J Sci Educ Technol
DOI 10.1007/s10956-017-9694-0

Stitching Codeable Circuits: High School Students' Learning About Circuitry and Coding with Electronic Textiles

Breanne K. Litts[1] · **Yasmin B. Kafai**[2] · **Debora A. Lui**[2] · **Justice T. Walker**[2] · **Sari A. Widman**[2]

Abstract Learning about circuitry by connecting a battery, light bulb, and wires is a common activity in many science classrooms. In this paper, we expand students' learning about circuitry with electronic textiles, which use conductive thread instead of wires and sewable LEDs instead of lightbulbs, by integrating programming sensor inputs and light outputs and examining how the two domains interact. We implemented an electronic textiles unit with 23 high school students ages 16–17 years who learned how to craft and code circuits with the LilyPad Arduino, an electronic textile construction kit. Our analyses not only confirm significant increases in students' understanding of functional circuits but also showcase students' ability in designing and remixing program code for controlling circuits. In our discussion, we address opportunities and challenges of introducing codeable circuit design for integrating maker activities that include engineering and computing into classrooms.

Keywords Circuitry · Electronic textiles · Assessment · Making · STEM · Maker movement

Introduction

The push to promote STEM topics in K-12 education (Katehi et al. 2009; Smith 2016) has recently been joined by efforts to promote computational thinking for all (Wing 2006). Maker activities in which youth engage with hands-on experiences have been seen as a particularly promising vehicle to engage youth in interdisciplinary STEM activities (Honey and Kanter 2013; Peppler et al. 2016a, b). One such maker activity which combines engineering, crafting, and programming is electronic textiles (hereafter, e-textiles), which are circuits constructed with conductive thread, sewable LEDs and sensors, and sewable microcontrollers, like the LilyPad Arduino (Buechley 2006). Using e-textiles activities in classrooms and afterschool workshops has been shown to raise girls' interest in computing (Kafai et al. 2014a), women's engagement in the larger DIY community (Buechley and Hill 2010), and students' overall interest in science and STEM careers (Tofel-Grehl et al. 2017).

So far, most of the research examining what students learn through making e-textiles has focused on assessing students' understanding of simple functional circuits (e.g., Peppler and Glosson 2013) or their learning of programming concepts (e.g., Kafai et al. 2014b). We know much less about students' learning at the intersection of crafting, circuitry, and coding with electronic which could further highlight possible STEM integration. Learning how to make codeable circuits sits at this very intersection and involves students in designing and crafting a functional circuit that also can be controlled via code. For instance, students can learn to design a complex circuit that can turn on or off particular LED lights with a switch or when a sensor reads data above or below a set value. Such activities not only provide a rich demonstration of how to integrate electronics and computing with crafting and creativity in science education but also present complex challenges for the design of instruction and assessment.

In this paper, we examine how the design and crafting of functional simple circuits can be expanded to circuits with programmable features. We conducted a 15-session e-textiles unit during which 23 high school students (ages 16–17) learned how to make e-textiles. This was the first pilot in the

✉ Breanne K. Litts
breanne.litts@usu.edu

[1] Utah State University, 2830 Old Main Hill, Logan, UT 84322, USA

[2] University of Pennsylvania, 3700 Walnut Street, Philadelphia, PA 19104, USA

Published online: 29 May 2017

development of an e-textiles curricular unit for the Exploring Computer Science curriculum, an introductory high school course (Goode et al. 2014). We asked the following research questions: (1) Does students' understanding of a functional circuit improve in a complex e-textiles design project? (2) Can students read, design, and remix codeable circuits after completing a complex e-textile project? Our analyses not only show significant improvements between pre/post tasks of students' understanding functional circuits but also showcase students' ability to read, design, and remix code for controlling circuits. In our discussion, we address how this type of circuit design can lead to a better understanding of functionality and provides a promising context for integrating maker and computational activities in science classrooms.

Background

Efforts to integrate new technologies in science education aim to engage "students in carrying out complex project that connect advances in technology with important science topics" (Linn 2003). We present e-textiles design projects as a complex problem space in which students learn circuitry, coding, and their interdependencies. The research on examining learning with e-textiles has been most closely connected with prior work in science education that examined students' understanding and misconceptions of circuit design. Osborne's (1981, 1983) seminal work found that elementary school students tend to struggle with understanding circuitry and typically generate linear representations rather than loop-based representations of circuits. Moreover, these misconceptions persist with high school or college-age students, even with instruction (Duit and von Rhöneck 1998; Fredette and Lochhead 1980; Kock et al. 2013; Shepardson and Moje 1994). Issues like current flow (e.g., Shipstone 1984) and polarity (e.g., Osborne 1981) are two of the most prevalent misconceptions in students' learning of simple circuitry and scholars often point to the abstract nature of traditional teaching models and learning materials as prime contributors. Aware of these challenges, circuitry is often taught using materials like breadboards, wires, light bulbs, and batteries.

More recently, circuitry teaching and learning has expanded to include other conductive materials such as play dough (Johnson and Thomas 2010), circuit stickers (Qi and Buechley 2014), or conductive thread, sensors, and microcontrollers (Buechley et al. 2007). For example, sewable microcontrollers, such as the LilyPad Arduino (Buechley 2006) or Adafruit's Flora (Stern and Cooper 2015), enable users to craft interactive, wearable designs by stitching circuits with conductive thread. The emergence of these new tools and materials has sparked scholars to explore their effectiveness at content delivery, especially in science education in relation to the Next Generation Science Standards (Tofel-Grehl et al. 2016). In one study, Peppler and Glosson (2013) measured students' ability to create a functional circuit before and after students participated in an out-of-school e-textiles workshop using stickers of LilyPad-specific components (an LED, a battery holder with battery, and a switch). Findings revealed that students' ability to create a working circuit, as well as students' understanding of current flow, connections, and polarity, significantly increased from pre to post (Peppler and Glosson 2013). Furthermore, Halverson et al. (2016) examined whether students learning circuitry through a music-based maker activity in museum and classroom settings. The authors employed a similar sticker-based circuitry diagram assessment as Peppler and Glosson (2013) and found that a significant number of students shifted from a linear-based to a loop-based representation of circuits (Halverson et al. 2016).

While these studies of students' electronic textiles designs have demonstrated promising findings in helping students to learn about simple and parallel circuits and further their understanding of key concepts such as polarity and current flow, they did not address student learning of programming circuits controlled by microcontrollers. It is at this intersection between engineering and computing that circuit design and code control takes students' understanding into STEM application areas that integrate computational thinking (Grover and Pea 2013; Wing 2006). The design of circuits becomes foundational for the design of software (and vice versa) that control the input/output interaction of sensors, lights, and motors. One study that has examined this integrational dimension of electronics and computing has focused on college students' ability to successfully engage in creative prototyping with modular electronics (Sadler et al. 2016). Others have made efforts to assess students' abilities to problem solve, troubleshoot, and integrate scientific and design principles with fabrication technologies (including microcontrollers), but do not go so far as to specifically study learning of codeable circuitry (Blikstein et al. 2017). While there has much research on student understanding of computer science knowledge and programming concepts *on the screen* (Guzdial 2015; K–12 Computer Science Framework 2016; Soloway and Spohrer 1989), no studies exist that have specifically examined K-12 students' understandings of codeable circuit design with modular electronics or electronic textiles, which have components both *on and off the screen*.

