# CircuitStyle: A System for Peripherally Reinforcing Best Practices in Hardware Computing

**Josh Urban Davis[1,*], Jun Gong[1,*], Yunxin Sun[1,3], Parmit Chilana[2], Xing-Dong Yang[1]**
Dartmouth College[1], Simon Fraser University[2], Tongji University[3]
{josh.u.davis.gr, jun.gong.gr, xing-dong.yang}@dartmouth.edu, pchilana@cs.sfu.ca,
martin.yunxin.sun@gmail.com

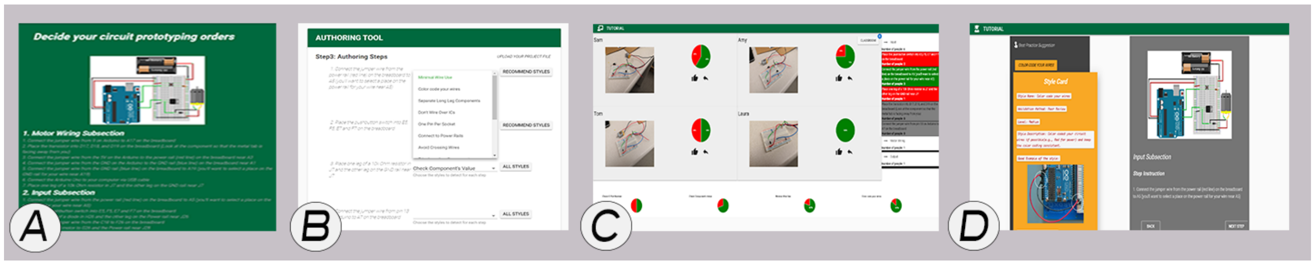


**Figure 1. Overview of the *CircuitStyle* interface. (A) Students sort steps of the tutorial; (B) Style authoring tool; (C) Instructor interface for at-a-glance awareness of classroom performance; (D) Student in-situ tutorial interface with dismissible style guide.**

## ABSTRACT
Instructors of hardware computing face many challenges including maintaining awareness of student progress, allocating their time adequately between lecturing and helping individual students, and keeping students engaged even while debugging problems. Based on formative interviews with 5 electronics instructors, we found that many circuit style behaviors could help novice users prevent or efficiently debug common problems. Drawing inspiration from the software engineering practice of coding style, these circuit style behaviors consist of best-practices and guidelines for implementing circuit prototypes that do not interfere with the functionality of the circuit, but help a circuit be more readable, less error-prone, and easier to debug. To examine if these circuit style behaviors could be peripherally enforced, aid an in-person instructor's ability to facilitate a workshop, and not monopolize instructor's attention, we developed *CircuitStyle*, a teaching aid for in-person hardware computing workshops. To evaluate the effectiveness of our tool, we deployed our system in an in-person maker-space workshop. The instructor appreciated *CircuitStyle*'s ability to provide a broad understanding of the workshop's progress and the potential for our system to help instructors of various backgrounds better engage and understand the needs of their classroom.

ACM ISBN 978-1-4503-6816-2/19/10...$15.00
https://doi.org/10.1145/3332165.3347920

## Author Keywords
Software learning; real-time teaching assistance; hardware.

## INTRODUCTION
In breadboard circuit prototyping, circuit style (akin to coding style in programming) refers to a set of rules that uniforms the appearance and construction process of a breadboard circuit to make it readable, understandable, and maintainable (Figure 2). Examples of good circuit style behaviors include avoiding crossed wires while prototyping, using as little wire as possible, and checking the polarity of components before insertion. Practicing good circuit style behaviors results in breadboard circuits that are less error-prone, easier to debug, and easier to share.

Traditionally, breadboard circuit style has only been taught in universities or colleges to students pursuing a degree in electronics or related fields. However, increasingly, many novice and untra[*]ined users, such as in maker communities, are experimenting with breadboard prototyping on their own to incorporate electronics into art projects. In addition, the formal education backgrounds of high school electronics teachers, maker instructors, and workshop facilitators has also broadened beyond the traditional background of electronic engineering [4]. Unfortunately, the majority of tutorials and teaching materials available to these new learners and instructors consists of lessons on traditional electronics prototyping and focuses less enforcing appropriate circuit styles [25].

Given the increasing diversity and evolving needs of both educators and novice users in the hardware computing community, there is need for lightweight learning tools that

[*] Indicates co-first-authorship

support users from non-traditional backgrounds and promote useful habits for electronic prototyping.

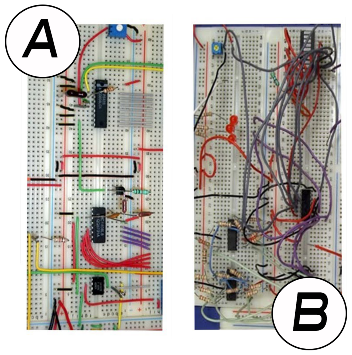


**Figure 2: (A) Example of good circuit style: wires go around ICs instead of over, components lay flat against board, etc. (B) Example of poor circuit style: wires long and tangled, components crammed together, etc.**

Using a user-centered approach, we first conducted semi-structured interviews with 5 instructors from various backgrounds to understand the current practice of teaching breadboard circuit style and any challenges the instructors face. From these insights, we assessed the importance of circuit styles and constructed a compiled list of common physical computing best behaviors (Table 1), and weighed the importance of the individual behaviors within this list of circuit styles. Although the instructors agreed that enforcing circuit style behaviors was an important aspect of an electronics education, most instructors did not have the time nor the ability to encourage and reinforce these behaviors for student individually.

To address the challenges delineated by the instructors, we created *CircuitStyle*, a workshop management tool to help instructors construct hardware prototyping tutorials for in-person workshops, keep track of participant behavior, and peripherally reinforce good circuit style behaviors without monopolizing the instructor or participant's attention (Figure 1). Unlike other workshop or classroom management tools, this paper focuses on the *peripheral* applicability of circuit styles to follow-along in-person workshop tutorials. We provide a list of circuit styles extracted from literature and interviews with workshop instructors. In addition, our system supports several features which peripherally reinforce these styles for students. These tools aim to encourage classroom engagement and offload style reinforcement to the software and student interaction.

For our evaluation, we deployed our system to a makerspace workshop-like environment and conducted a field study to evaluate the effectiveness of our system. We found that our system helped the instructor better engage with their students by reducing the amount of attention monopolized by tracking student progress and reminding students of common behaviors. The instructor also appreciated the tutorial authoring tool and its ability to deepen the instructors understanding of the course material and better anticipate common errors their students may encounter. In addition, workshop participants agreed that the tutorial system was helpful for navigating the process of circuit implementation and receiving circuit style reminders was helpful for debugging their circuit.

The main contributions of this paper are: 1) insights into the current challenges of teaching breadboard circuit style in makerspace workshops; 2) a classroom management system to help workshop instructors peripherally enforce circuit style behaviors without interfering with participants' learning of functional circuit construction or monopolizing instructor's attention; and 3) insights from a case study investigating users' initial impressions of the system's usability and usefulness. We discuss several insights for future research in HCI to better support style behaviors for electronics prototyping.

