1-6-2023

Toward a Debugging Pedagogy: Helping Students Learn to Get Unstuck With Physical Computing Systems

Colin Hennessy Elliott
University of Colorado Boulder

Alexandra Gendreau Chakarov
San Jose State University

Jeffrey B. Bush
University of Colorado Boulder

Jessie Nixon
Utah State University, jessie.nixon@usu.edu

Mimi Recker
Utah State University, mimi.recker@usu.edu

Recommended Citation

Hennessy Elliott, C., Gendreau Chakarov, A., Bush, J.B., Nixon, J. and Recker, M. (2023), "Toward a debugging pedagogy: helping students learn to get unstuck with physical computing systems", Information and Learning Sciences, Vol. 124 No. 1/2, pp. 1-24. https://doi.org/10.1108/ILS-03-2022-0051

Toward a Debugging Pedagogy: Helping students learn to get unstuck with physical computing systems

Journal:	*Information and Learning Sciences*
Manuscript ID	ILS-03-2022-0051.R2
Manuscript Type:	Article
Keywords:	Debugging Pedagogies, Computational Thinking, Interaction Analysis, Science Education, Physical Computing, Situated Inquiry

Dear Reviewer,

Thank you for your detailed review that pushed us to clarify our methods for analysis, our sociocultural theories of learning, and the implications this case study has on future research and design. We have addressed each of your comments and detail how in the table below.

Comment	**How we addressed it.**
One suggestion I have for the authors is to cite a seminal work when they point to sociocultural theories of learning on p.2 of the paper. It would be great if they could also be more precise about the aspect of the sociocultural theory they are drawing on for this paper. The authors can take this opportunity to situate interaction analysis within the theoretical framework guiding this study.	We agreed that there was more possibility to flesh out what we meant by sociocultural theories of learning using citations and to connect it to our choice of IA as a method. We did this on page 3, stating "From a sociocultural perspective attending to the resources and developing relations (Cole, 1996; Nasir and Hand, 2008; Vygotsky, 1987), we used interaction analysis (Jordan and Henderson, 1995) to analyze her work with students as they debugged their systems." Citing Cole (1996) Nasir and Hand (2006), reference the kinds of sociocultural theories we are drawing on, but our addition of text make it clear that we are attending to resources and relations as part of analyzing the moment-to-moment using IA. We further expanded this by bringing in Esmonde (2017) on page 9 and 14, where we describe analytic methods.
I would appreciate it if the authors could have a visual representation to clarify the corpus of data they had and the specific videos/episodes they chose for reporting the findings. The visualization could even be a table with cells shaded. That would help the readers understand and situate the selected episodes within the broader class context.	We understood that our analytic methods, especially the choices we made to go from the data corpus to the moments shared in the paper, needed further clarity. We created tables 2 and 3. Table 2 (p. 14) shares the details of video data corpus of both fo Gabrielle's implementations and references the class period that the focal episodes come from. Table 3 (p. 15) outlines our steps in analysis and the sources we used. we feel these two tables, along with the additional text that cites them, help the readers understand and situate the episodes within the broader context of our data.
"Episodes were then reviewed repeatedly by the group of researchers together and independently, iterating on the transcript and a working depiction of Gabrielle's practice each time." It would be better if the authors could be more precise here and indicate the size of the research team, the frequency of the meeting, and the relationship between the authors of the paper and this research team. We know that the second author participated in the analysis and contributed to data collection, but what about	We added language that shared 1) the research team working this data consisted of the five authors on the paper (p.14), and 2) author 2 was the only member of the team who spend a significant amount of time with Gabrielle in professional development settings and in her classroom, while the other four authors have worked on the larger project (p. 16) but participated directly in the analyses for this paper.