In this paper, we take on the challenge of investigating the high school students' understanding of functional circuit and software design, or codeable circuits, through a combination of pre/post tasks by examining two key aspects: (a) reading codeable circuit designs and (2) designing and remixing functional code for controlling circuits. What is both interesting and challenging about codeable circuits is that they integrate two different modalities—visual architecture of the circuit and written text of programs—and how these are connected via a microcontroller. In each instance, students have to be able to interpret and produce not only the blueprints of circuit designs but also the code for microcontrollers (in our case the LilyPad

Arduino board), which collectively work to perform certain actions such as turning on or off LED lights in patterns or reading data from light or touch sensors and reacting accordingly. Furthermore, in designing or remixing code for circuit diagrams, the intersecting features become critical for how inputs and outputs can be controlled. For instance, LEDs in a parallel circuit design can only be turned on or off concurrently giving the designer no control over individual lights' functionality. We have chosen to examine different competencies such as reading, designing, and remixing both code and circuit designs to provide students with different contexts in which to showcase their understanding, fully realizing that these contexts vary in difficulty for novice student designers.

Methods

Participants and E-Textile Unit

We conducted this study with a class of 23 high school juniors (4 boys, 19 girls, 16–17 years old) within a STEM elective class during the school day, based at a charter school in a northeastern metropolitan city. Student demographics mirrored those of the school with 44% African American, 35% Caucasian, 13% Hispanic, 3% Asian, and 3% Multiracial students. Most students (n = 18) had completed an introductory e-textiles project at the end of the previous academic year (Litts et al. 2015), but six students were new to the STEM elective class and were introduced to e-textiles for the first time. The participating teacher, a trained biologist, was also the STEM coordinator of the school. The teacher put students in pairs aiming to balance skills and expertise, personality traits, and existing friendships. Student absences occasionally occurred throughout the unit; while two students had very high absentee rates (5 or more times out of 15 days), four others had lower absentee rates (2–3 times out of 15), and the rest had minimal absentee rates (0–1 times out of 15). All 23 students were included in the analyses.

The teacher worked together with our team to prepare and guide 15 sessions each lasting about 90 min that took place over the course of 3.5 weeks during the STEM elective class within the school day. The teacher co-taught the curriculum with Author 5, though Authors 1 and 3 were in the room helping facilitate activities as needed. Over the course of the e-textiles unit, student pairs collaboratively constructed an interactive sign for the school that would be exhibited in a high-traffic area of the school. Each student pair received a letter printed onto canvas, which was designed by an art major student at the same school; collectively the letters constructed a sign of the school's name. Each pair also received a LilyPad Arduino, LEDs, sensors, switches, and other e-textiles materials to make the sign interactive. Figure 1 pictures an art student's design featuring an orange-colored letter ("E") superimposed over a patchwork of five different street scenes. The student pair added to this artwork by sewing a LilyPad circuit that incorporated nine LEDs (four in the upper left, two at middle left, and three at lower center), as well as one switch and two conductive patches (lower right), which were used for activating four distinct light patterns (for example, blinking back and forth between blue and white lights or turning on the different "traffic light" LEDs). It should be noted here that pairs divided the work in different ways; while a few divided their efforts evenly (each doing half the coding and half the sewing), most other pairs divided the work according to domain (one member acting as coder, and another as sewing).

Data Collection and Analysis

Before and after this e-textiles unit, all students participated in a series of different circuit and coding tasks, which were administered by researchers (Authors 3 and 5) within an interview format. These interviews were videotaped and transcribed, and all artifacts such as student-generated diagrams and handwritten comments were captured. Two of these tasks were implemented before and after the unit (designing a functional circuit; reading a codeable circuit design), while a third was included only after the unit (designing, reading, and remixing a codeable circuit). Even though all 23 students participated in these tasks, one student did not complete the entire series due to time constraints. As a result, there was a resulting N = 23 for tasks 1, 2, and 3a, and N = 22 for tasks 3b and 3c. Table 1 provides an overview of time point and data types of the three tasks analyzed in this study, and are described further below.

Designing a Functional Circuit (Pre and Post)

The structure of task 1 was modeled after Peppler and Glosson's (2013) e-textile-based circuitry task that used stickers with lights, switches, and batteries. However, we took a more open-ended approach to the task by asking students to "draw a working circuit with a light and a battery" (see Fig. 2). This more open-ended approach has been used effectively in previous studies (e.g., Osborne 1983; Shepardson and Moje 1994) and was more appropriate for our context, because students used microcontrollers in their projects rather than light, battery, and switch components.

We adapted the coding schemes from Peppler and Glosson (2013) and Halverson et al. (2016) for analyzing student performance in regard to three aspects of a simple circuit: (a) polarity, (b) connection, and (c) current flow. We coded each specific feature as 1 (present/correct) or 0 (not present/incorrect). We determined *polarity* by the presence of battery polarity, light polarity, and matching polarity (+ to +, − to −); we ascertained *connection* types as either loop or linear connection; and we established *current flow* by missing connections, redundant lines, redundant extra components, and crossed

Fig. 1 Student pair e-textile design for the letter "E" of school sign (*left*, full design; *right*, close-up of LilyPad and three LEDs)

lines/short circuits (for more detail, see "Designing a Functional Circuit (Pre and Post)" section). Taking account of these three aspects, we also coded whether or not students' drawn circuits would function properly, that is, powering a light with the battery (1—yes, 0—no). Due to the small sample size and binary coding scheme, we conducted Mcnemar's tests to determine significance of changes in students' understanding for each of the circuitry features.

Reading a Codeable Circuit Design (Pre and Post)

The structure of task 2 was modeled after existing research on assessing computer science knowledge—namely, if students can *read* or interpret an existing computer program (Guzdial 2015; K–12 Computer Science Framework 2016). However, this task moves beyond that skill through addition of another component: a codeable circuit. Students were thus provided text of a simple Arduino program (that could make an LED blink continuously) *and* a simple circuit design that included one LED and the LilyPad Arduino microcontroller, which were both printed on a single piece of paper (see Fig. 3). It should be noted here that this code was a basic pattern most students learned in their previous year's e-textiles project. We asked students two questions: "Do you have a sense of what the function of this code might be?" and "Do you think this code would work with the following LilyPad circuit, why or why not?" Answers were recorded on video and then transcribed.

Drawing from existing research on student understanding of programming concepts (Guzdial 2015; Soloway and Spohrer 1989), the qualitative coding scheme for this task focused on students' understanding of the basic computer science concepts including (a) functions, (b) fundamentals, and (c) looping. Reading through the transcripts, we coded each specific feature as 1 (present) or 0 (not present). We established *functions* by looking at whether or not students explained the two basic components of a blink: (1) "digitalWrite" as a function that could turn an LED on or off (through the values "HIGH" and "LOW"); and (2) "delay" as a command that controls time. We determined computing *fundamentals* by seeing at whether or not they discussed (3) where and how variables were named, and (4) what the inputs/outputs were for the program (e.g., LED as an output). We also ascertained *looping* by mention of (5) continuous or repeated action of the commands. Finally, we coded for whether or not students could determine (6) whether the program as a whole matched the circuit design shown (for more detail, see "Reading a Codeable Circuits Design (Pre and Post)" section). After coding students' answers, each student received a number ranging between 0 and 6, with a higher score reflecting a more thorough understanding of the code. Due to the small sample size and continuous coding scheme, we conducted a Wilcoxon two-sample paired signed ranks test on these data to determine significance in changes in students' understanding.