## RELATED WORK

This work builds upon existing research in the design of classroom management and learning tools, circuit prototyping tools, and insights from teaching coding styles.

### Classroom Management and Learning Tools

Several commercial products already exist which give an instructor access to the activity of their students' screens (e.g., *Softlink* and *NetSupport* School). These systems make the instructor aware of each student's activity on the computer and provide the instructor with coarse intervention options, such as freezing a student's input, or taking control over their computer. Another class of tools in research also provides general support for coordination in the classroom. For example, *GroupScribbles* [19] extends the concept of sticky notes to digital classroom media. *FireFlies2* supports cognitive offloading through the use of tangible pixel devices distributed through the classroom [34]. In contrast to these tools, our system provides contextual information about students' activity in a specific hardware skill being taught. The idea is to help instructors with early detection of potential problems, and to develop a series of robust "best-practices" to prevent errors.

Closely relevant to our project is *Maestro* designed for in-person 3D modeling tutorials [15, 16] where the tutor sees a dashboard displaying each learner's editor and can assess their progress. Our system takes inspiration from Maestro's approach, but examines a different and largely unexplored domain: hardware prototyping. This alters the problem in several key ways (1) monitoring student progress is difficult because it requires knowledge of a physical object being used by the student, and not a virtual environment or system, (2) the system is meant to be used as a support device that peripherally re-enforces good practices instead of providing the principle means of learning the material, and (3) support tools should ambiently aid the instructor, and students, not monopolize their attention.

Finally, a number of software learning systems have used data from software logs to enhance software tutorials [18, 28], or provide improved help or capabilities within featurerich software [6, 11, 15]. These projects, however, focus on an individual learning or using the software on their own. We are interested in exploring software systems that support learning for hardware computing in group settings, a topic that has only received limited attention [10, 20].

### Circuit Prototyping Tools

Prior work has shown that novice users face substantial difficulty in designing and building physical computing systems [4, 25]. Some challenges include choosing correct components, wiring components together, programming logic, specifying variable nomenclature, and debugging.

Several research systems have been developed to address these challenges. For example, *Toastboard* [9] is an intelligent breadboard that assists novices with debugging through LED indicators on the board itself, and a software interface that provides troubleshooting tips. *Bifröst* [24] instruments both the hardware and software components of embedded computing projects to help users trace the system state and assists in debugging. *Trigger-Action Circuits* [2] enables users to specify desired functionality at a behavioral level, and generates designs and corresponding instructions for assembling them. *PICL* [12] allow users to create sensor-based interactive systems using “programming by demonstration” (i.e. using demos to view actions and modifying them for use). Other systems teach fundamental concepts of circuit design, and programming. For example, *Programmable Bricks* [30] allows children to develop electronic hardware using LEGO bricks embedded with computers, sensors, and actuators. *ElectroTutor* approaches this problem by integrating interactivity into traditional step-by-step tutorials for hardware prototyping on the Arduino [35]. Finally, a number of systems have been developed that aid in sensing the state of the electronics components in embedded systems [9, 33, 36], data which could aid in debugging and troubleshooting.

Unlike the systems focused on developing novel hardware and sensing techniques or improving individual instruction, our work examines how circuit style practices can be communicated to novices and reinforced peripherally. We use the metaphor of software coding style as inspiration for developing our system. We also examine this problem from the instructor’s perspective and our approach supplements the in-person mentorship provided by an instructor.

### Teaching Coding Style

Evidence in the literature suggests that effectively teaching and enforcing coding style in programming (e.g., indentation, whitespace, naming conventions, etc.) significantly mitigates the number of bugs in a programmer’s code, preemptively prevent programmers from making common errors, and promotes the readability of the code [3]. Although coding style has little effect on the program’s behavior, it does have a significant influence on sustainability and readability for developers [5,26]. The choice of coding style is largely a matter of developer preference and evolves from their programming experience [27]. Although compliance with coding standards across an institution or project team can enhance team communication, reduce program errors, and improve overall code quality [1,13], developers and students do not consistently follow such conventions [22].

Another class of tools assist in enforcing good coding styles. For example, *Foobaz* is a tool that allows educators to provide custom feedback to students on variable names at scale [14]. Similarly, *PeerStudio* allows students to receive feedback from fellow students, reducing wait-time for help and improved learning [17]. *Style Avatar* visualizes student’s source code style as facial expressions to peripherally reinforce programming concepts [23]. *AutoStyle* is a research system that provides automated, adaptive style hints which suggest syntax shortcuts and code skeletons that enforce good coding style [7].

The above systems show that enforcing various aspects of good coding style can improve readability, portability, and maintainability of code while reducing the rate of error. This early experience of reading quality code and experiencing less frustration while debugging is especially crucial for novice users [22]. To better understand if this style behavior could be equally useful within the domain of physical computing, we considered the design space of how style suggestions could be integrated peripherally into circuit prototype training in a group setting.

## FORMATIVE STUDY AND DESIGN CONSIDERATIONS

Based on the above literature review, our first goal was to understand instructors’ current use of hardware computing style protocols when teaching novice users.

### Procedure

We devised a semi-structured interview protocol and recruited 5 instructors who taught electronics prototyping at various institutions including formal primary education classrooms, makerspace workshops, and higher education. We examined common teaching tools used, difficulties instructors faced when facilitating in-person circuit tutorials, and common style behaviors they repeatedly reinforced to their students. We also presented the instructors with a list of potential stylistic choices for hardware computing and asked the instructors to rank their importance on a 7-point Likert scale (Table 1).

### Participants

The instructors in our study had a variety of backgrounds including Visual Art, Physics, Electrical Engineering, and English. Some had very little experience in teaching hardware computing before they were asked to begin conducting tutorials on circuit prototyping. This was very surprising to us, so we asked the instructors to elaborate further how they learned the material they taught in their workshops or classes. They reported that often they learned the material while preparing for their lecture, often completing the circuit themselves the night before class. In this way, some of the instructors learned some of the material in tandem with their students.

