off
  digitalWrite(led6, HIGH);    //start with pin off               *** bug fix (sewing)
  digitalWrite(led9, HIGH);    //start with pin off
  digitalWrite(led10, HIGH);    //start with pin off
}

// Loop routine (continually repeats)
void loop() {
  digitalWrite(led5, LOW);    // turn arms on
  digitalWrite(led10, LOW);   // turn arms on
  digitalWrite(led6, HIGH);   // turn legs off
  digitalWrite(led9, HIGH);   // turn legs off

  delay(waitTime);         // wait 1 seconds

  digitalWrite(led5, HIGH);   // turn arms off
  digitalWrite(led10, HIGH);  // turn arms off
  digitalWrite(led6, LOW);    // turn legs on
  digitalWrite(led9, LOW);    // turn legs on

  delay(waitTime);         // wait 1 seconds                    *** bug fix
  }
```

APPENDIX L

Programming Activities Handout

Protosnap Programming Activity 1

Learning Goals: Experience the Arduino programming environment. Upload programs. Run programs. Learn how to place comments in code. Learn how to declare input or output type. Explore Arduino sample programs.

Tools & Materials:
Protosnap Kit (Protosnap board, FTDI connector), USB mini cord, Computer, Arduino software)

Description:
Begin with the Blink Program
Alternate Blinking LEDs

Figure L125. Tools needed for programming the LilyPad Protosnap board.

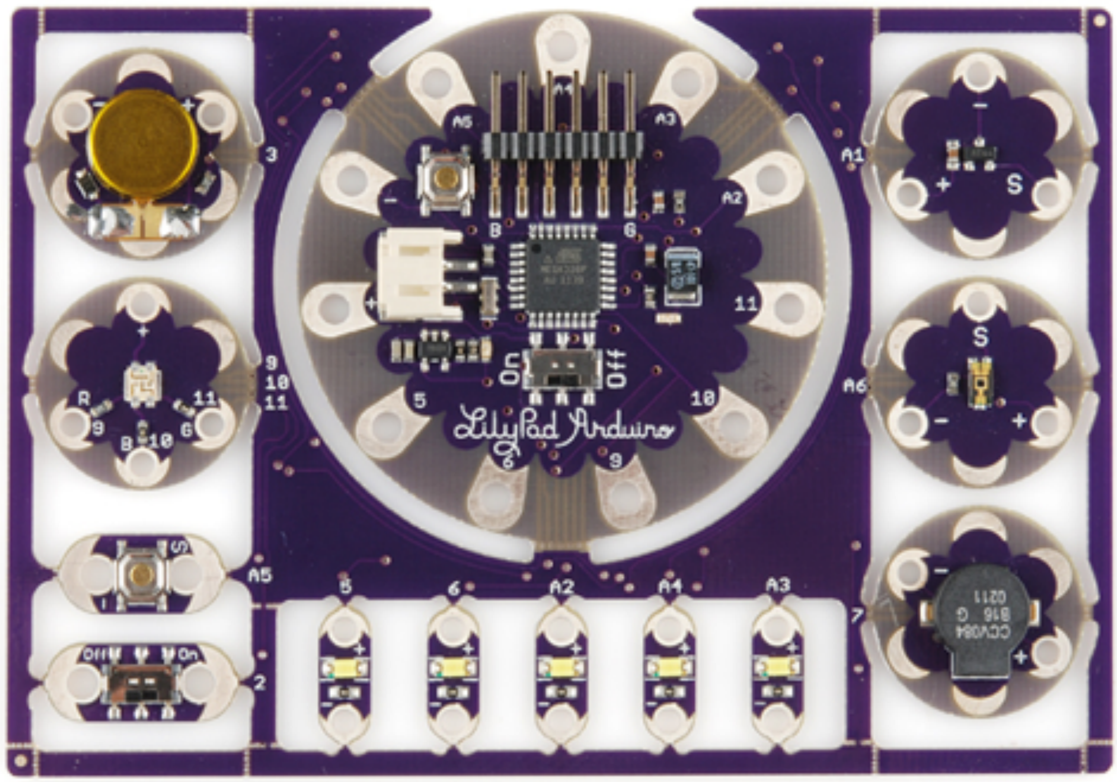


Figure L126. Close up of LilyPad Protosnap board.

The code in the Blink program below is an Arduino example, in the Basics folder. There are two ways to make comments. For one-line comments you can use //. For longer comments enclose the block between /* and */

```
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.
This example code is in the public domain.
*/
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;
// the setup routine runs once when you press reset:
void setup() {
// initialize the digital pin as an output.
pinMode(led, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(1000);               // wait for a second
digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
delay(1000);               // wait for a second
}
```

The code above in Blink makes the LED ON the LilyPad blink. We will change the code to make the LED attached to pin 5 blink and make the blink twice as fast. Changes are highlighted in yellow.