Designing, Reading, and Remixing a Codeable Circuit (Post Only)

Only occurring *after* the unit, the structure of task 3 is a more complex extension of both tasks 1 and 2. Whereas task 1 required the construction of a simple circuit and task 2 required

Table 1 Data sources for circuit and coding tasks

Task	Timepoint	Data type	Data source	Coding task
1	Pre/post	Artifact	Simple circuit diagram	Designing
2	Pre/post	Transcript	Explanation of codeable circuit	Reading
3a	Post	Artifact	Advanced circuit diagram	Designing
3b	Post	Transcript	Explanation of codeable circuit	Reading
3c	Post	Transcript and artifact	Remixed code + explanation	Reading + remixing

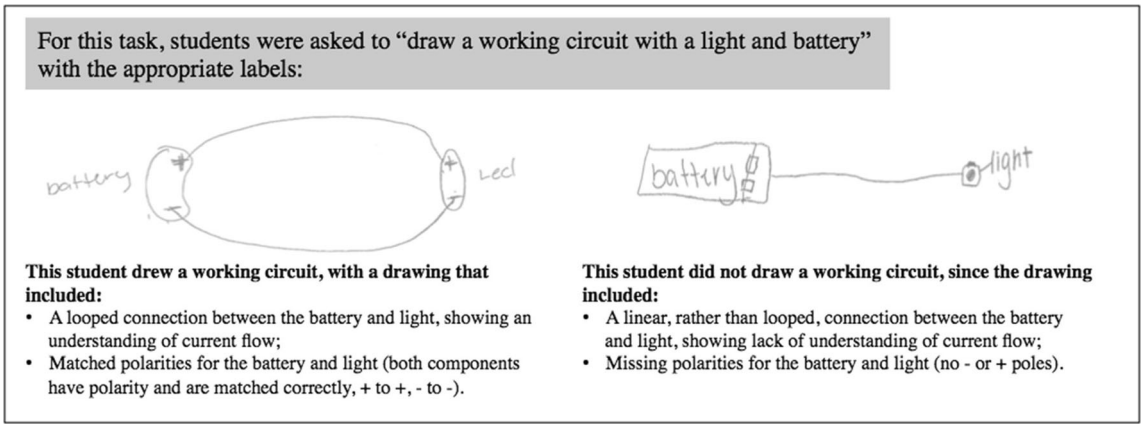

Fig. 2 Examples of students' drawing of a functional circuit (*left*) and a non-functional circuit (*right*) diagram (Task 1)

students to read a program for the LilyPad Arduino, task 3 additionally asked students to complete these tasks at a more integrated and advanced level, since it asked them to design, read, and remix a codeable circuit. In other words, rather than just being asked to engage with a circuit or a program, students are required to work on these simultaneously within this task—a specially designed circuit that can be controlled via code.

The task was split up into three separate parts. In task 3a, students needed to design a circuit with four given components (a LilyPad, two LEDs, and a switch) that corresponded with a pre-existing program. In task 3b, students needed to read and explain this given program, which was notably more complex than the program given for task 2 since it was switch-based and included a conditional statement (if-then-else) and custom functions (see Fig. 4). In task 3c, students needed to remix parts of this code to change the behavior of the circuit from blinking asynchronously to blinking synchronously. Throughout the tasks, students responded to questions either verbally or by writing down their answers (including drawings, comments, and code), and these data were video-recorded, transcribed, and documented accordingly.

The analytic coding scheme for the circuit design (task 3a) focused on students' ability to design a codeable circuit based on an existing program. Modifying the coding scheme for task 1 (drawing a simple circuit), we coded 1 (present/correct) or 0 (not present/incorrect) for three specific features: (a) connection, (b) polarity/grounding, and (c) current flow. In terms of *connection*, we looked at how each component (two LEDs and one switch) was connected to the LilyPad. As with task 1, this required a looped rather than a linear connection, but additionally required a connection to a particular LilyPad pin as identified in the code (as opposed to just the + pole). In order to successfully complete this task, students have to be able read the variable declaration section within the Arduino code, and identify the correct pin for the correct component. In terms of *polarity/grounding*, we examined two elements: whether or not each individual component was correctly grounded (somehow connected to the negative pole of the

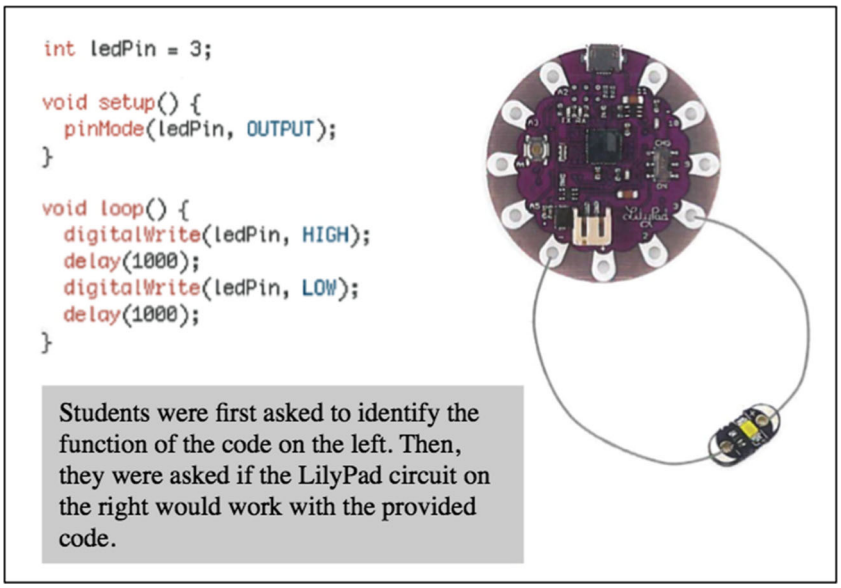

Fig. 3 Codeable circuit task (Task 2)

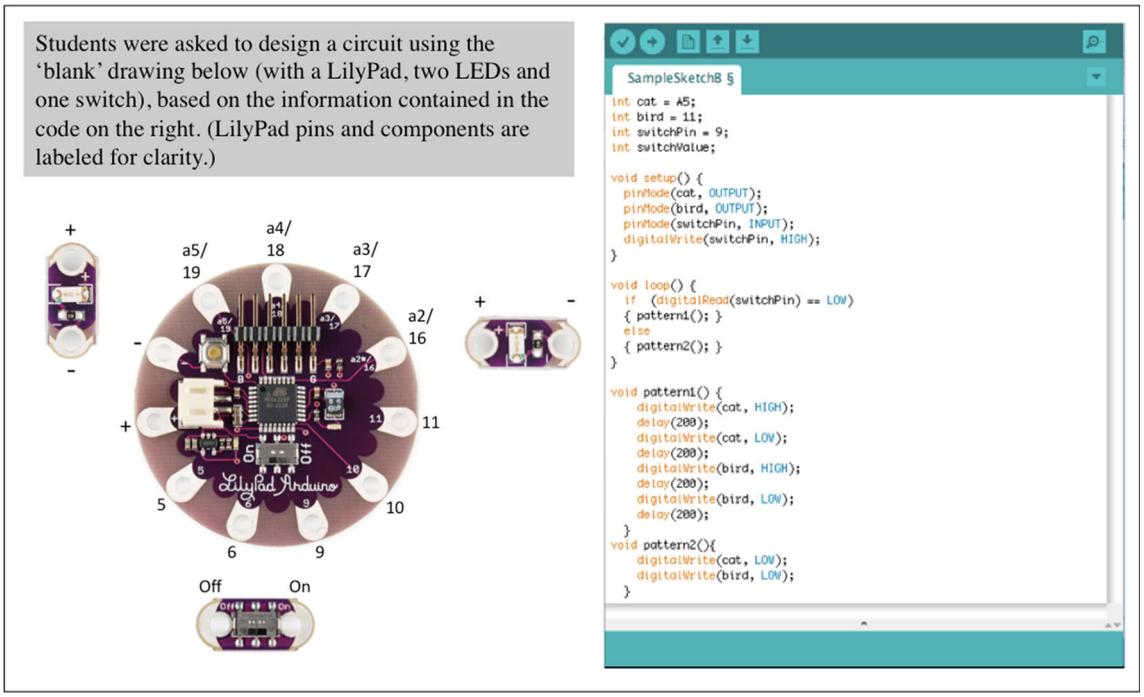

Fig. 4 Designing a working codeable circuit task: LilyPad with two LEDs and switch component (*left*) and Arduino code (*right*) (Task 3a)

