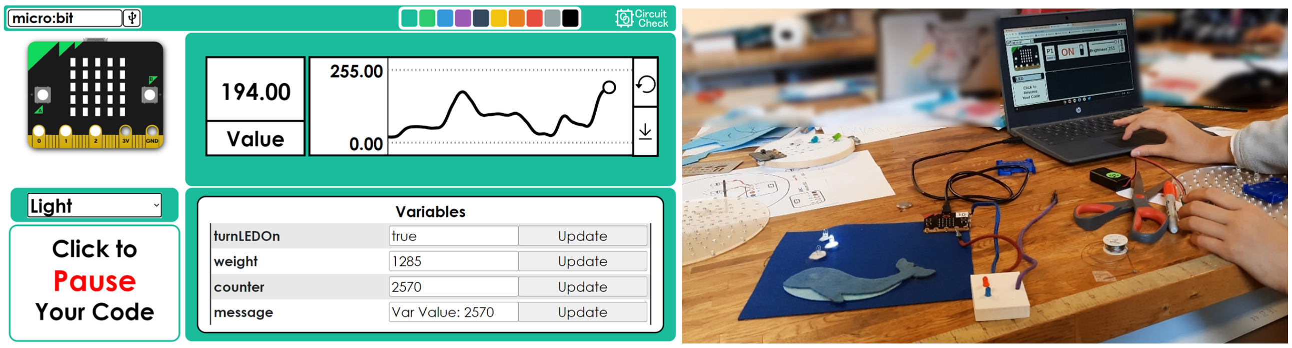
Designing scaffolds to support students in debugging e-textiles

Michael Schneider
Michael.J.Schneider@colorado.edu
University of Colorado Boulder
Boulder, Colorado, USA



Variables

turnLEDOn	true	Update
weight	1285	Update
counter	2570	Update
message	Var Value: 2570	Update

Figure 1: Circuit Check's User Interface (left) and a student debugging an e-textile project with Circuit Check (right)

ABSTRACT

My doctoral research focuses on the design of tools that scaffold the debugging process for students crafting e-textiles, a type of physical computing where circuits are woven together with conductive thread and textile fabrics. While this can be a creative medium for children to learn and experience computing, they struggle with locating errors in this mixed hardware/software environment - is the LED not turning on due to a fault in the circuit, an issue within the code, or some combination of the two? To address this issue, my study will follow a Design-Based Research approach to investigate and iterate on the design of debugging scaffolds. My primary scaffold is Circuit Check, an interactive web-based debugger that enables students to easily observe and test their hardware components. Preliminary findings from classroom observations have shown both the strong need for, and benefits of, Circuit Check's approach of supporting debugging through system exploration.

CCS CONCEPTS

• **Applied computing** → **Education**; • **Software and its engineering** → **Software testing and debugging**.

KEYWORDS

debugging, e-textiles, physical computing, educational scaffolds

ACM Reference Format:
Michael Schneider. 2023. Designing scaffolds to support students in debugging e-textiles. In *Interaction Design and Children (IDC '23), June 19–23, 2023, Chicago, IL, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3585088.3593925

1 INTRODUCTION

Traditionally, students learn computer science topics through designing and programming software within the virtual space of their computer, but there has been growing interest in teaching students programming through crafting physical computing systems [5, 9]. Topics such as conditional logic and loops are still covered, but instead of interacting with digital artifacts on their screen, a student creates a project that senses and interacts with the physical world. A unique style of physical computing can be found in e-textiles where instead of designing and constructing circuits with insulated wires and breadboards, students sew electronic components into fabrics with conductive thread[4]. While blending textile crafting and electrical engineering enables students to create meaningful projects, these e-textile projects can be tricky to debug [10]. Debugging, the act of finding and fixing bugs (or errors), has long been a challenge for students learning to program [12] but that challenge is intensified in physical computing, where students struggle to locate errors that can occur not only in their software but also in their hardware [3, 5, 10]. Part of the the reason why it is so difficult for students to locate errors comes from the naive approaches they often employ in debugging.