| Circuit Styles | Avg. Severity (1-7) |
|---|---|
| Ensure component's polarity is correct before insertion (e.g. batteries, LEDs) | 7.00 (σ=0.00) |
| Avoid changing components on a breadboard whenever the board is powered. | 6.88 (σ=0.22) |
| Measure the component's value (resistance/ capacitance/ inductance) before insertion. | 6.63 (σ=0.41) |
| Check IC part number before insertion. | 6.50 (σ=0.61) |
| Build and test in subsections. | 6.50 (σ=0.87) |
| Begin by placing the ICs first. Then connect relevant components which directly reach out from the IC pins. | 6.38 (σ=0.54) |
| Always check the whole circuit and connect the power supply to the circuit last | 6.25 (σ=1.30) |
| Verify the power supply voltages and input signals with an oscilloscope or voltmeter. | 6.13 (σ=0.54) |
| Use the power rails to connect power supply. | 6.13 (σ=0.74) |
| Push the component down firmly until it cannot go any further. | 5.88 (σ=0.74) |
| Avoid laying wires or components over ICs. | 5.88 (σ=1.14) |
| Use as little wire as possible. | 5.63 (σ=0.96) |
| Ensure wires/components are trimmed to lay flat against the breadboard. | 5.50 (σ=1.06) |
| Don't insert two pins of different components or two wires into the same socket. | 5.50 (σ=2.06) |
| Keep pin 1 of all IC's pointing the same direction. | 5.38 (σ=1.47) |
| If more than one IC is involved, make sure they are separated by several rows. | 5.25 (σ=1.03) |
| Avoid placing two components with long legs close to each other. | 5.00 (σ=1.46) |
| Run a circuit simulation before building. | 4.50 (σ=1.12) |
| Understand the breadboard connection (Watch out for split power rails) before you start. | 4.50 (σ=1.50) |
| Color code the wires of your circuit (e.g., red for power, black for ground). | 4.50 (σ=1.80) |
| Carefully check the component's row and column number before inserting into board. | 4.13 (σ=1.75) |
| Avoid cramming components into compact areas; use the whole breadboard space uniformly. | 3.88 (σ=1.88) |
| Keep the relative position of components as similar to the diagram as possible.. | 3.75 (σ=1.89) |
| Avoid crossing wires. | 3.38 (σ=1.85) |
| Begin by connecting power and ground rails. | 3.25 (σ=1.79) |
| Bend each wire at 90°. | 3.00 (σ=2.03) |
| Use software to plan the breadboard circuit first. | 2.63 (σ=1.71) |

**Table 1. Compiled list of circuit styles aggregated from instructor interviews with average importance score associated with each style.**

## Results

Most of the instructors agreed that circuit style behaviors were important to learn, but difficult to teach because they required repeated reinforcement. Most of the instructors, for example, mentioned the need to repeatedly remind students to check the polarity of various components before inserting the component into their breadboard, or to use a multimeter to ensure their component is working properly. The interviewed instructors agreed that enforcing these behaviors was important, but doing so in a workshop setting required considerable time and attention.

We also found that most instructors had directly taught their students some of the stylistic protocols we identified through our literature review, even though they had not previously been aware of the concept of "style choices" for circuit prototyping. For example, almost all of the instructors repeatedly reminded students to complete the implementation of their circuits before powering their system. Instructors varied significantly in their severity rankings for the compiled list of circuit style rankings (summarized in Table 1). Some argued, for example, that it was extremely important for students to consistently color code their wires, while others insisted this was not necessary.

We also found that instructor's attention was significantly strained during lectures, and as a result of this, enforcing these stylistic behaviors or other best practices is difficult:

*[P4] It's frustrating…when you just told the entire class [to check the polarity of their components] and then you immediately get a question from one student about why [their circuit] isn't working…and then the same questions from another student…only to discover that they didn't check the polarity of the resistors.*

This frustration reflects the exhausting demands placed upon the instructor to reinforce these practices in addition to facilitating the class. A reinforcement system that assists the instructor in peripherally supporting these style behaviors could potentially alleviate the instructor's burden while assisting new students in developing good implementation stylistic practices.

## Design Considerations

Based upon our literature review and our interviews, we synthesized the following criteria for designing a new system that reinforces circuit style practices.

***Glance-able awareness of student's progress.*** Since an instructor's attention may be monopolized by lecturing and assisting individual students, she should be able to monitor a student's progress throughout the duration of a workshop in a quickly-digestible manner. An instructor should preemptively identify and prevent common errors students may encounter without compromising with their focus.

***Supplement, not replace, an in-person instructor.*** Although log data can be useful in enforcing good style practices, this approach cannot capture the nuanced understanding of individual student skills and motivations like an in-person instructor. A style enforcement tool should allow the instructor to decide which stylistic choices should be enforced, as well as when and how these style choices will be enforced.

***Reinforced Mastery.*** Since many instructors do not have formal training or expertise in electronics prototyping and circuit styles, these instructors should be able to reinforce their own knowledge and mastery of styles. This design consideration is intended to allow instructors to find as many mistakes as possible in order to prevent these same mistakes in their student's work.

***Ambient Nature.*** To prevent distraction from the main task, a style reinforcement tool should be displayed ambiently. This can reduce the time-cost and required user interaction to access the help content and not add any cost when not using the guidance system (i.e., it can be easily dismissed).

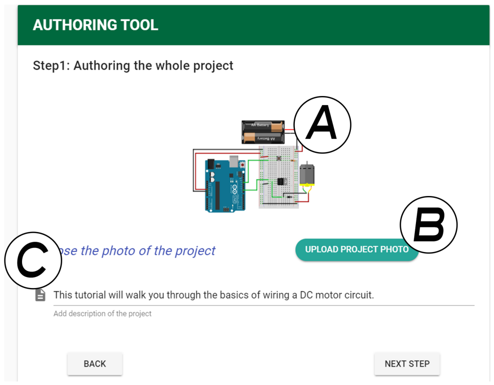
To minimize distraction, user should be rarely interrupted.

## DESIGN AND IMPLEMENTATION OF *CIRCUITSTYLE*

To address the above design considerations, we implemented *CircuitStyle*, a web-based tool to reinforce good circuit styles without monopolizing the student or teacher's attention. *CircuitStyle* allows instructors to author circuit tutorials and use the classroom management feature for live instruction, and a student interface for following tutorials and carrying out peer reviews.

### Tutorial Authoring Tool

*CircuitStyle* first allows instructors to interactively compose a step-by-step tutorial for students to follow (Figure 3). In addition, an instructor can assign circuit style behavior guidelines to individual steps in the tutorial, and decide how these behaviors will be evaluated.



Step1: Authoring the whole project

se the photo of the project

This tutorial will walk you through the basics of wiring a DC motor circuit.

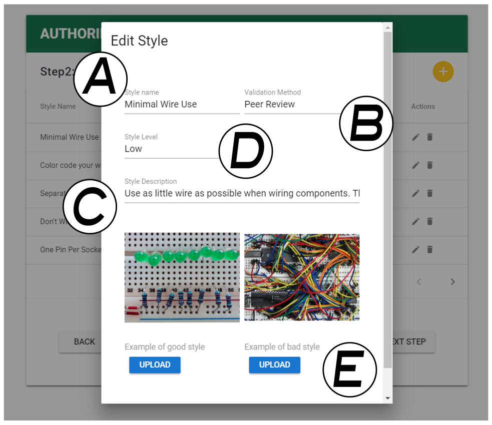
Add description of the project

**Figure 3: Overview of tutorial authoring; (A) Instructors can preview their uploaded circuit diagram; (B) Multiple photos can be uploaded to create a slide show; (C) Instructor can input additional instructions and step details.**

***Integration with Existing Tools:*** The instructors are first asked to indicate which circuit they would like to build. Many of the instructors that we interviewed suggested that their tutorials were often taken from websites such as *Sparkfun* or generated using the open source *Fritzing* software. To accommodate this, we allowed instructors to upload a csv file containing the steps of their tutorial generated by Fritzing. This also includes uploading a picture of the completed circuit for student reference.