LilyPad); and the techniques of grounding, specifically looking if components were as follows: directly connected to negative pin of the LilyPad; connected to each other; connected to a negative line; or connected to other non-named pins. Finally, we examined *current flow* by checking for missing connections, redundant lines, and crossed lines/short circuits. We also coded (1—yes, 0—no) if the circuit as drawn would indeed work with the existing program (for more detail, see "Designing a Codeable Circuit" section).

For task 3b (see Fig. 4), we coded for students' ability to read the given switch and sensor-based program using the same coding scheme as task 2 (see "Reading a Codeable Circuit Design (Pre and Post)" section. Again, this focused on student demonstration of basic computer science concepts including the following: *functions* including (1) digitalWrite as a function that would turn an LED on or off; (2) delay as a command relating to time; *fundamentals* including (3) declaration of variables; (4) setting up inputs/outputs of the program; and *looping*, that is (5) repetition of action. However, instead of asking whether or not the code would work with a given circuit design (something that could not occur since students designed this circuit themselves), we additionally scored students about their ability to (6) understand the conditional if/then commands within the program—a more complex computer science concept (Soloway and Spohrer 1989) that was not present in the more basic program presented in task 2 (for more detail, see "Reading Program for a Codeable Circuit" section). This coding scheme resulted in each student receiving a number ranging between 0 and 6, with a higher score reflecting a better understanding of the program.

Finally, the coding scheme for task 3c was based on students' ability to remix given lines of the code to shift the program output from asynchronously blinking LEDs to synchronously blinking LEDs. Drawing from existing standards in computer science education, this task looked at how well students understood the *control structures* of the program, that is, the sequential ordering of steps within a program that leads to different behaviors or outcomes (K–12 Computer Science Framework 2016). Most student verbally described, rather than typed or wrote, what they would change. As a result, the coding scheme only captures the general revision required for this remix rather precisely capturing these changes (e.g., with the correct syntax and/or exact ordering). Students were therefore scored based on the following appropriate revisions: (1) move together the "digitalWrite HIGH" lines, (2) move together the "digitalWrite LOW" lines, and (3) move or eliminate the "delay" lines from the program (for more detail, see "Remixing Program Code for a Codeable Circuit" Section). The maximum score for this task was three points. Again, because of the nature of this coding scheme, the students who gained the highest score (3) would still have needed to make minor revisions for the correct behavior. However, their changes would have been close enough to keep them on the right track.

Findings

We first present results of students' ability in designing a functional circuit. We also highlight particular circuitry concepts in which students demonstrated their greatest

improvements. Second, we share results regarding students' ability in reading a codeable circuit. Finally, we describe results of students' ability to design and remix a codeable circuit after participating in the e-textiles unit during which they designed similar codeable circuits.

Designing a Functional Circuit (Pre and Post)

In Task 1, we first sought to determine whether students significantly improved their ability to draw, label, and explain a working circuit diagram (see Fig. 2). Like in previous studies (Peppler and Glosson 2013; Halverson et al. 2016), we found that students' ability to draw a working circuit significantly increased ($p < .05$, $p = .000$) from pre- to post task. Specifically, after the e-textiles unit, 78% (18) of students were able to draw a functional circuit design whereas only 26% (6) were able to do so before the unit.

Furthermore, we examined what elements of a circuit were the biggest areas of improvement (see Table 2 for an overview) and found that students significantly improved their understanding of matching polarity ($p < .05$, $p = .002$). Students also significantly increased their understanding of circuitry as a loop ($p < .05$, $p = .004$). In terms of common mistakes, students significantly reduced the number of missing connections they had in their circuit prior to the unit ($p < .05$, $p = .021$), whereas redundant connections and short circuits did not appear to be major issues either before or after the class. These results confirm findings from prior studies (Peppler and Glosson 2013; Halverson et al. 2016) demonstrating that students can learn about circuits as loops in making activities.

Reading a Codeable Circuits Design (Pre and Post)

In Task 2, we also investigated students' ability to understand the relationship between circuit design and program code by explaining a given code and diagram (see Fig. 3). In this task, the Arduino code generates a basic blinking pattern, turning on and off the LED attached to the LilyPad with a repeated delay of 1000 ms.

We found that students' ability to read the code to control a circuit improved after the e-textiles unit: while only one student in the pre-task got a score between 4 and 6, indicating high understanding, over 56.5% (13) of students reached this level in the post-task. The average score on the pre-task was 1.13 ($N = 23$) and on the post-task it jumped to 3.35 ($N = 23$) indicating that students significantly improved in their ability to decipher a program and circuit design (see Table 3 for a student example). A Wilcoxon two-sample paired signed ranks revealed that the median post-assessment scores were significantly higher than median pre-assessment scores ($Z = -3.7364$, $p < .01$).

A more nuanced understanding of students' knowledge about codeable circuits was revealed through examination of the different aspects of the program that were coded. The class as a whole made the highest gains in understanding the digitalWrite function (a command that could turn an LED on or off through the values "HIGH" and "LOW"), with 16 more students demonstrating this knowledge after the unit than before (from 2 to 18 students). In general, this is not surprising. The digitalWrite function is the most readily observable aspect of the code, since it literally controls whether or not the LEDs attached to a LilyPad turn on or off. Almost all the students had experience manipulating this function within their own project programs in order to cause different LED behaviors. On the other hand, other aspects of the program (the looping, the fundamentals such as declaration of variables and inputs/outputs) were coded less often within this analysis. The review of the interview answers revealed one reason why this occurred: these sections, while essential to the function of the program, generally seemed less tangible than the digitalWrite command and often taken for granted within people's answers. For instance, many students who provided brief descriptions describe the function (e.g., it makes the LED blink) tended not to explicate each part within their answers. From this perspective, student understanding of codeable circuits seems distinct from student understanding of on-screen code; it appears that tangibility of concepts can become key influence here aiding student learning.