```
/*
Blink –Altered

Turns on an LED on for one second, then off for one second, repeatedly.
This example code is in the public domain.
*/
// Blink LED 5
//
int led = 5;
// the setup routine runs once when you press reset:
void setup() {
// initialize the digital pin as an output.
pinMode(led, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(1000/2);             // wait for half a second
digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
delay(1000/2);             // wait for half a second
}
```

We will change the code to make the LEDs attached to pin 5 and 6 alternate blinks. Changes are highlighted in yellow.

```
/*
Blink –Altered-2
Turns on an LED on for for half a second, then off for half a second, repeatedly.
LED6 alternates with LED5.
*/
// Blink LEDs 5 & 6
//
int led5 = 5;
int led6 = 6;
// the setup routine runs once when you press reset:
void setup() {
// initialize the digital pins as an output.
pinMode(led5, OUTPUT);
pinMode(led6, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
digitalWrite(led5, HIGH);    // turn the LED on (HIGH is the voltage level)
digitalWrite(led6, LOW);     // turn the LED off (LOW is the voltage level)
delay(1000/2);               // wait for half a second
digitalWrite(led5, LOW);     // turn the LED off by making the voltage LOW
digitalWrite(led6, HIGH);    // turn the LED on by making the voltage HIGH
delay(1000/2);               // wait for half a second
}
```

Open the Fade Example in the Arduino-Examples-Basics. Change the LED to 5.

```
/*
  Fade

  This example shows how to fade an LED on pin 5
  using the analogWrite() function.

  This example code is in the public domain.
  */

int led = 5;          // the pin that the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup()  {
  // declare pin 5 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop()  {
  // set the brightness of pin:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount ;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

Protosnap Programming Activity 2

Learning Goal: Understand the behavior of a conditional statement in the program. Add conditional to a program to produce desired behavior. Understand the behavior of a loop in the program. Alter the behavior of the loop.

Tools & Materials:
Protosnap Kit (Protosnap board, FTDI connector), USB mini cord, Computer, Arduino software)

Notes:
If statement
Loops

Description:

Alter program. Insert IF statement, Loop

Figure L127. Tools used for programming LilyPad Protosnap board.