***Style Authoring Tool and Default Styles:*** Our formative study showed that the breadth and importance of certain style choices varied greatly from instructor-to-instructor. To account for this, the second step of our authoring tool allows instructors to create their own style behaviors, or modify a list of default stylistic choices (Figure 4). Previous work indicated that coding styles are most effective when consistent across a project or organization, and malleable according to the instructor's needs [1, 13, 21]. Thus, allowing the instructors to modify the stylistic choices from project to project provides the flexibility instructors desire with the consistency students need.

Each circuit style (Figure 5) contains information pertaining to the proper implementation of the style, and indicates the level of severity which can adjust student's attenuation to various styles [22, 29]. In addition, instructors can upload photographs of good and bad implementations of the style which has been demonstrated to better engage students and assist in students being able to distinguish between good and bad circuit style implementation [22, 31]. Instructors can choose between validating their circuit styles through either a quiz or a student-driven peer review depending on which method they feel better evaluates that particular style [14]. Default style choices were compiled from our formative instructor interviews and aggregated according to their severity rankings. (e.g. wire color coding, connecting ground and power rails, etc.) These default styles are also editable by the instructor according to their needs.



Edit Style

Style name: Minimal Wire Use

Validation Method: Peer Review

Style Level: Low

Style Description: Use as little wire as possible when wiring components. Tl

Example of good style

Example of bad style

**Figure 4: Style Authoring Tool; (A) New styles can be created or default styles can be modified; (B) Styles verified using either quiz or peer review; (C) Description of proper style implementation; (D) Style details; (E) Photo examples of both good and poor style implementation can be uploaded.**

***Subsection Authoring and Scalability of Tutorial Complexity:*** Next we ask the instructor to organize the steps of their tutorial into subsections since we observed that this was done mostly manually by instructors. This step helps the overall organizational flow of the workshop, especially when working on larger, more complicated circuits. We designed this feature to help streamline this process and help scale workshops to larger circuits.

The instructors then step through the tutorial themselves and assign behaviors to each step. Since instructors may come from a variety of backgrounds, many of them may need to complete the tutorial themselves to fully understand the material. Our system accounts for this by encouraging tutorial authors to photograph their own circuit after each completed step. This is optional and can be replaced by images from Fritzing or other software. We encourage instructors to participate in this aspect of the tutorial authoring system, as mistakes made by the instructor could prove useful in preventing the same mistakes being replicated by students. The purpose of this step is to reinforce the instructor's understanding of the material and allow the instructor to teach a representational sample of coding styles to better familiarize students with good style habits [22]. In addition, this allows instructors to automate the style reinforcement process by scheduling trigger events that reinforce circuit style behaviors [32]. By completing the circuit themselves instructors may be better aware of which stylistic behaviors could prevent common mistakes.

### Interactive Live Tutorials for Students

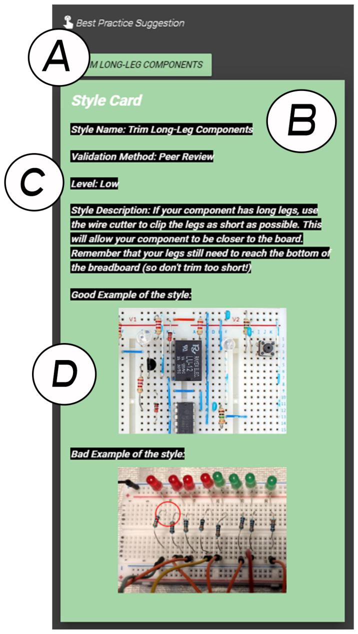
The student interface consists of a web application in tandem with a mobile phone application for live workshops. Our goal was to peripherally reinforce good circuit style behaviors in hardware computing for students without monopolizing too much student or instructor attention.

***Step Sorting:*** Once the instructor has completed authoring their tutorial, students can log into the phone application and web interface. Once logged-in, students are presented with an overview of the tutorial (Figure 1A). Many of our interviewed instructors expressed frustration with ensuring student engagement and attention when overviewing the circuit. To account for this, students are met with an image of the finished circuit, accompanied by a randomized sorting of the tutorial steps. Students are asked to sort the steps into the correct order before continuing. This peripherally reinforces common wiring procedure steps, as well as ensuring that students pay attention to the instructor as they walk students through the circuit.

***Step-by-Step Tutorial and Style Guide:*** Students are next guided through the construction of the circuit step-by-step as authored previously by the instructor (Figure 1D). The left expandable panel contains information regarding the particular style behavior indicated by the instructor during tutorial authoring for that step (Figure 5). This includes a detailed description of the style, and photos contrasting good and bad examples of the style behavior. Having access to style references has been documented in the literature as an effective and commonly-used method for incorporating style behaviors into programming since it mitigates the amount of memorization demanded by the student [31]. For this reason, we have included this feature here to facilitate a similar reference experience for hardware computing.

***Peer Review:*** To mitigate the amount of attention demanded by the instructor when evaluating these stylistic behaviors, we designed a peer evaluation system to allow students to review each other's work and provide feedback (Figure 6). This experience is akin to the coding style review process in the software engineering industry, and has demonstrated effectiveness at engaging students to learn through example and practice [22].

At the end of each section, students are asked to photograph their circuit. We chose to conduct the peer review process during section breaks to mitigate the participant's cognitive load that may result from code switching. Students are then presented with photos of other student's work, and asked if a particular style is evident in the photograph. Students can choose between "yes", "no", and "I don't know" in addition to leaving a comment (Figure 6). Encouraging peer-to-peer and peer-to-instructor interaction has been shown to be an effective reinforcement technique for coding styles [21].



Style Card

Style Name: Trim Long-Leg Components

Validation Method: Peer Review

Level: Low

Style Description: If your component has long legs, use the wire cutter to clip the legs as short as possible. This will allow your component to be closer to the board. Remember that your legs still need to reach the bottom of the breadboard (so don't trim too short!)

Good Example of the style:

Bad Example of the style:

**Figure 5: Style card from left of student tutorial screen in Figure 1D; (A) Expand/collapse style guide button; (B) Overview of style information and proper implementation; (C) Background color corresponded with severity level; (D) Photograph examples of both good and poor style execution.**

The number of circuits a student is asked to review at the end of each section depends on how many students are in the class, and how many styles the instructor designated for examination during tutorial authoring. This aims to reinforce the student's understanding of the peripheral style materials, mitigate potential errors, and reduce the amount of attention needed by the instructor. In addition, this practice is intended to encourage students to identify good circuit style behaviors, which has been demonstrated in the coding style literature to be an essential skill for developing good style behaviors [31]. After the student's peer review is completed by their fellow students, a Style Reminder slides-in from the top right hand corner indicating which styles the student performed well and possible violations.