other authors and researchers? How are they related to the study, particularly the analysis?	
The third finding reported as "positioning as a learner" still feels very lean compared to the earlier two subsections. Can the authors draw more evidence to support how the teacher positioned herself as a learner? Right now, most of the argument is built on observational data. I wonder how other data sources add to this or raise further questions.	We added references to other data sources that supported our claim that Gabrielle positioned herself as a learner with her students, both on page 26. This included sharing a quote from her addressing her whole class during the second implementation where she describes the tutorial supporting their programming as "designed... for people like me who are just clueless -- which is me with coding..." Further we added a quote from her interview after the second implementation where she stated that her experience with coding wasn't "the best" but she could support students. Both of these further flesh out Gabrielle's ways of positioning herself as a learner (first instance) and her reasoning why.
The authors can better articulate the implications of the findings for future research and practice. The authors conclude with remarks on Gabrielle's approach to debugging "with" students but without specific connections to how this can inform future design and research efforts	We really appreciated this push to broaden our implications. On page 30, we added two sentences that explicitly state implications for designing professional development for computer science educators (in formal and informal settings), particularly those new to physical computing. In conjunction with our final paragraph that calls for further research which explores how teachers new to physical computing use and develop skills for supporting their students, this shares a broad set of implications from the case study.

Toward a Debugging Pedagogy:
Helping students learn to get unstuck with physical computing systems

Structured Abstract

Purpose

The purpose of this paper is to examine how a middle school science teacher, new to programming, supports students in learning to debug physical computing systems consisting of programmable sensors and data displays.

Approach

This case study draws on data collected during an inquiry-oriented instructional unit in which students learn to collect, display, and interpret data from their surrounding environment by wiring and programming a physical computing system. Using interaction analysis, we analyzed video recordings of one teacher's (Gabrielle) pedagogical moves as she supported students in debugging their systems as they drew upon a variety of embodied, material, and social resources.

Findings

We present Gabrielle's debugging interactional grammar, highlighting the pedagogical possibilities for supporting students in systematic ways, providing affective support (e.g., showing them care and encouragement), and positioning herself as a learner with the students. Gabrielle's practice, and therefore her pedagogy, has the potential to support students in becoming better debuggers on their own in the future.

Originality

While much of the prior work on learning to debug focuses on learner actions and possible errors, our case focuses on an educator's debugging pedagogy centered on the educator ***debugging with*** the learner. This case study illustrates the need for educators to exhibit deft facilitation, vulnerability, and orchestration skills to support student development of their own process for and agency in debugging.

Toward a Debugging Pedagogy:
Helping students learn to get unstuck with physical computing systems

Physical computing in middle school science

There is considerable and growing demand to offer computing education to youth of all ages. One promising approach, called physical computing, enables youth to pursue personally relevant design tasks using a combination of hardware and software tools (e.g., using an Arduino or micro:bit with block programming or text coding). With the growing use of physical computing in classrooms, libraries, and other STEM learning spaces, educators need to learn to help students, especially as they debug hardware and software issues that inevitably arise (DesPortes and DiSalvo, 2019). Many educators serving youth, including middle school teachers, often have little experience with computing, programming, or physical computing systems (Warner *et al.*, 2019), which leads to challenges for them in supporting youth when they get stuck working on their systems. Teachers at the K-8 level especially find obstacles in supporting students in becoming "unstuck" (Haduong and Brennan, 2019).

In this paper, we describe a teacher's practice using a sensor-based physical computing system, called the Data Sensor Hub (DaSH), developed for middle school science and STEM inquiry instructional units. Using the BBC micro:bit – an increasingly common physical computing technology used in K-8 schools which debuted in 2016 (Sentance *et al.*, 2017) – and associated hardware, the DaSH enables students to programmatically analyze and display data collected from a variety of environmental sensors.

A collection of researchers and educators have codesigned instructional units (Gendreau Chakarov, et al., 2021), including one to introduce middle school students to the DaSH (Biddy, et al., 2020) as a tool to support inquiry in their science and STEM classes. During this introductory unit, students learn about the capabilities of the DaSH by creating their own physical data displays. During these activities, students often encounter various problems during this process and ask their teacher for help in getting "unstuck."

To illustrate how educators support students in instances of debugging, we present a case study of one teacher, Gabrielle[1], who was new to physical computing and programming as she implemented the introductory unit with two different science classes of 25 students aged 11-13 years. From a sociocultural perspective attending to the resources and developing relations (Cole, 1996; Nasir and Hand, 2006; Vygotsky, 1987), we used interaction analysis (Jordan and Henderson, 1995) to analyze her work with students as they debugged their systems. Analysis highlights the pedagogical possibilities for: 1) systematically, and multimodally, supporting students to notice bugs and possible fixes, 2) providing affective support by showing care and encouragement, and 3) positioning oneself as a learner of programming. In this way, we illustrate how Gabrielle's approach has the potential to support students in becoming better debuggers on their own rather than simply helping students fix the issue at hand.