Table 2 Changes in students' understandings of circuit features (Task 1)

Circuit feature	Pre ($N = 23$)	Post ($N = 23$)	Significance
Polarity (matched)	7	19	.002*
Connection type (loop)	14	23	.004*
Current flow			
Missing connections	9	1	.021*
Redundant connections	1	0	1.000
Crossing lines/short circuits	3	0	.250
Circuit functionality	5	18	.000*

*Significance differences at the $p < .05$ level

Table 3 Pre/post interview excerpts for single student (Task 2)

Pre-Interview Answer of One Student for Simple Codeable Circuit Task (SCORE = 0 out of 6)	
INTERVIEWER: I don't know if you've ever seen code or programs before, but this is a program. It's code that does something, so if you were looking at that do you have a sense of what that does? STUDENT: No. INTERVIEWER: Okay, that's fine. And that is actually the stuff we would be working with, and this [points to circuit diagram] might potentially work with this [points to code], but… I guess you don't know if this works with this [points to circuit diagram again] cause you don't know what this is? [points to code again] STUDENT: [shaking head] No.	This student scored a zero on the pre-task because she was unable to identify any parts of the code when asked about its function. Additionally, she could not identify if the circuit would work with the code as written.
Post-Interview Answer from Same Student for Simple Codeable Circuit Task (SCORE = 6 out of 6)	
INTERVIEWER: So do you have any sense of what the function of this code might be, like what it's doing? STUDENT: It's turning it on and off I think. INTERVIEWER: And can you go through it? Can you take me through it? STUDENT: This is **the name of the pin [points to 1st line of program]. And that's on 3 [points to pin 3 on the LilyPad circuit]**. And **this is an output, so it turns on I think [points to 2nd line of code]** and then **high is on** and the light is 1000 [points to 1st line of code in the void loop section], and **then low is off, the light is off** [points to 3rd line of code in void loop section] and the light is 1000 [points at LED]. So it says on, goes off and then **it loops, so it keeps doing that.** INTERVIEWER: And so would it work with the circuit? STUDENT: Yeah.	The same student scored a five on this post-task, since she was able to identify several parts of the code (bolded), including: • the naming of the LilyPad pins (+1); • the input/output section of the code (+1); • that 'HIGH' and 'LOW' mean on and off, respectively (+1); • that 'void loop' indicates a repeated action (+1). However, she was *not* able to identify: • that 'delay' is a command that controls time (+0). Additionally, she was able to correctly identify whether the circuit would work with the provided program (+1).

Designing, Reading, and Remixing a Codeable Circuit (Post Only)

The last task (3a, 3b, and 3c) focused on designing, reading, and remixing of a codeable circuit, thus giving us better insights into students' understanding of the relationship between software design and the circuit design for the LilyPad. As compared to the simple circuit design in task 1, this design was more complicated because it included more components (two lights and a switch) as well as a microcontroller with multiple possible sites for connection. Additionally, compared to the basic blink program provided in task 2, this program was more complex since it contained a conditional statement (if-then-else) with a switch, control of multiple LEDs, and custom light pattern functions (see Fig. 4).

Designing a Codeable Circuit

In Task 3a, we examined students' ability to design a working circuit that matched components such as two LED lights and switch with a pre-existing program, which includes a conditional statement (if-then-else) using the switch and custom functions (see Fig. 4). As with task 1, this challenge required students to design a grounded circuit that facilitated current flow. However, students were additionally required make connections between the different components and the LilyPad that corresponded to the given program.

In terms of connections, most students were able to create looped connections for each component (one connection each from the positive and negative ends): 91.3% (21) of students did this for the LEDs and 78.2% (18) did this for the switch. Students were also able to connect the positive end of the components to the correct LilyPad pin based on their understanding of the program: 78.2% (18) of students were able to properly connect the "Cat" LED to pin A5, 82.6% (19) were able to properly connect the "Bird" LED to pin 11, and 87.0% (20) were able to properly connect the switch to pin 9. In Fig. 5, the top drawing illustrates a student's correct design of a codeable circuit.

However, students had more difficulty connecting the negative pole of each component appropriately in order to properly ground the circuit: 73.9% (17) of students were able to

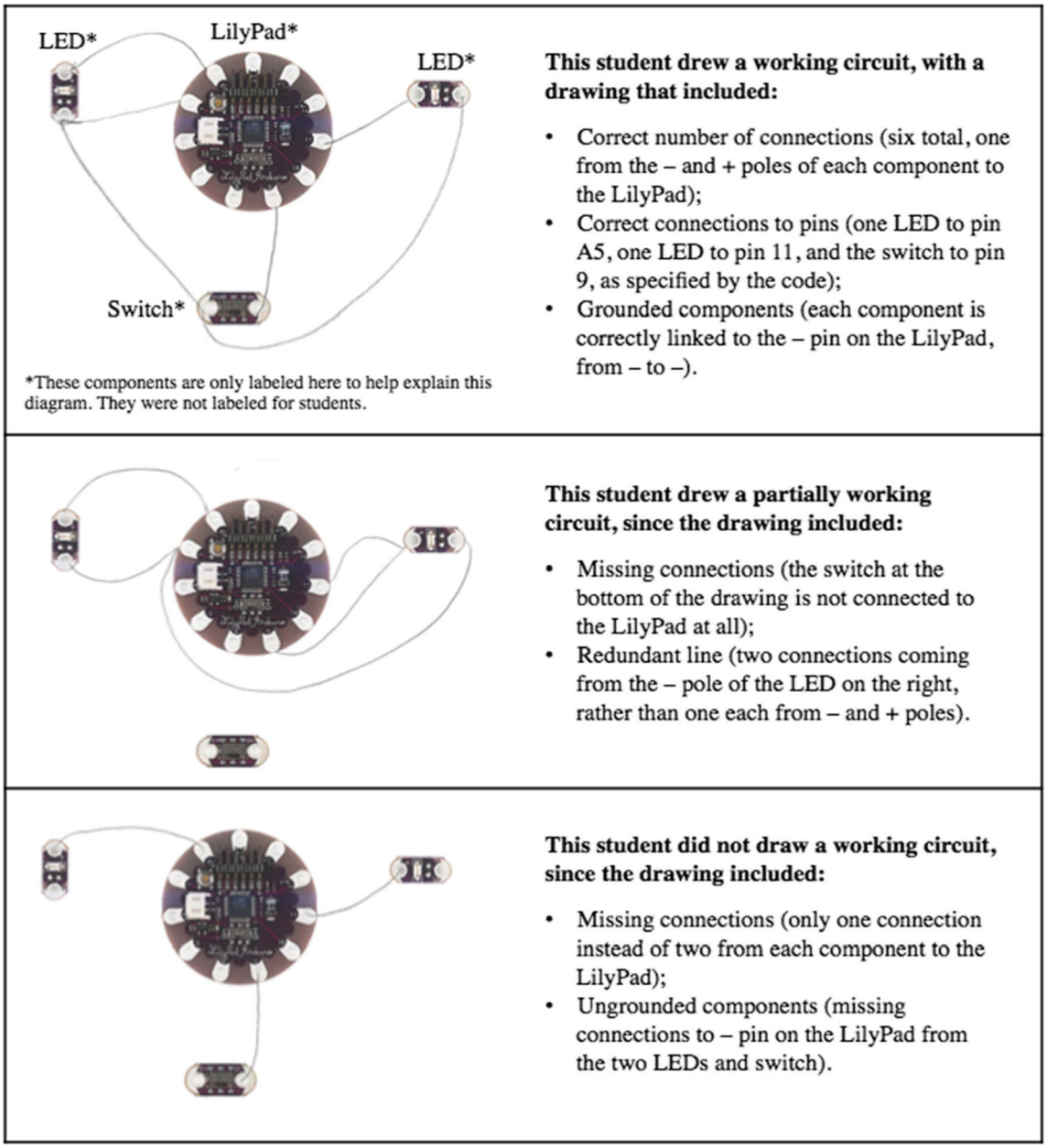

Fig. 5 Student drawings of codeable circuits design: functional design (*top*), partial (*middle*), and non-working (*bottom*) designs (Task 3a)

ground the "Cat" LED, 65.2% (15) of students were able to ground the "Bird" LED, and 52.2% (12) of students were able to ground the switch. In terms of their techniques for grounding, we found that they accomplished this using one or more methods including the following: 73.9% (17) of students by connecting them directly to the negative pin of the LilyPad, 34.8% (8) of students by connecting to another component (which was eventually connected to the negative pin), and .04% (1) student by connecting it to negative connecting line. It should be noted here that the latter two techniques are considered more advanced, since they are an efficient use of thread and sewing. Interestingly, 17.4% (4) of students attempted to ground components by connecting them to undesignated pins on the LilyPad. While this approach would not have worked with the pre-existing program, further analysis of the interviews illustrated how this resulted from conversations that we had in the unit about this particular technique of writing extra code in order to program a pin negative.