***Style Reminder:*** After the peer review phase, students will receive feedback in the top right corner slide-out window which describes which styles they successfully performed and which they may have missed. A collapsible style guide directs the students to review any styles they may have missed, or performed poorly during their last peer review. The style reminder feature works in tandem with the collapsible style guide and the peer review system to reinforce good style.

### Student Progress Observational Interface

During live workshops and tutorials, we provide an interface for instructors to gauge workshop behavior and progress at-a-glance (Figure 1C). Our goal in the design of this system was to support the instructor by providing essential information on student performance, while not monopolizing instructor attention. This consists of 3 principle components described below.

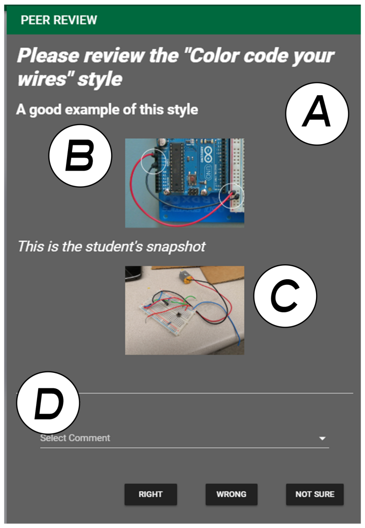
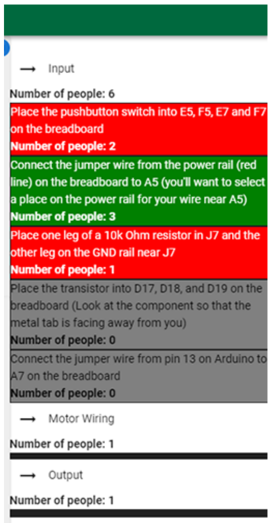

PEER REVIEW
Please review the "Color code your wires" style
A good example of this style
This is the student's snapshot
Select Comment
RIGHT
WRONG
NOT SURE
A
B
C
D

**Figure 6: The Peer Review Interface; (A) Name of particular circuit style to be evaluated; (B) Photograph example of good style implementation; (C) Photograph submitted by fellow student to be reviewed; (D) Student feedback options.**

***Tutorial Progress:*** The right hand panel contains an expandable hierarchy of the steps and sections in the tutorial as authored previously by the instructor (Figure 7). Next to each step is a number indicating how many students are currently completing that step of the tutorial.

In addition, tutorial steps and sections are color coded according to the following scheme: grey indicates that no students are currently working on this step, red indicates that very few (<0.25) are currently working on this step, and green indicates that the majority of students are currently working on this step (>.50). This color scheme was designed to allow instructors to momentarily glance at the screen, and have an understanding of the class's overall progress through the tutorial [8].



→ Input

Number of people: 6

Place the pushbutton switch into E5, F5, E7 and F7 on the breadboard
Number of people: 2

Connect the jumper wire from the power rail (red line) on the breadboard to A5 (you'll want to select a place on the power rail for your wire near A5)
Number of people: 3

Place one leg of a 10k Ohm resistor in J7 and the other leg on the GND rail near J7
Number of people: 1

Place the transistor into D17, D18, and D19 on the breadboard (Look at the component so that the metal tab is facing away from you)
Number of people: 0

Connect the jumper wire from pin 13 on Arduino to A7 on the breadboard
Number of people: 0

→ Motor Wiring

Number of people: 1

→ Output

Number of people: 1

**Figure 7: Collapsible step hierarchy from Instructor's in-situ tutorial screen in Figure 1C. Steps are color coded according to density of students currently completing that step.**

***Student Submissions:*** The center of the screen can be button-toggled to either reflect information regarding a specific step, or view student's most recent submissions for peer review (Figure 1C). The tutorial view allows the instructor to view the information pertaining to any particular tutorial step, including instructions and photographs related. By pressing the toggle button, the student submission window displays the most recently submitted student photos for peer review, as well as the student's name, and a pie chart reflecting their successful performance of circuit style behaviors as evaluated by peer-reviews and quizzes (Figure 8A).

This is intended to provide awareness of each students past behavior history, current status [8]. The screen also allows instructors to provide positive feedback to students in the form of a thumbs-up, direct intervention in the form of freezing the student's screen, constructive feedback via comments, as well as mitigate various issues regarding the peer review process such as contested reviews (Figure 8C). We designed this feature to encourage individualized feedback that complements the automated triggered feedback of quizzes and peer review [17, 22, 32].

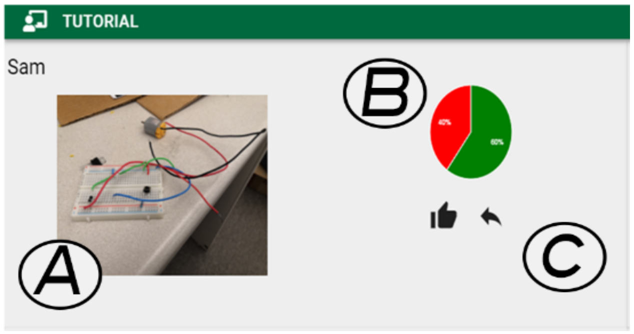


**Figure 8: Detail of intervention methods from Instructors in-situ tutorial screen in Figure 1C; (A) Student information and latest photo submitted as part of peer review; (B) Pie chart reflecting student's successful performance of circuit styles; (C) Intervention methods (e.g., thumbs-up, direct messaging)**

***Behavior Performance:*** The bottom panel contains a series of pie charts (Bottom of Figure 1C) showing student stylistic behavioral performance (the green indicates the proportion of students who successfully completed that stylistic behavior during the previous peer-review). These charts are sorted from worst-performing behavior to best-performing behavior, allowing the instructor to gauge which behaviors might need to be further reviewed. Providing awareness of overall classroom performance is necessary to understanding areas of needed reinforcement and direct intervention [22].

## USING CIRCUITSTYLE IN PRACTICE: A CASE STUDY

We deployed *CircuitStyle* in a workshop setting to investigate whether circuit style behaviors could be peripherally enforced by our system and to gather users' initial impressions of the system's usability and usefulness.

### Procedure
We recruited 11 students for a 90-minute instructor-led electronic prototyping workshop where students were asked to create a circuit and used CircuitStyle for assistance. The instructor was asked to construct a typical makerspace circuit tutorial using our interactive authoring tool. We asked the instructor to narrate their experience using our tool as they proceeded with the tutorial composition process using our software. The resulting circuit tutorial guided students through the creation of a button-activated, battery powered motor circuit. The instructor was given a brief overview of our software before construction their tutorial.

Students were then brought into the workshop area and asked to complete a brief preliminary questionnaire regarding their previous experience with electronics and awareness of coding style practices, as well as circuit style behaviors. Next, the students were instructed to log into our web application and phone application and guided by the instructor and software to complete the tutorial. We also recorded video of student progress which helped us extract various quantitative measurements of student performance.

Upon completion of their circuit, students were asked to complete an exit questionnaire and interview. We also conducted an exit interview with the instructor to better understand their experience with our system.