This paper heeds Myer's (2019) call for research on teachers integrating computational thinking (CT) into their disciplinary classes. We articulate how one teacher responded to the "need [for teachers] to be confident enough with technology to open pathways for learners, even if they are never expert coders themselves" (Myers, 2019, p. 261). Learning from the routines and practices of one novice teacher, we argue for developing an explicit pedagogy for debugging within the adaptation of CT across various learning spaces (e.g., library makerspaces, afterschool programs, and formal science classrooms).

Our case study analysis is guided by the research question: *how does a teacher, new to programming and physical computing systems, facilitate students learning to debug their DaSH systems?*

Prior work on debugging

Prior research on students learning to program has characterized: 1) typical programming bugs (e.g.,, Ahmadzadeh *et al*., 2005), 2) novice misconceptions that lead to bugs (e.g. the superbug; Grover and Pea, 2013; Pea, 1986), and 3) successful debugging approaches for learners (Kinnunen and Simon, 2010; Lewis, 2012) and teachers in professional learning (Kim *et al*., 2018). Developing debugging skills is a core component of computer science (Grover and Pea, 2013) and computational thinking (Blikstein and

[1] All names in this study are pseudonyms

Moghadam, 2019; Brennan and Resnick, 2012; Webb, 2010). Building on computer science education literature, we refer to debugging as a two-part process of finding bugs, or sources of experienced breakdown in the system, and then fixing them (McCauley *et al*., 2008). Errors in programming contexts (bugs) result from complex interactions between the user, environment, and technology (Ko and Myers, 2005). Therefore, finding bugs is a significant challenge in learning programming skills (Fitzgerald *et al.*, 2008). McCauley *et al.*'s (2008) thorough literature review of debugging research suggests that an explicit focus on debugging is beneficial for students learning to program; in other words, debugging can be taught.

Finally, getting stuck, and unstuck, is an emotional journey especially for youth who are learning to program and work with physical computing systems for the first time. Programming is a dynamic process intersecting problem solving, emotion, and identity, especially for youth from underrepresented groups in computer science (Dahn and DeLeima, 2020). Scholarship has addressed this by: creating bugs for students to figure out that are not part of their own computational work (Fields *et al*., 2016; Morales-Navarro *et al.*, 2021; Webb, 2010), facilitating explicit curricular designs that ask students to reflect on the emotional journey (Dahn *et al*., 2020, DeLiema *et al*., 2019), and supporting the local material and social resources teachers bring to developing inclusive computing learning environments (Shaw, *et al*., 2020).

Physical Computing Systems

Physical computing systems provide an especially challenging context for computing novices because bugs can occur in any part of the system: the software (or program), hardware, or hardware-software connections (DesPortes and DiSalvo, 2019; Jayathirtha *et al*., 2020). Yet compared to software-only programming environments, physical computing systems can decrease the conceptual distance between the abstraction of programming and students' experiences of the material world.

Physical computing systems are becoming more accessible for use in primary and secondary education classrooms because of user-friendly interfaces and decreasing costs (Anastopoulou *et al*., 2012; Blikstein, 2013; Blikstein and Moghadam, 2019). Generally, physical computing offers an alternative set of introductory activities to engage students in computing and broaden their definition of computer science as a discipline (Kafai *et al*., 2014). For example, electronic textiles (e-textiles) provide an opportunity to integrate computer science and art through the fabrication of textiles (such as clothing) enhanced with electronics (Buechley *et al.*, 2008; Peppler, 2013). As another example, programmable sensors allow students to collect their own large streams of data to use in their own investigations (Hardy *et al*., 2020). Sensors can make the invisible visible by allowing students to "see" the phenomenon they are investigating (e.g., using biometric sensors to visualize the internal function of the organs of the body [Norooz *et al*., 2015]).

A focus on the teacher role

The curriculum teachers use, pedagogical choices they make, and their professional learning opportunities are each integral to the kinds of experiences their students are afforded with physical computing (e.g., Witherspoon and Schunn, 2019). Interviewing teachers and students (ages 11-12) who used the BBC Micro:bit early on in classrooms, Sentance *et al*. (2017) articulated three teacher types who integrated physical computing into classroom teaching with distinct theories of learning: inspirers, providers, and consumers. Inspirers described offering open-ended exploration in line with their pedagogical beliefs, providers facilitated students learning to program with specific resources and activities that aligned with their espoused pedagogical beliefs, and consumers described using the tutorials developed and available on the micro:bit website with their students with no real mention of their pedagogy. Although not explicit, approaches to learning how to debug inform these overall approaches to integrating physical computing.