Inexperienced students often apply a trial-and-error approach to debugging where they will make changes to their code as they evaluate it line-by-line [16]. But with these changes comes the possibility of introducing new bugs [14], which must then also be fixed. Now making mistakes while debugging is not necessarily a bad thing. Mistakes are a normal part of the learning process and can help highlight student misconceptions and provide an opportunity to grow their debugging skills [12]. But this approach can become problematic when a student loses track of haphazardly made edits, becomes overwhelmed with the debugging process, and then stops trying to debug and instead wait for someone else to fix the bug for them [14]. While some amount of struggle or difficulty can be beneficial for learning [2], a balance must be maintained to ensure that the amount of struggle a student experiences does not lead to anxiety and/or abandonment of their project [1]. Researchers have proposed different approaches to support students in overcoming the challenges of debugging, from explicitly teaching debugging strategies [13] to novel interactive software debugging tools [11]. But most of these interventions were designed around the needs of students learning to write code, not those working to create with physical computing systems like e-textiles [10]

Few tools are designed for developing and debugging e-textile circuits [15] and while software tools for physical computing can be adapted to e-textiles they are either designed for professional engineers [6], or are simple but difficult, initially, for students to correctly implement (e.g. proper placement of print statements or selection of code to comment-out [7]). The paucity of tools is problematic because in order to debug their e-textile project, a student must be able to test and observe its components (e.g. sensors, actuators, and variables). I have addressed this need by creating Circuit Check, a web-based debugging tool that enables students to easily observe and test their e-textile components while also making it easier for teachers to guide their students through the debugging process. The design and evaluation of Circuit Check has been informed by qualitative research techniques, coupled with informal observations of students crafting e-textiles and formal classroom observations of teachers supporting students in crafting e-textile projects.

2 RESEARCH QUESTIONS

RQ1 What are the design considerations for debugging tools in the domain of e-textiles for supporting student exploration of system behavior?

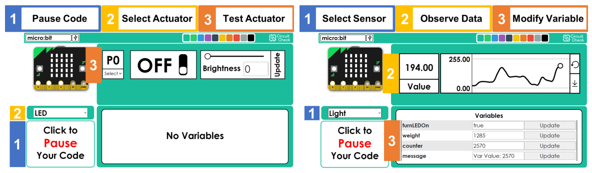
RQ2 What resources do teachers require to facilitate the debugging process with their students, within physical computing?

3 INTERVENTION

When debugging, students often struggle with gathering the necessary information to understand what is happening within their e-textile system - what is the current reading for their light sensor or is the microcontroller providing power to a given LED? Traditionally, in order to find this information a student would need to modify their project's code by either injecting new lines of code (e.g. print statements to observe a sensor) or by removing existing lines of their code (e.g. commenting out blocks of code to isolate and test specific components). However, these traditional techniques are not ideal for students or their teachers. Guiding a student who is new to programming and physical computing through the necessary steps of traditional debugging techniques can not only be time consuming, but anytime modifications are introduced, there is the potential to create new bugs [14]. These new bugs will in turn need to be resolved, which only further increases the difficulty of fixing the original bug. Circuit Check, Figure 2, has been designed to address this issue by enabling students to observe live sensor readings and test hardware components without requiring any code modifications. Instead, Circuit Check's debugging features are embedded into the student's compiled program through a provided library (Arduino) or extension (MakeCode). This means that at any point in the development process, from initial draft to finalized project, the student can evaluate their system with Circuit Check.