Finally, we analyzed whether or not the whole configuration of a students' designed circuit would ultimately work with the pre-existing program as intended. Overall, 39.1% (9) of students created a circuit that would fully work with the existing code, meaning that it all the intended actions written into the program (for the two lights and the switch) would function. In some cases (6 students, or 26.1%), students' circuit diagrams would work for some but not all components, so we did not rate these as working. Regarding these mistakes, 39.1% (9) of students were missing connections within the entire circuit and 8.7% (2) of students had redundant lines. However, as with task 1, none had crossed lines or short circuits. The middle drawing of student in Fig. 5 showcases an example a partially working codeable circuit with a redundant line and missing connections (marked as non-working), while the drawing on the bottom included linear rather than looped connections (which indicates missed connections), and was thus not appropriately grounded.

Reading Program for a Codeable Circuit

In Task 3b, we asked student to read and explain program code that controlled a LilyPad Arduino and components. Students' overall average score for this task was 3.45 ($N = 22$) out of 6 with a distribution as follows: no students scored a 0, 4.5% (1) of students scored a 1, 31.8% (7) scored a 2, 18.2% (4) scored a 3, 13.6% (3) scored a 4, 22.7% (5) scored a 5, and 9.1% (2) scored a 6 indicating the highest level of comprehension. As with task 2, students demonstrated the greatest understanding of the function digitalWrite (with the values of high and low) as corresponding to turning the LEDs on and off (95.7% or 22 of 23). Again, this is not unexpected since almost all the students had opportunities to deal with this function in their own program code, and experienced its tangible output (lights going on or off). However, students demonstrated the lowest understanding of the conditional statement (if-then-else) statements (39.1% or 9 of 23 students). While this concept was fundamental to the outcome of students' projects (since it controlled their switch function), their lack of understanding can be likely explained by the pair working arrangements of the class. As described within the methods section of the paper, many students divided their work according to domain. While some students focused more on crafting and circuit design, others focused more intensely on coding and were thus exposed to more complex aspects of the program (i.e., the conditional statement, inputs/outputs).

A closer examination of the answers themselves additionally revealed a more complex picture about student understanding of this code. Students' answers varied in specificity which also made it challenging to develop a consistent coding scheme. For example, some students, whom we knew were proficient at coding, gave succinct answers which demonstrated an overall understanding of the program (i.e., that turn the switch on and off would change the light pattern). However, they may have not provided a step-by-step breakdown of this action, and therefore did not describe individual functions such as delays, digitalWrite, or loops. On the other hand, some students who we knew were less proficient at coding were generally more explicit in describing these individual features. While they received scores that may have demonstrated high understanding of the program, they could not ultimately synthesize these into a comprehensive explanation. Thus, the scores did not always reflect a level of understanding of computational concepts at-large but instead students' abilities to recognize individual aspects of code.

Within this task, we also noticed students' abilities in linking together their understanding of the two different modalities of circuitry and coding. Specifically, this involved how they used their understanding of the *text* of the code to design the *visual* circuit (seen in the previous task 3a), and how they used their understanding of the *visual* circuit to interpret the *text* of the code (seen in this task). When students were asked to explain the provided Arduino code, more than half of them (12 of 22) ended up physically pointing to different parts of the LilyPad circuit that they had designed as part of their answer. For example, this can be evidenced by this interview excerpt, where a student is reading from the code and explaining her answers, while simultaneously pointing toward the drawn circuit diagram: "This BIRD LED [points to right LED in circuit diagram] comes on then it goes off [points to pattern 1 text in code]." While we cannot empirically confirm that their understanding in one area depended on the other area, their use of the circuits as an explanatory tool seems to indicate their developing fluidity moving in between the realm of software (the code) and hardware (the circuit).

Remixing Program Code for a Codeable Circuit

In Task 3c, we investigated students' ability to remix existing program code to change the behavior from synchronous to asynchronous blinking for a codeable circuit. We rated students on whether or not they made the following changes in the program code: (1) move together the "digitalWrite HIGH" lines, (2) move together the "digitalWrite LOW" lines, and (3) move or eliminate the "delay" lines from the program (see Fig. 6). Students' average score was 2.09 ($N = 22$) distributed as follows: no one scored 0, 22.7% (5) scored 1, 31.8% (7) scored 2, and 40.9% (9) scored 3, the highest score. As stated earlier, this task primarily engages with students' understanding of the sequential ordering of steps within programs to cause different outcomes, or the control structure of the program. Most students indicated that they would move the "digitalWrite HIGH" lines or "digitalWrite LOW" lines together (19 or 86.4%, and 16 or 72.7%, respectively), but only half indicated an interest in moving the delays (11 or 50%). While this finding indicates that these students would not be able to exactly rewrite the code to achieve the desired behavior change, we still argue that this does indicate an understanding of the role of sequencing steps to change outcomes. This can be demonstrated, for instance, by one students' answer to the question of how to modify the existing code make the "Cat" and "Bird" lights blink together (rather than one at a time): "[The] Cat and Bird would both be right here [points to first two digitalWrite lines of the code] and they'd both be high. Then Cat and Bird would be right here [points to second digitalWrite lines] and they would both be low." Here, the student does not mention the necessity of shifting the delay functions within the code, but seems to indicate that pushing the two Cat and Bird high lines together (indicating a function that turns both lights on), and the two Cat and Bird low lines together (indicating a function that would turn both lights off) would mean that both lights would be blinking at the same time. From this perspective, while this student (and others) would not have the exact code needed for the behavior change, her impulse to move these lines together would put on her on the right track to accomplishing this goal, while still missing some details. Because of the large percentage of students that did indicate an interest in this shift (86.4 and 72.7%, respectively for the high and low lines), we assert that our finding indicates that students had developed a basic understanding of control flow, or the sequencing of lines within a program to shifts its outcome.