Prior work has described environments conducive to help students develop debugging skills. For example, Rich and colleagues (2019) developed a learning trajectory for K-8 debugging where students are asked a series of questions to help them locate and fix a bug in a Scratch program. In physical computing, Fields *et al*. (2019) describe a strategy for developing debugging skills as having students fix programmable electronic textiles with planted bugs in their hardware and software. These examples focus on what students are doing during the debugging process. Less research has examined how teachers support their students, specifically in developing their own debugging processes and strategies. In one example, Solomon *et al*. (2020) have shown that embodied moves are key learning resources for novice programmers because they support a representation of the abstract.

A portion of the computer science education literature focuses on debugging as a practice "that needs to be taught" (McCauley *et al*., 2008). However, pedagogies and theories focusing on how teachers support students as they navigate both the process and the affective dimensions of debugging (e.g., how students learn to persevere when challenged) are more limited (Dahn and DeLiema, 2020: DeLiema *et al*., 2019). Prior research has investigated the promise for learning during moments of breakdown (Ko and Myers, 2003), terming it "productive failure" (Kapur, 2008; Kapur and Bielaczyc, 2012). When physical computing systems break down, students have the opportunity to become better debuggers if supported in important ways (Kafai *et al*., 2019). When teachers support a greater focus on the process of debugging rather than the product (i.e., a debugged and functional program), they are putting into practice the conceptualization of debugging as a core component in learning computer science, not just as a means to an end (Kafai *et al*., 2020).

Thus, there is a need to better understand how teachers, many of whom lack robust computer science expertise, draw upon their own pedagogical expertise to orchestrate groups of students as they engage in debugging tasks and help them develop perseverance and problem-solving skills while working to become "unstuck." This paper contributes to a conceptualization of pedagogies for teachers supporting students in debugging by exploring the interactional routines of a teacher's practice across two implementations of an eight-to-ten-day unit.

Conceptual Framework: Teaching debugging as a multimodal situated inquiry

We conceptualize debugging a physical computing system as a situated inquiry (Flood, *et al*., 2020; Suchman, 1987) where participants develop an iterative process of understanding (Miyake, 1986) as they tinker with the software and hardware, leading to developing perspectives on the system as a whole, something often hard for novice programmers (Vessey, 1985). From this perspective, we view the sources of error, or bugs, as interactionally produced (Flood *et al*., 2018) when recognized by participants in the debugging process not as static issues that are there to be found. Jayathirtha *et al*. (2020) describe debugging physical computing artifacts as navigating "the complex system of tools and representations" (p. 1048) that encompass the physical computing system and available external supports. In our case study, we focus on one teacher's practices supporting students' work navigating the complex system when they reach stuck points, noticing that their sensor-based computing system is not working as expected.

When a teacher joins a group of students debugging, they use gestures and movements, in conjunction with discourse, to support developing particular relations to the physical system, the software, and to debugging as a practice. Thus, debugging is a multimodal interactional achievement (Goodwin, 2018; Keifert, 2020) where embodied representations, as well as discourse, are essential for learning. In the context of computing, "embodied representations are likely some of the few resources students can use to understand the abstract" (Solomon, *et al*., 2020, p.2139). Goodwin (2018) referred to these gestural representations as "environmentally coupled gestures" where the gesture becomes part of the interactional landscape for communicating with others.

From a sociocultural perspective (Cole, 1996; Nasir and Hand, 2006; Vygotsky, 1987), learning to debug is the process of developing skills and practices in and through producing social and material relations. Teaching debugging as a multimodal situated inquiry, teachers support students in navigating the complexity of tools, representations, and relations by providing various kinds of help including offering useful resources, guidance, support, and productive questions. In our analysis we define the resources teachers offer to students as embodied (Solomon, *et al*, 2020) material (e.g., color coded wire-diagrams, the MakeCode programming interface) and non-material (e.g., peer and teacher relationships, prior experiences and content knowledge, disciplinary relationships) (Shaw, *et al*., 2020).