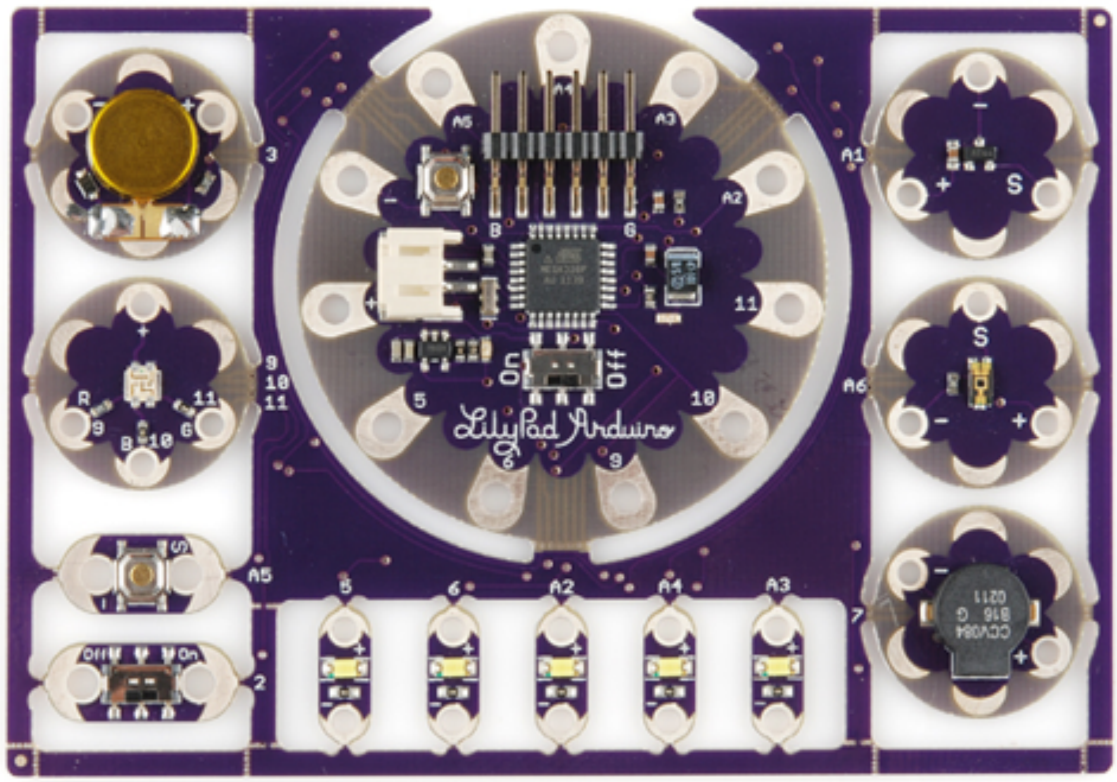


Figure L128. Close-up of the LilyPad Protosnap board.

Run Conditional Example. Note the use of == in the if statement.

```
/*
Conditional Example
If it is the first time through the loop, turn on LED 5,
otherwise turn on LED6.
*/
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int ledB = 13;
int led5 = 5;
int led6 = 6;
int startCount;
// the setup routine runs once when you press reset:
void setup() {
// initialize the digital pin as an output.
pinMode(ledB, OUTPUT);
pinMode(led5, OUTPUT);
pinMode(led6, OUTPUT);
startCount = 0;
}
// the loop routine runs over and over again forever:
void loop() {
if (startCount == 0)
{
digitalWrite(led5,HIGH);
}
else
{
digitalWrite(led6,HIGH);
}
digitalWrite(ledB, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(1000);               // wait for a second
digitalWrite(ledB, LOW);    // turn the LED off by making the voltage LOW
digitalWrite(led5,LOW);
digitalWrite(led6,LOW);
delay(1000);               // wait for a second
startCount = startCount + 1;
}
```

Protosnap Programming Activity 3

Learning Goal: Use the serial port to display program progress.

Tools & Materials:
Protosnap Kit (Protosnap board, FTDI connector), USB mini cord, Computer, Arduino software)

Notes:
Serial Output

Description:

Alter loop program to show serial output

Figure L129. Tools for programming LilyPad Protosnap board.

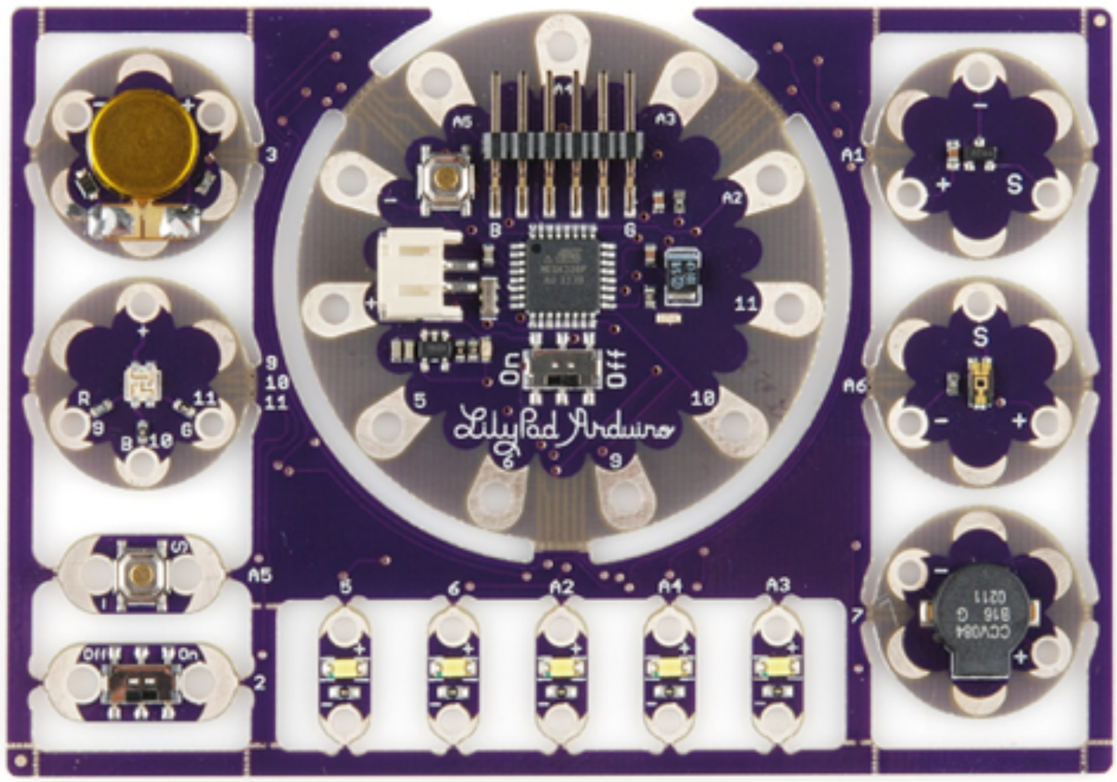


Figure L130. Close up of LilyPad Protosnap board.