### Study Participants
Our workshop was led by an experienced maker-space workshop leader who also authored the tutorial and guided the workshop. He was recruited from another university due to his extensive experience and interests in conducting mentorship practices in makerspaces and running workshops in hardware computing. His formal education consisted of film studies although he now is pursing a doctorate in computer science and business. The 11 workshop participants (2 females, 8 males, 1 non-binary) ranged in age from 20 – 29 and possessed some familiarity with electronics. But most participants (63%) did not have a formal background in electronics, even if their current work involved prototyping with electronics. The majority of participants (45%) were studying software engineering or computer science related disciplines while others were concentrating in electrical engineering (27%), music (19%), or design (9%). While almost all (91%) of our participants were familiar with programming coding styles, only 36% were familiar with circuit styles. Those that were familiar, indicated that they had learned these behaviors from instructors or by learning on their own. None of the participants were involved with this project's research in any capacity beyond participating in the study.

### Key Findings from the Workshop
We found that students overall enjoyed the workshop and that the instructor found the tool useful for planning their lesson as well as surveying the performance of the workshop. We discuss our key results by first focusing on the facilitator impressions followed by student feedback.

#### *Facilitator Impressions*
In this section, we revisit our design goals presented earlier in context of feedback provided by the workshop instructor.

***Glance-able awareness of student's progress:*** The instructor overall appreciated *CircuitStyle*'s assistance in keeping track of the overall progress of the class. In particular, he emphasized the utility of the automatic indication of which students had completed which steps. He highlighted that the visual nature of many UI elements helped mitigate the amount of attention he had to attenuate to the system. For example, he found the color coding of individual steps to be intuitive and helpful:

*It was useful to see where people were in the steps… the color coding was helpful because it let me know where they were getting stuck without having to look too hard.*

The instructor reported that usually with step-by-step tutorials, he had little information on how students were progressing in the project. He was enthusiastic about the follow-along detection feature in *CircuitStyle* that allowed him to see the most recent photo of the student's circuit:

*I'm a visual person so I liked seeing the individual circuits…this was the most helpful aspect for me because I could immediately see how the students were doing…was really helpful to keep track of where they might get stuck.*

In a traditional workshop environment, the instructor would spend a lot of time walking around to see how students were performing. With *CircuitStyle*, he could spend more time actually helping the struggling students:

*[With CircuitStyle]…I could hone in on a few students and help them as opposed to walking around to find out who was struggling.*

He also expressed appreciation at our positive re-enforcement tools, such as the ability to give a student's most recent submission a "thumbs-up" or send a quick note. He indicated that novice students often are unsure if they completed a step correctly, and this allowed him to provide positive re-enforcement of good work.

***Supplement, not replace, an in-person instructor's capabilities:*** In terms of supplementing in-person instructor capabilities, we were surprised to find that the instructor expressed an increase in their engagement with the class. In interacting with our behavior performance feature, he mentioned that although it was a secondary task to examine the pie charts at the bottom of the screen, it was helpful to see if one particular style was severely being missed:

*I liked that it showed me potential challenges students might face…because they weren't following the style guide…and seeing individual student circuits told me who needed help.*

Despite the instructor's enthusiasm for our system, he indicated that the system might be even more helpful for students in larger workshops. In our workshop, we noticed that if a student encountered a problem, they would usually ask their immediate neighbor for assistance in debugging the problem and not always rely on the peer review. Examining the usefulness of this feature in learning contexts where one-to-one interaction may be more difficult for students and instructors is a key area for future investigations.

***Reinforced Mastery:*** We asked the instructor to comment on the interactive authoring tool and its ability to help clarify the instructor's understanding of the material. Overall, he found the step-by-step nature of the authoring tool helpful and appreciated the ease of finding potential pitfalls that students might encounter, and how adhering to a specific style may prevent those errors:

*I noticed that students might not wire their ground and power on the same rail…which was good, because connecting the rails with a wire is a good thing to do.*

The instructor also mentioned how helpful it was to lay out and design a specific tutorial and learn from their own mistakes in implementing the project before the student:

*It was nice because I had time to prepare and it was much more defined…less things can go wrong this way and if things do go wrong, I can spend less time debugging it.*

He also suggested that adding features to allow note-taking at various levels might be helpful as well. Both while prepping the tutorial and while facilitating the workshop, the instructor mentioned that they took copious notes which became unwieldy to organize according to step, section, and overall workshop activity.

***Ambient Nature:*** A key concern of this project was to not monopolize or further strain the instructor's attention with our system. The instructor expressed appreciation of the peer-review system's ability to enforce these behaviors without relying on persistent instructor intervention:

*I didn't have to worry about identifying who was struggling… was good because I knew who needed help from my screen and could go straight to them.*

In addition, the instructor expressed that the overall UI reduced the burden of enforcing good circuit implementation practices by peripherally encouraging these behaviors with our system.

### *Student Feedback*

All students were able to successfully complete the circuit well within the allotted 90 minutes. The average completion time of the total circuit tutorial was 41min 16s (9min 14s SD). Only 3/11 students who used the style guide and followed the step-by-step tutorial system required additional assistance from the TAs outside of the instructor's lecture. Questionnaire feedback indicated that students were overall enthusiastic about the workshop, rating their enjoyment an average of 4.1 out of 5. Figure 9 shows a summary of students' responses to individual components of the interface on a standard 5 point Likert scale based on the following criteria, i) Helpful, ii) Distracting, iii) Confusing, iv) Difficult, and v) Engaging.

***Sorting step:*** We asked students if the sorting step was helpful for reinforcing their understanding of proper procedure. Overall, our results indicate that many students (in particular, those with little to no background in electronics) found this step to be particularly confusing (See Figure 9). Our main motivation for creating the sort-ordering task was to make it easier for instructors to tell their students to complete their circuits in a proper order (e.g., inserting ICs first and powering the circuit last). Although somewhat useful, our results showed that novice students found this sort-ordered task to be confusing.

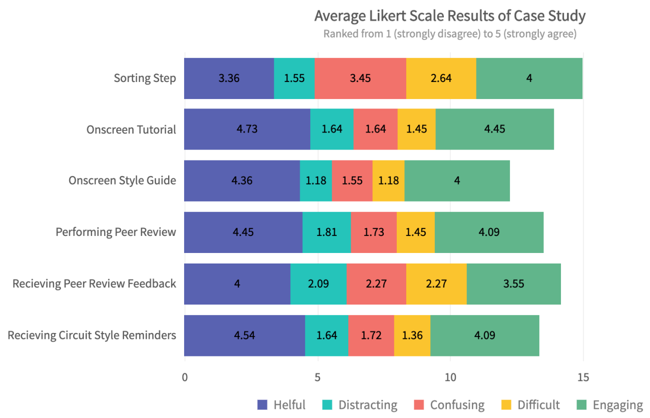


**Figure 9: Likert scale responses for CircuitStyle case study.**

***Onscreen tutorial:*** Overall, students were enthusiastic about following along with the instructor's pre-written tutorial. Many participants expressed that this was their favorite aspect of the system. We also noticed that having both written instruction and a picture was helpful. Some participants reported relying more heavily on the image than the text, and vice versa. One student initially ignored the tutorial and style guide altogether because they were confident in their abilities, but struggled later to complete the circuit and eventually found several missed steps.