Technology, Instruction, and Information Context

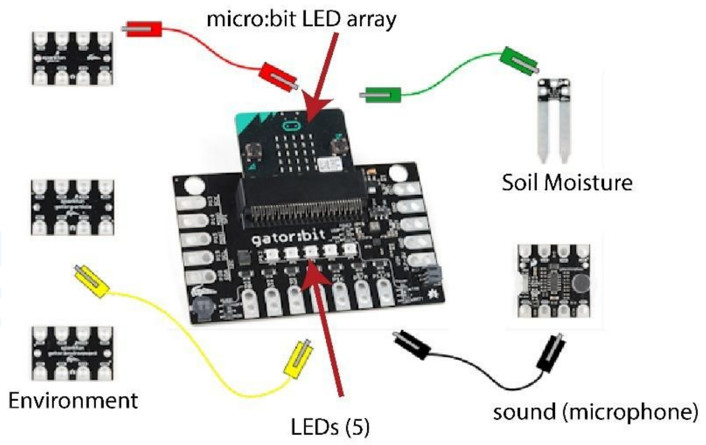
Data Sensor Hub (DaSH)

The Data Sensor Hub (DaSH; Figure 1) is a physical computing system composed of a BBC micro:bit (Sentance, *et al*., 2017), a gator:bit which makes additional programmable pins on the micro:bit easily accessible for alligator clip connectors, and a variety of alligator clippable sensors. We have codesigned multiple units with participating teachers that integrate the DaSH into middle school science and STEM instruction. In a codesigned introductory unit, called the Sensor Immersion Unit (SIU), students interact with a teacher-created DaSH that utilizes three sensors to display environmental data about their classroom over the course of five lessons that, in the case of our focal teacher, took 8 class periods of 55 minutes. After a series of inquiries into what the sensors can do and how to program them, students, in groups, construct their own DaSH that uses one or two sensors to display information from data collected in their environment.

The SIU facilitates students using three alligator-clippable sensors including an environment sensor (that measures temperature, humidity, barometric pressure, carbon dioxide, and total volatile organic compounds), a soil moisture sensor, and a sound (microphone) sensor. Students program the micro:bit using MakeCode (Devine *et al*., 2018), a browser-based programming interface that allows students to program in both blocks and text (JavaScript or Python) and upload their program from the website to their micro:bit (described further in Gendreau Chakarov *et al.,* 2021).

In the SIU, students are provided with material resources to support their engineering of their own systems. These resources include unit-specific programming tutorials developed in MakeCode, color coded diagrams that illustrate how to wire individual sensors as part of the DaSH, different color alligator clip wires, the labeled connection ports on the gator:bit and the sensors, and unit slides that describe different uses of the sensors.

Figure 1.
The Data Sensor Hub (DaSH)



Note: *The DaSH is composed of a micro:bit (top center), gator:bit (middle), and a set of alligator clippable sensors that can measure different environmental conditions such as temperature, noise level, or UV level. The micro:bit and gator:bit support the display of information through onboard LEDs and a speaker.*

Stuck Points

While completing tutorials (with accompanying figures and diagrams) and creating their versions of the DaSH as physical data displays, students invariably encounter challenges where the system does not perform as expected. Students become stuck when faced with these bugs and, often, look to the teacher for help in fixing the system. Drawing on a sociocultural approach to analysis (Esmonde, 2017), we view bugs as interactionally produced (Flood *et al*., 2018) when initially recognized by either the student, teacher, or both. Students, or their teacher, communicate the presence of bugs – and approaches to fixing it – through talk, gesture, and coordination of various resources. Further, they identify and rectify bugs by testing and changing the hardware, the software (i.e., their MakeCode program), and the connections between them. Jayathirtha *et al*. (2020) describe this as coordinating the "multi-representational space" of hardware, software, and students plans for the system.

In another study (Gendreau Chakarov, *et al.,* 2021), we identified several stuck points that emerged while students created physical data displays in three categories: hardware, hardware-software, and software, aligning with DesPortes and DiSalvo's (2019) characterization of bugs for novices using Arduinos. Table 1 provides a synopsis of the categories, example bugs, and resources provided in the unit.