Summary of Findings

After completing the e-textiles unit, students significantly increased not only their ability to design a functional circuit, but also to design and understand the functionality of a codeable circuit. Codeable circuits are a unique type of design in which both the blueprint of the circuit design and the control

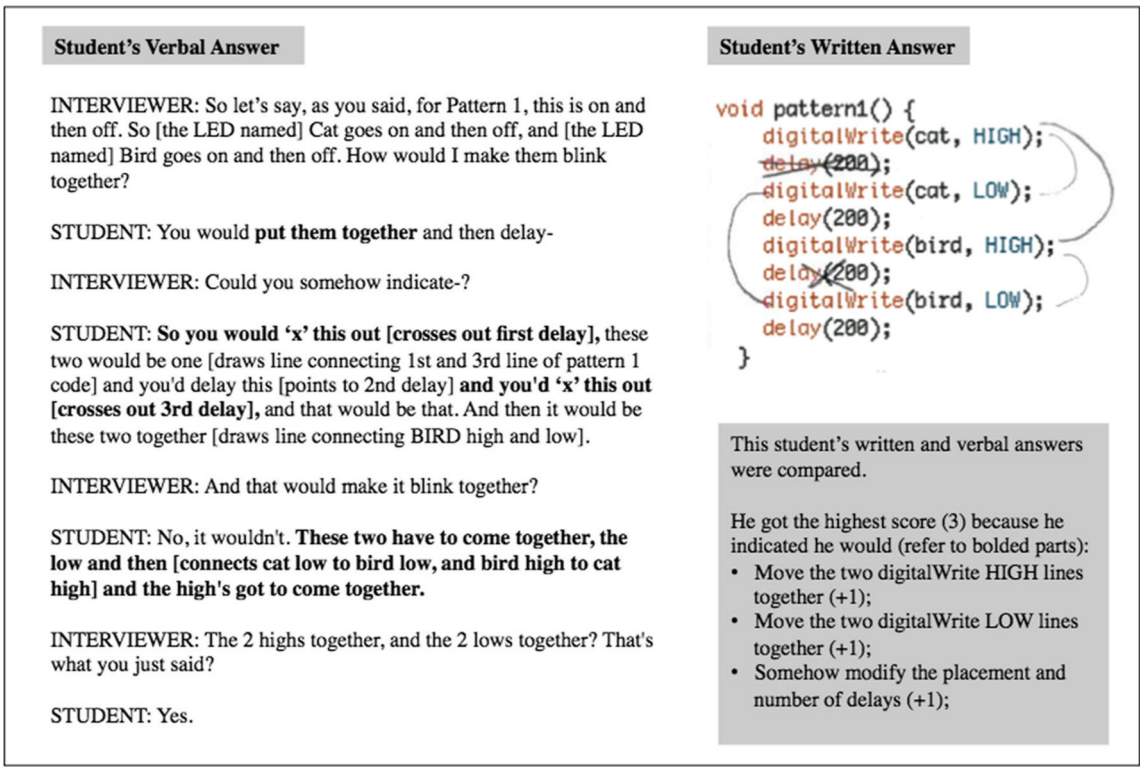

Fig. 6 Sample of student answer for remixing code task (Task 3c)

structure of the code must align in order for the LED, sensors, and switches to perform desired behaviors, hence, making circuit codeable. In this study, students not only exhibited increases in reading a codeable circuit, but also developed more sophisticated skills in designing and remixing codeable circuits in post tasks. The codeable circuit tasks successfully captured students' ability to read, design, and remix code at a basic level, and task 3 allowed us to tease apart these different areas while guiding students through a natural progression of difficulty.

Discussion

As digital and maker technologies proliferate our lives, there is a growing interest in integrating computer science and engineering education in K-12 curriculum. The crux of our study is built on the understanding that as new technologies are born out of this movement we must investigate the new learning that follows. Specifically, with e-textiles, learning happens at the unique intersection of crafting, circuitry, and coding thus providing a promising example for the type of integrated STEM learning called for in national reports and initiatives (Katehi et al. 2009; Smith 2016). Building on previous research that showcased students' increased understanding of circuit designs, we extended this understanding by adding software design. The research on students' understanding of codeable circuits presented in this paper sits at the intersection of engineering and computing—two key STEM topics. In the following sections, we discuss the opportunities and challenges in understanding and assessing this type of intersectional learning in maker activities.

Understanding STEM Learning in Maker Activities

Successfully working with codeable circuits requires a grasp of both circuit design and coding, which makes these types of circuits particularly challenging for students crafting e-textiles projects. In our study, most students read and some students successfully designed and remixed codeable circuits after completing a complex e-textiles project. This means that most students understood the interconnectedness of circuit design and coding enough to interpret it, but only some were able to apply this knowledge by producing new code. Since about half the students took on the role of "coder" in their pair, this could explain the differences between student abilities in the post-tasks. Thus far, only college students have demonstrated learning this intersectional knowledge found in codeable circuits (Sadler et al. 2016), and our study sheds light on high school students' learning in this area.

Recognizing the integrated nature of maker activities, we suspect that the disciplinary knowledge is interdependent in many of these activities. Put differently, that understanding one disciplinary area (e.g., circuitry) could be supportive of learning other areas (e.g., coding or design). We have some evidence supporting this, specifically, when students made references to circuit diagrams when discussing their program code, thus providing an answer that integrated both visual and textual elements. This dimension of STEM learning can be especially challenging when students need to connect concepts from different disciplines, such as engineering and computing, with which they are often unfamiliar and which are rarely introduced together. The situated, multimodal, and interdisciplinary learning we investigated in this study is precisely the form of STEM learning for which maker scholars are arguing.

By the same token, the opportunities presented by these maker activities also pose significant challenges to teachers and students. To support these interdisciplinary activities in the classroom, teachers will likely need to gain some familiarity with the intersecting disciplines and/or develop interdisciplinary collaborations with their peers, which are typically siloed in traditional schooling settings. Here we need to investigate how teachers can provide differentiated support for students often at various completion points of their projects. Another way to support student learning is through designing technologies that support interdisciplinary and multimodal learning. In the context of e-textiles, for example, Modkit, a visual programming software for the LilyPad Arduino (Millner and Baafi 2011), is one case of integrating the visual diagrams with the coding process. The trade-off with this specific tool, however, is that it eliminates the more authentic computer science practice of text-based coding. We need additional work in this area focused on designing and developing tools and resources to support the interdisciplinary, multimodal learning which maker activities require.

Developing Assessments for Maker Activities

Maker activities like e-textiles are different from more traditional science or computing activities primarily due to their interdisciplinary nature, which also make them especially difficult to assess. Peppler and Glosson (2013) and Halverson et al. (2016), whose findings we replicated in this study, have collectively worked to design assessment tools to capture a particular aspect of engineering knowledge—namely circuitry—in maker activities and contexts. Their assessments tools, though, primarily look at designing simple circuits. However, during this e-textiles unit, we pursued more complex interdisciplinary projects involving codeable circuits, which required understanding of multiple domains, namely the visual design of a circuit along with the written construction of a program. The order of tasks mirrors the order in which we taught the unit; while all the students were required to design simple circuits, fewer were required to learn how to read and remix code because of the pair structure of the course. The

particularly of the tasks is something that might limit our ability to employ these tasks in other contexts. For this study then, we worked to develop codeable circuit assessment tools, which represent our initial efforts to capture this interdisciplinary knowledge and skill.

Within our design, we incorporated elementary circuit design assessments with various computational practices such as reading, writing, and debugging code. Besides the first free drawing circuit design task, the other tasks presented in this paper were designed in order highlight the integrated nature of the activity. They highlighted this aspect in two ways. First, the visual presentation of the tasks was multimodal; that is, students were simultaneously presented with the text of a program alongside a LilyPad circuit design, they did not see one without the other. In this way, the presentation reinforced their integration. Second, while the tasks themselves focused on one mode over another (e.g., asking students to design a circuit, or asking them to explain or remix the code), the most correct answers depended upon their understanding and ability to manipulate both modes simultaneously. Thus, our design of these tasks, while preliminary, works to expand the growing body of assessment tools in the maker literature with an eye toward its interdisciplinary, multimodal nature.