3.1 Debugging Scenario

Early projects students design often use an LED to act as a signal, where the student's program will turn the LED on or off based on some condition - for example, a specific sensor value or range of values being reached. To figure out why an LED won't turn on, a student must check for the presence of hardware and software faults. To check for hardware faults, a student can pause their running program, isolating the hardware from the software, and test if the LED is physically able to turn ON (Figure 2 (Left)). To check for logical errors in their software, a student can use Circuit Check to observe a live sensor reading. As shown in Figure 2 (Right), a student can select any of their system's sensors and observe a live readout of its data. In addition to viewing sensor data, the student can watch and update their project's variables - monitoring for changes after each iteration of their forever loop or testing the system by setting a variable to a specific value.



Variables		
turnLEDOn	true	Update
weight	1285	Update
counter	2570	Update
message	Var Value: 2570	Update

Figure 2: Screenshots of Circuit Check's User Interface - (Left) Testing an actuator to check for hardware errors, (Right) Observing live sensor data and modifying variables to check for logical errors in the software.

4 METHODOLOGY

The conceptual framework guiding my dissertation is grounded in Dr. Warshauer's Productive Struggles Framework (PSF) [17], a framework for qualitatively analyzing teacher-student interactions during moments of struggle in K-12 mathematics classrooms. Due to key differences in the style of student-teacher interactions in math versus computer science classrooms, PSF cannot be directly mapped to the context of debugging. For example, when a teacher supports a student struggling to answer a math problem the teacher typically knows what solution they are working towards. In contrast, when supporting a student struggling with debugging the teacher often does not know what bug or error the student needs to find and fix. This in turn affects the type of struggles and responses that will occur in a debugging interaction. In a related research project [8], I have assisted in analyzing the pedagogical moves teachers use to support their students in debugging physical computing systems. I plan to use this study to adapt the Productive Struggles Framework for debugging, where instead of focusing on the student's cognitive load the focus will be on how a teacher's responses and actions support *debugging with* or *debugging for* a student. By *debugging for* a student, a teacher can help a student quickly overcome a challenging bug but this comes at the cost of the student's growth in debugging - they cannot learn to debug if they are not given the opportunity to work through the debugging process. To understand how student-teacher interactions during debugging fit within this modified framework, I plan to collect and analyze data from classroom observations and professional development workshops with teachers, both will focus on creating and debugging e-textiles.

5 RESEARCH TIMELINE

My dissertation consists of three observational studies, which focus on (1) e-textile summer camp programs, (2) e-textiles integrated into STEM classrooms, and (3) professional development workshops for teachers covering e-textile curriculum.

Summer Camp Programs - Completed - RQ1

Summer '21 and '22: During the summer camp programs I taught middle school aged students to craft e-textile projects, which covered the basics of circuitry, sewing, and programming. Serving as the teacher, I helped the campers with debugging their projects by using early prototypes of Circuit Check. From my analysis of the transcripts of our debugging sessions, my personal notes, and images taken of their projects, I found that the most pressing need for the campers was to explore. Everything, from coding to sewing, was new to them and they needed the opportunity to play with their hardware components and see what they could do. Based on this, I simplified both Circuit Check's UI and its supporting library to better ease students into exploration while debugging and reduce the amount of time spent learning how to use Circuit Check.

STEM Classrooms - In Progress - RQ1, RQ2

Fall 2022: I co-designed e-textile curriculum with two high school STEM teachers that incorporated Circuit Check for debugging. I collected observational data during its implementation where the teachers taught the material and I served as an additional support for debugging. The data I collected includes video recordings of each class period, audio recordings from the teacher and myself, photos of student projects, and video recordings of my closing interviews with each teacher.

Spring, Summer 2022: I have not yet analyzed the recorded data. I am working on a plan with my adviser and dissertation committee for analyzing the data and plan to complete my analysis by the end of summer 2022.

Professional Development - Not Commenced - RQ2

Summer, Fall 2023: For the final stage of my dissertation I plan to create and run professional development workshops to train K-12 teachers both in how to create e-textile projects and integrate e-textiles into their existing curriculum. I plan to cover debugging strategies and how to use Circuit Check to guide students through the debugging process.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1742081. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Ashok R Basawapatna, Alexander Repenning, Kyu Han Koh, and Hilarie Nickerson. 2013. The zones of proximal flow: guiding students through a space of computational thinking skills and challenges. In *Proceedings of the ninth annual international ACM conference on International computing education research.* 67–74.