We will change the Conditional example to output to the monitor. Save as ConditionalSerial. Note Serial.print vs. Serial.println. Changes are highlighted in yellow.

```
/*
Conditional Serial Example
If it is the first time through the loop, turn on LED 5,
otherwise turn on LED6.
Display output on serial monitor
*/
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int ledB = 13;
int led5 = 5;
int led6 = 6;
int startCount;
 // the setup routine runs once when you press reset:
void setup() {
// initialize the digital pins as an output.
pinMode(ledB, OUTPUT);
pinMode(led5, OUTPUT);
pinMode(led6, OUTPUT);
Serial.begin(9600);      // prepare to use serial monitor
startCount = 0;
}
// the loop routine runs over and over again forever:
void loop() {
Serial.print("startCount is  "); // note this is print not println
Serial.print(startCount);
if (startCount == 0)
{
digitalWrite(led5,HIGH);
Serial.println(", Blink Led 5");
}
else
{
digitalWrite(led6,HIGH);
Serial.println(", Blink Led 6");
}
digitalWrite(ledB, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(1000);              // wait for a second
digitalWrite(ledB, LOW);    // turn the LED off by making the voltage LOW
digitalWrite(led5,LOW);
digitalWrite(led6,LOW);
delay(1000);              // wait for a second
startCount = startCount + 1;
}
```

APPENDIX M

Scripts to Accompany Measures

Scripts to Accompany Measures for
Development of an E-Textile Learning Module to Increase
Computational Thinking Skills in Graduate Education Students

Script for Computer Experience Questionnaire

Thank you for agreeing to be part of the research for Victoria Kim's doctoral dissertation entitled Development of an E-Textile Learning Module to Increase Computational Thinking Skills in Graduate Education Students.

Your participation in the study is voluntary; you can withdraw at any time. You can choose to skip any question at your own discretion. Your responses will be kept confidential.

Neither your class grade nor your standing in the Doctor of Learning Technologies program will be affected if you decide not to participate or decide to withdraw from the research study.

The computer experience questionnaire that you are being asked to complete contains 12 questions that can be answered by selecting a number from 1 to 6 that corresponds to your level of agreement. The questionnaire should take approximately five minutes to complete.

Script for Computational Attitude Questionnaire

Thank you for agreeing to be part of the research for Victoria Kim's doctoral dissertation entitled Development of an E-Textile Learning Module to Increase Computational Thinking Skills in Graduate Education Students.

Your participation in the study is voluntary; you can withdraw at any time. You can choose to skip any question at your own discretion. Your responses will be kept confidential.

Neither your class grade nor your standing in the Doctor of Learning Technologies program will be affected if you decide not to participate or decide to withdraw from the research study.

The computational attitude questionnaire that you are being asked to complete contains 20 questions, broken up into four categories. The questions can be answered by selecting a number from 1 to 6 that corresponds to your level of agreement with the contrasting word on either side. This type of question is known as a semantic pair, and the words represent two ends of a spectrum. The questionnaire should take approximately five to ten minutes to complete.

Script for E-Textile Self-Efficacy Questionnaire

Thank you for agreeing to be part of the research for Victoria Kim's doctoral dissertation entitled Development of an E-Textile Learning Module to Increase Computational Thinking Skills in Graduate Education Students.

Your participation in the study is voluntary; you can withdraw at any time. You can choose to skip any question at your own discretion. Your responses will be kept confidential.

Neither your class grade nor your standing in the Doctor of Learning Technologies program will be affected if you decide not to participate or decide to withdraw from the research study.

The e-textile self-efficacy questionnaire that you are being asked to complete contains 11 questions that can be answered by selecting a number from 1 to 6 that corresponds to your level of agreement. The questions are similar, varying only the support method that is referred to. The questionnaire should take approximately five minutes to complete.