***Onscreen style guide:*** Participants were enthusiastic about the onscreen style guide, noting that it helped keep their circuit tidy and organized. Most students, even those with electrical engineering training, referenced the style guide while constructing their circuit and found it helpful:

*[P8] It was good because most of these things I learned by making mistakes...some of [the styles] I learned before and it was a nice review, but some I had not learned yet.*

Many novice users also indicated that the style guide helped them feel more secure in their circuit and reassured that they were progressing through the tutorial adequately.

***Performing Peer Review:*** In an attempt to minimize the attention needed by the instructor to reinforce style behaviors, we required the students to perform peer review. Students overall reported that performing peer review helped reinforce their understanding of the material, but wondered if it was useful in such a small class:

*[P4]…it was useful to see how others were doing…but I think this might be more useful in a larger class…it might also be difficult with more complicated circuits since you won't really be able to see [the style] from a picture.*

Novice users also reported feeling insecure about the helpfulness of their comments to other students. While participants with formal training in hardware computing found providing feedback to be an intuitive process, novice users were unsure if their input was helpful:

*[P2] I wasn't sure if my comments were helpful since I didn't have much experience working with circuits…I might say something wrong.*

Although students have an option to select "I don't know" and are not obligated to leave a comment, students may feel obligated to provide more input than required.

***Receiving Peer Review Feedback:*** Participants overall appreciated receiving a peer reviewed report, but felt that the current system did not provide enough feedback. Our current method of delivering feedback to students involves a small scroll-in window in the top right-hand corner of the screen to make it as unobstructed as possible. However, many students reported looking for further information on their performance and not being able to locate it.

***Receiving Circuit Style Reminders:*** We also asked participating students to evaluate our style reminder system and found that this feature helped students identify potential problem areas while debugging:

*[P5] I had trouble getting my circuit to work at the end but remember that one [style reminder] said that I hadn't connected my ground rails…it was great. I knew where to debug and get it working.*

## DISCUSSION

We have contributed the design of *CircuitStyle*, a tool for peripherally reinforcing circuit style behaviors for in-person workshop tutorials. We have also demonstrated the value of techniques which assist instructors in facilitating workshop activity, as well as laid the preliminary work for exploring the applicability of circuit style behaviors to improving hardware computing education. Although we only examined *CircuitStyle*'s utility in a single small workshop setting, a more in-depth investigation into how this tool impacts instructor and students' experience at a large scale can provide additional useful insights. In this section, we discuss some limitations and avenues for future work.

### Scaling Peer-Review Features

When asked to complete peer review, several students asked their neighbors to evaluate if the work had been done correctly. This was unexpected, and a practice that could be encouraged by a differently configured peer review system. In future work, it would also be useful to understand the performance of our current system at a larger scale workshop. Similarly, our peer review system may prove difficult to scale for more complicated circuits since identifying style behaviors from a single photograph of a complex circuit may be challenging, especially for new users. This issue could also possibly be alleviated by encouraging in-person peer evaluations as opposed to virtual peer evaluations.

### Accounting for Varying Skill Levels

Some students also expressed insecurity in their ability to provide useful feedback to their fellow students, particularly if the student was a complete novice user. We could account for this by calibrating our system prior to the workshop with user background information, and establish a "virtual mentoring" system by encouraging more experienced users to provide feedback to less experienced users. Finally, our system requires the instructor to author a step-by-step tutorial for the students, and thus our current system is only reliable for follow-along workshops. Evaluating our system's usability in free-form workshop environments is a principle area of future work.

### Alternative Workshop Structures

Although our system proved sufficient for pre-structured tutorials, not all workshops employ follow-along instruction methods for teaching hardware computing. Since instructor and student activity in these free-form workshops differs significantly, additional design considerations must be accounted for employing our system in this domain. Additionally, deploying our system in alternative workshop settings such as formal education classrooms could provide insight into the versatility and longevity of our system. Identifying, adapting, and evaluating our system for such workshop structures is a key area for further investigation.

### AR-based Approaches

Incorporating additional input modalities and interaction techniques could further mitigate some of the attention demands of the system. Incorporating AR into our system could provide additional methods of communicating and peripherally reinforcing circuit style and tutorial material to students, as well as further aiding the instructor in facilitating the activity of the workshop. Additionally, this interaction modality could encourage physical activity and peer interaction during the workshop which could be particularly useful during peer review.

## CONLCUSION

This work provided initial validation for the applicability of circuit styles in follow-along tutorials as well as supported the notion that circuit style behaviors could be peripherally reinforced. Our prototype system and case study evaluated a series of techniques that aide instructors in authoring tutorials, facilitating workshop activity, maintaining awareness of class progress, and reinforcing good circuit prototyping practices without monopolizing instructor attention. Our work lays the foundation for architecting a future where instructors collaboratively share the experience of teaching with a trusted system, allowing the instructor to fully focus on enjoying the mentoring of their students. More broadly, our work calls for more HCI research in the domain of hardware computing to better support the growing number of novice and untrained users.

## REFERENCES

[1] Parasoft. Understanding the Workflow in a Coding Standards Implementation. 2005. Accessed July 19, 2019.

docs.parasoft.com/display/CPPDESKV1033/Best+Practices+and+Workflows

[2] Fraser Anderson, Tovi Grossman, and George Fitzmaurice. 2017. Trigger-Action-Circuits: Leveraging Generative Design to Enable Novices to Design and Build Circuitry. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology.* 331–342. DOI=https://doi.org/10.1145/3126594.3126637

[3] Ronald E. Anderson. 1992. Social Impacts of Computing: Codes of Professional Ethics. *Soc Sci Comput Rev* 10( 2). 453-469.

[4] 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. DOI=https://doi.org/10.1145/2858036.2858533

[5] Amiangshu Bosu, Michaela Greiler, and Christian Bird. 2015. Characteristics of Useful Code Reviews: An Empirical Study at Microsoft. *Proceedings of Working Conf. Mining Software Repositories*. 146–156.

[6] Hsiang-Ting Chen, Tovi Grossman, Wei Li-Yi, Ryan M. Schmidt, Björn Hartmann, George Fitzmaurice, and Maneesh Agrawala. 2014. History Assisted View Authoring for 3D Models. *Proceedings of the ACM Conference on Human Factors in Computing Systems*. 2027–2036.

[7] Rohan Roy Choudhury, HeZheng Yin, and Armando Fox. 2016. Scale-Driven Automatic Hint Generation for Coding Style. In *13th International Conference on Intelligent Tutoring Systems.*

[8] Sunny Consolvo, Katherine Everitt, Ian Smith, James A. Landay. 2006. Design Requirements for Technologies that Encourage Physical Activity. *Proceedings of the SIGCHI Conference on Human Factors in Computing.* 457–466.