[2] Elizabeth L Bjork and Robert A Bjork. 2011. Making things hard on yourself, but in a good way: Creating desirable difficulties to enhance learning. *Psychology and the real world: Essays illustrating fundamental contributions to society* 2, 59-68 (2011).

[3] Tracey Booth, Simone Stumpf, Jon Bird, and Sara Jones. 2016. Crossed wires: Investigating the problems of end-user developers in a physical computing task. In *Proceedings of the 2016 CHI conference on human factors in computing systems.* 3485–3497.

[4] Leah Buechley, Kylie Peppler, Michael Eisenberg, and Kafai Yasmin. 2013. *Textile Messages: Dispatches from the World of E-Textiles and Education. New Literacies and Digital Epistemologies. Volume 62.* ERIC.

[5] Kayla DesPortes and Betsy DiSalvo. 2019. Trials and tribulations of novices working with the Arduino. In *Proceedings of the 2019 ACM Conference on International Computing Education Research.* 219–227.

[6] Jan Dolinay, Petr Dostálek, and Vladimír Vašek. 2021. Advanced debugger for Arduino. *International Journal of Advanced Computer Science and Applications* (2021).

[7] Sue Fitzgerald, Gary Lewandowski, Renee McCauley, Laurie Murphy, Beth Simon, Lynda Thomas, and Carol Zander. 2008. Debugging: finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education* 18, 2 (2008), 93–116.

[8] Colin Hennessy Elliott, Alexandra Gendreau Chakarov, Jeffrey B Bush, Jessie Nixon, and Mimi Recker. 2023. Toward a debugging pedagogy: helping students learn to get unstuck with physical computing systems. *Information and Learning Sciences* 124, 1/2 (2023), 1–24.

[9] Steve Hodges, Sue Sentance, Joe Finney, and Thomas Ball. 2020. Physical computing: A key element of modern computer science education. *Computer* 53, 4 (2020), 20–30.

[10] Gayithri Jayathirtha, Deborah Fields, and Yasmin Kafai. 2018. Computational concepts, practices, and collaboration in high school students' debugging electronic textile projects.. In *Conference Proceedings of International Conference on Computational Thinking Education 2018.*

[11] Amy J Ko and Brad A Myers. 2008. Debugging reinvented: asking and answering why and why not questions about program behavior. In *Proceedings of the 30th international conference on Software engineering.* 301–310.

[12] Renee McCauley, Sue Fitzgerald, Gary Lewandowski, Laurie Murphy, Beth Simon, Lynda Thomas, and Carol Zander. 2008. Debugging: a review of the literature from an educational perspective. *Computer Science Education* 18, 2 (2008), 67–92.

[13] Tilman Michaeli and Ralf Romeike. 2019. Improving debugging skills in the classroom: The effects of teaching a systematic debugging process. In *Proceedings of the 14th workshop in primary and secondary computing education.* 1–7.

[14] David N Perkins, Chris Hancock, Renee Hobbs, Fay Martin, and Rebecca Simmons. 1986. Conditions of learning in novice programmers. *Journal of Educational Computing Research* 2, 1 (1986), 37–55.

[15] Irene Posch and Geraldine Fitzpatrick. 2021. The matter of tools: designing, using and reflecting on new tools for emerging eTextile craft practices. *ACM Transactions on Computer-Human Interaction (TOCHI)* 28, 1 (2021), 1–38.

[16] Iris Vessey. 1985. Expertise in debugging computer programs: A process analysis. *International Journal of Man-Machine Studies* 23, 5 (1985), 459–494.

[17] Hiroko Kawaguchi Warshauer. 2015. Productive struggle in middle school mathematics classrooms. *Journal of Mathematics Teacher Education* 18 (2015), 375–400.