Script for Six Debugging Design Scenarios

Thank you for agreeing to be part of the research for Victoria Kim's doctoral dissertation entitled Development of an E-Textile Learning Module to Increase Computational Thinking Skills in Graduate Education Students.

Your participation in the study is voluntary; you can withdraw at any time. You can choose to skip any question at your own discretion. Your responses will be kept confidential.

Neither your class grade nor your standing in the Doctor of Learning Technologies program will be affected if you decide not to participate or decide to withdraw from the research study.

If you and your partner agree, your audio interactions with your partner will be recorded and later examined using a rubric that looks for evidence of computational thinking concepts, practices, and perspectives. You may still participate in the research study, even if you choose not to be recorded.

On the Consent for Participation form you indicated whether or not you give permission for your picture to be taken while working on the design scenario.

The design scenario activities are part of the course requirements and will take place over two days. Each day you will spend approximately 2 ½ hours in the design scenario activity.

Script for Group Interview

Thank you for agreeing to be part of the research for Victoria Kim's doctoral dissertation entitled Development of an E-Textile Learning Module to Increase Computational Thinking Skills in Graduate Education Students.

Your participation in the study is voluntary; you can withdraw at any time. You can choose to skip any question at your own discretion. Your responses will be kept confidential.

Neither your class grade nor your standing in the Doctor of Learning Technologies program will be affected if you decide not to participate or decide to withdraw from the research study.

This group interview will be recorded and later examined using a rubric that looks for evidence of computational thinking concepts, practices, and perspectives. Your opinions will also be used to improve future implementations of the learning module.

APPENDIX N

IRB Approval Letter

PEPPERDINE UNIVERSITY

Graduate & Professional Schools Institutional Review Board

August 13, 2013

Victoria Kim
[redacted]

Protocol #: E0713D02
Project Title: Development of an E-Textile Debugging Module to Increase Computational Thinking
Among Graduate Education Students

Dear Ms. Kim,

Thank you for submitting your application, *Development of an E-Textile Debugging Module to Increase Computational Thinking Among Graduate Education Students,* for exempt review to Pepperdine University's Graduate and Professional Schools Institutional Review Board (GPS IRB). The IRB appreciates the work you and your faculty advisor, Dr. Linda Polin, have done on the proposal. The IRB has reviewed your submitted IRB application and all ancillary materials. Upon review, the IRB has determined that the above entitled project meets the requirements for exemption under the federal regulations (45 CFR 46 - http://www.nihtraining.com/ohsrsite/guidelines/45cfr46.html) that govern the protections of human subjects. Specifically, section 45 CFR 46.101(b)(1) states:

> (b) Unless otherwise required by Department or Agency heads, research activities in which the only involvement of human subjects will be in one or more of the following categories are exempt from this policy:
>
> **Category (1) of 45 CFR 46.101**, This research will be conducted in an established educational setting and will involve normal educational practices. The research will be about the effectiveness of instructional techniques and curricula.

Your research must be conducted according to the proposal that was submitted to the IRB. If changes to the approved protocol occur, a revised protocol must be reviewed and approved by the IRB before implementation. For any proposed changes in your research protocol, please submit a **Request for**

Modification Form to the GPS IRB. Because your study falls under exemption, there is no requirement for continuing IRB review of your project. Please be aware that changes to your protocol may prevent the research from qualifying for exemption from 45 CFR 46.101 and require submission of a new IRB application or other materials to the GPS IRB.

A goal of the IRB is to prevent negative occurrences during any research study. However, despite our best intent, unforeseen circumstances or events may arise during the research. If an unexpected situation or adverse event happens during your investigation, please notify the GPS IRB as soon as possible. We will ask for a complete explanation of the event and your response. Other actions also may be required depending on the nature of the event. Details regarding the timeframe in which adverse events must be reported to the GPS IRB and the appropriate form to be used to report this information can be found in the *Pepperdine University Protection of Human Participants in Research: Policies and Procedures Manual* (see link to "policy material" at http://www.pepperdine.edu/irb/graduate/).

Please refer to the protocol number denoted above in all further communication or correspondence related to this approval. Should you have additional questions, please contact Veronica Jimenez, GPS IRB

Manager at gpsirb@peppderdine.edu.

On behalf of the GPS IRB, I wish you success in this scholarly pursuit.

Sincerely,

Doug Leigh, Ph.D.
Chair, Graduate and Professional Schools IRB

cc: Dr. Lee Kats, Vice Provost for Research and Strategic Initiatives
Ms. Alexandra Roosa, Director Research and Sponsored Programs
Dr. Linda Polin, Graduate School of Education and Psychology