[9] Daniel Drew, Julie L. Newcomb, William McGrath, Filip Maksimovic, David Mellis, and Björn Hartmann. 2016. The Toastboard: Ubiquitous Instrumentation and Automated Checking of Breadboarded Circuits. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology.* 677– 686. DOI=https://doi.org/10.1145/2984511.2984566

[10] Volodymyr Dzuibak, Ben Lafreniere, Tovi Grossman, Andrea Bunt, George Fitzmaurice. 2018. Maestro: Designing a System for Real-Time Orchestration of 3D Modeling Workshops. *Proceedings of ACM Conference on User Interface Software Technology.*

[11] Adam Fourney, and Ben Lafreniere. 2014. InterTwine: Creating Interapplication Information Scent to Support Coordinated Use of Software. *Proceedings of the ACM Symposium on User Interface Software and Technology*. 429–438.

[12] Adam Fourney and Michael Terry. 2012. PICL: portable in-circuit learner. in *Proceedings of the 25th annual ACM symposium on User interface software* and *technology*. 569-578.

[13] Josh Fryman. Coding Standards: Good Idea or Subtle Evil? 1999. Accessed July 15, 2019. http://freshmeat.sourceforge.net/articles/coding-standards-good-idea-or-subtle-evil

[14] Elena Leah Glassman, Lyla J Fischer, Jeremy Kenneth Scott, and Robert C. Muller. Foobaz: Variable Name Feedback for Student Code at Scale. DOI=609-617. 10.1145/2807442.2807495.

[15] Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2010. Chronicle: Capture, Exploration, and Playback of Document Workflow Histories. *Proceedings of annual ACM Symposium on User Interface Software and Technology*. 143–152.

[16] Philip J Guo. 2015. Codeopticon: Real-Time, One-To- Many Human Tutoring for Computer Programming. *Proceedings of the ACM Symposium on User Interface Software & Technology*. 599–608.

[17] Chinmay Kulkarni, Michael S. Bernstein, Scott Klemmer. 2015. PeerStudio: Rapid Peer Feedback Emphasizes Revision and Improves Performance. *L@S*. DOI=10.1145/2724660.2724670.

[18] Ben Lafreniere, Tovi Grossman, and George Fitzmaurice. 2013. Community Enhanced Tutorials: Improving Tutorials with Multiple Demonstrations. *Proceedings of International Conference on Human Factors in Computing Systems*. 1779–1788.

[19] Chee Kit Looi, Wenli Chen, and Foo Keong Ng. 2010. Collaborative Activities Enabled by GroupScribbles: An Exploratory Study of Learning Effectiveness. *Computers and Education* 54(1). 14–26.

[20] Wei Li, Tovi Grossman, and George Fitzmaurice. 2014. CADament: a Gamified Multiplayer Software Tutorial System. *Proceedings of the International Conference on Human Factors in Computing Systems*. 3369–3378.

[21] Xiaosong Li. 2006. Using Peer Review to Assess Coding Standards – A Case Study. *Proceedings of 36th Annual Conference on Frontiers in Education. DOI*=10.1109/FIE.2006.322572.

[22] Xiaosong Li, and Christine Prasad. “Effectively Teaching Coding Standards in Programming”. 2005. *Proceedings of the 6th conference on Information Technology Education*. 239–244.

[23] Jin-Su Lim, Jeong-Hoon Ji, Yun-Jung Lee, Gyun Woo. 2011. Style Avatar: A Visualization System for Teaching C Coding Style. *Proceedings of the 2011 ACM Symposium on Applied Computing*. 1210-1211.

[24] Will McGrath, Daniel Drew, Jeremy Warner, Majeed Kazemitabaar, Mitchell Karchemsky, David Mellis, and Björn Hartmann. 2017. Bifröst: Visualizing and Checking Behavior of Embedded Systems Across Hardware and Software. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 299–310. DOI=https://doi.org/10.1145/3126594.3126658

[25] David A. Mellis, Leah Buechley, Mitchel Resnick, and Björn Hartmann. 2016. Engaging Amateurs in the Design, Fabrication, and Assembly of Electronic Devices. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*. 1270– 1281. DOI=https://doi.org/10.1145/2901790.2901833

[26] Paul W. Oman, and Curtis R. Cook. 1990. A Taxonomy for Programming Style. *Proceedings of Annual Conf. Cooperation*. 244–250.

[27] Terence Parrand, Jurgen Vinju. 2016. Towards a Universal Code Formatter Through Machine Learning. *Proceedings of Int’l Conf. Software Language Engineering*. 137–151.

[28] Suporn Pongnumkul, Mira Dontcheva, Wilmot Li, Jue Wang, Lubomir Bourdev, Shai Avidan, and Michael F. Cohen. 2011. Pause-and-Play: Automatically Linking Screencast Video Tutorials with Applications. *Proceedings of the ACM Symposium on User Interface Software and Technology*. 135–144.

[29] Srđan Popić, Gordana Velikić, Hlavač Jaroslav, Zvjezdan Pavkovic, Marko Vulić. 2018. The Benefits of the Coding Standards Enforcement and its Impact on the Developers Coding Behaviour-A Case Study on Two Small Projects. DOI=10.1109/TELFOR.2018.8612149.

[30] Mitchel Resnick, Fred Martin, Randy Sargent and Brian Silverman. 1996. Programmable bricks: Toys to think with. *IBM Systems journal*, 35 (3.4). 443-452.

[31] Michael Smit, Barry Gergel, H. James Hoover, Eleni Stroulia. 2011. Code Convention Adherence in Evolving Software. *Proceedings of 27th IEEE International Conference on Software Maintenance.* DOI=10.1109/ICSM.2011.6080819.

[32] Katarzyna Stawarz, Anna L. Cox, Ann Blandford. 2015. Beyond Self-Tracking and Reminders Designing Smartphone Apps that Support Habit Formation. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems.* 2653–2622.

[33] Evan Strasnick, Maneesh Agrawala, and Sean Follmer. 2017. Scanalog: Interactive Design and Debugging of Analog Circuits with Programmable Hardware. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 321–330. DOI=https://doi.org/10.1145/3126594.3126618

[34] David Verweij, Saskia Bakker, and Berry Eggen. 2017. FireFlies2: Interactive tangible pixels to enable distributed cognition in classroom technologies. In *Proceedings of the ACM Conference on Interactive Surfaces and Spaces*. 260–269.

[35] Jeremy Warner, Ben Lafreniere, George Fitzmaurice, Tovi Grossman. 2018. ElectroTutor: Test-Driven Physical Computing Tutorials. *Proceedings of ACM Symposium on User Interface Software Technology.*

[36] Te-Yen Wu, Bryan Wang, Jiun-Yu Lee, Hao-Ping Shen, Yu-Chian Wu, Yu-An Chen, Pin-Sung Ku, Ming-Wei Hsu, Yu-Chih Lin, and Mike Y. Chen. 2017. CircuitSense: Automatic Sensing of Physical Circuits and Generation of Virtual Circuits to Support Software Tools. *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 311–319. DOI=https://doi.org/10.1145/3126594.3126634