Along with the affordances of the codeable circuit tasks described above, we should also note some of the constraints of these activities. In terms of implementation, we relied on paper and verbal responses to tasks rather than relying upon the more true-to-life modes of the screen or craft-based circuits, thus potentially yielding inauthentic responses and/or answers. Here we suggest developing true-to-life e-textile debugging activities as described by Fields et al. (2012) that engage students in debugging code and circuit designs in more tangible and real-world maker contexts. Additionally, another potential area of growth involves our interview protocol in this study. Because it was open-ended and follow-up questions varied across interviews, this sometimes resulted in inconsistent responses that might have captured more student articulateness but not always their understanding. For future work, we plan to explore both more structured interview protocols and screen-based tasks to accommodate realistic time constraints.

Future Research

The present study contributes to the growing body of research exploring learning in maker activities and designing activities and tasks to assess that learning. In previous work, we investigated the collaborative dimensions of maker activities by examining how student pairs can leverage distributed expertise to mitigate the challenges of these tasks. By working in pairs, students were able to rely on each other's strengths to troubleshoot and debug problems in various disciplines. Building on this work, we should continue explore how to best leverage collaborative learning arrangements as possible solutions to support learning in maker activities. Here we should also focus on the design of collaborative tasks themselves as a way to support and scaffold students' learning (Litts et al. 2015).

Moreover, though our primary focus in this study was on circuitry and coding pieces of the maker activities, we believe similar successes and challenges exist at other disciplinary intersections (e.g., between design and crafting). Thus, our work expands maker designs in STEAM by highlighting the significance of learning at the points of intersection between disciplines, where we can focus future research. Likewise, expanding our conceptions of learning requires that we also expand our notions and tools of assessment. The codeable circuits tasks we present in this study is a first example of how interdisciplinary, multimodal assessments could look like. Taken together, our study lays a foundation for future work to capture learning and rethink assessments in maker activities implemented in K-12 settings.

Acknowledgements This work was supported by a grant (#1509245) from the National Science Foundation to Yasmin Kafai, Jane Margolis, and Joanna Goode. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation, Utah State University, or the University of Pennsylvania.

References

Blikstein P, Kabayadondo Z, Martin A, Fields D (2017) An assessment instrument of technological literacies in makerspaces and fabLabs. J Eng Educ 106(1):149–175

Buechley L (2006) A construction kit for electronic textiles. In: The proceedings of the 10th IEEE international symposium on wearable computers

Buechley L, Hill BM (2010) LilyPad in the wild: how hardware's long tail is supporting new engineering and design communities. In: The proceedings of the 8th ACM conference on designing interactive systems

Buechley L, Eisenberg M, Elumeze N (2007) Towards a curriculum for electronic textiles in the high school classroom. In: The proceedings of the 12th annual ACM SIGCSE conference on innovation and technology in computer science education

Duit R, von Rhöneck C (1998) Learning and understanding key concepts of electricity. In: Tiberghien A, Jossem EL, Barojas J (eds), Connecting research in physics education with teacher education. International Commission on Physics Education

Fields DA, Searle, KA, Kafai, YB, Min, HS (2012) Debuggems to assess student learning in e-textiles. In: The proceedings of the 43rd ACM SIGCSE technical symposium on computer science education. ACM press, New York

Fredette N, Lochhead J (1980) Student conceptions of simple circuits. Phys Teach 18(3):194–198

Goode J, Margolis J, Chapman G (2014) Curriculum is not enough: the educational theory and research foundation of the Exploring Computer Science professional development model. In: Proceedings of SIGCSE'14. ACM, New York, pp 493–498

Grover S, Pea R (2013) Computational thinking in K-12: a review of the state of the field. Educ Res 42(1):38–43

Guzdial M (2015) Learner-centered design of computing education: research on computing for everyone. Synthesis Lectures on Human-Centered Informatics 8(6):1–165

Halverson E, Lawson C, Fleuchaus E, Bakker M, Saplan K (2016) Learning circuitry through making. In: The 2016 American educational research association annual meeting

Honey M, Kanter DE (eds) (2013) Design, make, play: growing the next generation of STEM innovators. Routledge, New York

Johnson S, Thomas AP (2010) Squishy circuits: a tangible medium for electronics education. In: The CHI 2010 proceedings, pp 4099–4104

K–12 Computer Science Framework (2016) Retrieved from http://www.k12cs.org

Kafai Y, Fields D, Searle K (2014a) Electronic textiles as disruptive designs: supporting and challenging maker activities in schools. Harv Educ Rev 84(4):532–556

Kafai YB, Searle KA, Fields, DA, Lee E, Kaplan E, Lui, D (2014b) A crafts-oriented approach to computing in high school: introducing computational concepts, practices and perspectives with E-textiles. Trans Comput Educ 14(1):1–20

Katehi L, Pearson G, Feder M (eds) (2009) Engineering in K-12 education: understanding the status and improving the prospects. National Academies Press, Washington, DC

Kock ZJ, Taconis R, Bolhuis S, Gravemeijer K (2013) Some key issues in creating inquiry-based instructional practices that aim at the understanding of simple electric circuits. Res Sci Educ 43(2):579–597

Linn M (2003). Technology and science education: starting points, research programs, and trends. International Journal of Science Education, 25(6):727–758.

Litts BK, Kafai YB, Dieckmeyer E (2015) Collaborative electronic textile designs by high school youth: Challenges and opportunities in connecting crafts, circuits, and code. In: The proceedings of the 2014 ACM SIGCHI FabLearn conference on creativity and fabrication in education. Seattle, p 381–386

Millner A, Baafi E (2011) Modkit: blending and extending approachable platforms for creating computer programs and interactive objects. In: The proceedings of the 10th annual ACM SIGCHI conference on Interaction Design and Children

Osborne R (1981) Children's ideas about electric current. New Zealand Science Teacher 29:12–19

Osborne R (1983) Towards modifying children's ideas about electric current. Research in Science & Technological Education 1(1):73–82

Peppler K, Glosson D (2013) Stitching circuits: learning about circuitry through e-textile materials. J Sci Educ Technol 22(5):751–763

Peppler K, Halverson E, Kafai YB (eds) (2016a) Makeology: makerspaces as learning environments (Volume 1). Routledge, New York

Peppler K, Halverson E, Kafai YB (eds) (2016b) Makeology: Makers as learners (Volume 2). Routledge, New York

Qi J, Buechley L (2014) Sketching in circuits: designing and building electronics on paper. In The proceedings of the 2014 ACM SIGCHI conference on human factors in computing systems

Sadler J, Shluzas L, Blikstein P (2016) Building blocks in creative computing: modularity increases the probability of prototyping novel ideas. International Journal of Design Creativity and Innovation 3(1):1–17

Shepardson DP, Moje EB (1994) The nature of fourth graders' understandings of electric circuits. Sci Educ 78(5):489–514

Shipstone DM (1984) A study of children's understanding of electricity in simple D.C. circuits. Eur J Sci Educ 6(2):185–198

Smith M (2016) Computer science for all. Retrieved from https://www.whitehouse.gov/blog/2016/01/30/computer-science-all

Soloway E, Spohrer JC (eds) (1989) Studying the novice programmer. Lawrence Erlbaum, Hillsdale

Stern B, Cooper T (2015) Getting started with Adafruit FLORA: making wearables with an Arduino-compatible electronics platform. Maker Media, San Francisco

Tofel-Grehl C, Litts B, Searle K (2016) Getting crafty with the NGSS: using crafty circuits to teach electricity in elementary school. Sci Child 54(4):48

Tofel-Grehl C, Fields DA, Searle K, Maahs-Fladung C, Feldon D, Gu G, Sun V (2017) Electrifying engagement in middle school science class: improving student interest through e-textiles. J Sci Educ Technol

Wing J (2006) Computational thinking. Commun ACM 49(3):33–35