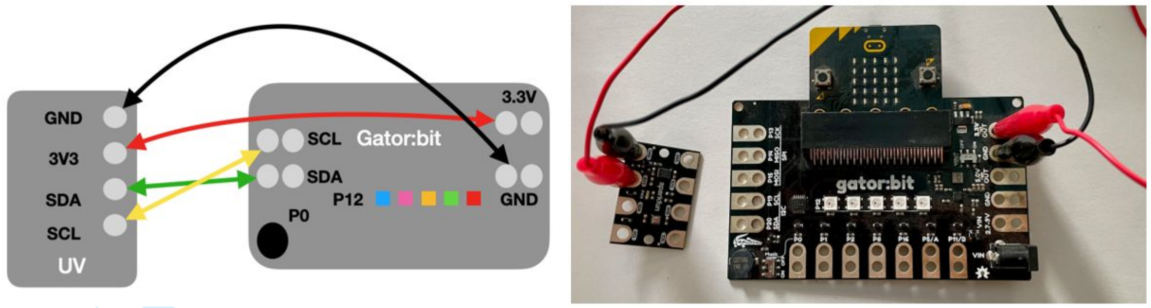
Table 1.

Sources of error, bug descriptions, and resources provided to support students with them

Stuck Point	**Example Bugs Description**	**Material Resource(s) Provided to Students**
Hardware	Students struggle to attach the alligator clips securely or forget to use all the required wires, often only wiring the power and ground and forgetting the data transfer wires (see **Figure 2**).	● Wiring Diagram ● Gator:bit and sensor port labels
Software	A point of confusion for many students revolves around what value from the sensor is being used in a conditional block. For example, as shown in the code in **Figure 3**, when the micro:bit's button A is pressed and students make a loud noise (such as yelling), they expect to see the sound level displayed (e.g., "850"), followed by an "X". The next value they see is a smiley face because in this program the sound intensity value is retrieved before the conditional statement is executed.	● Built in programming tutorial with example code ● Color coding and shapes of MakeCode blocks
Hardware-Software connection	Errors occur in transferring the program from the computer to the micro:bit, which involves dragging the file from the computer to the micro:bit icon on the computer, similar to how a file is copied to a thumb drive. Students regularly thought they had uploaded the program onto the micro:bit but had not completed this step correctly. Another issue was the student's code references the wrong data pin for a sensor.	● Visual Download button, with popup directions, on MakeCode browser site. ● Embedded description of "get moisture on Pin___… with power on pin___" on MakeCode block. Both blanks are dropdowns.

Figure 2.

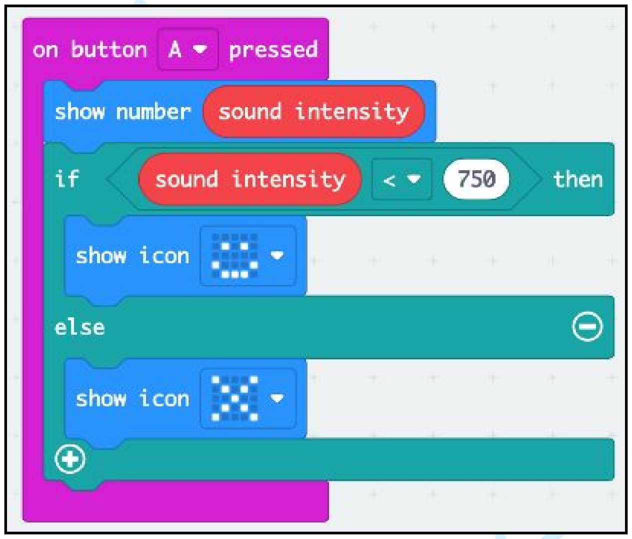
Example wiring error causing hardware bug.



Note: The image on the left shows the wiring diagram used by students to wire the UV sensor. The image on the right shows a common issue that students encounter where they have only wired the power wire (red) and ground wire (black).

Figure 3.

Example programming error causing software bug.



```
on button A pressed
    show number sound intensity
    if sound intensity < 750 then
        show icon
    else
        show icon
```

Note: This code snippet represents a stuck point for students where the value displayed from the show number block is not the same value as used by the conditional logic statement.

Methods and Study Context

This study comes from a larger project that is collaborating with a large urban school district as part of a Research-Practice Partnership (Coburn *et al.*, 2013) started in 2017 to develop instructional units that integrate computing in middle school science and STEM classes (Gendreau Chakarov, *et al.*, 2020). The school district, located in the Western U.S. – which enrolls over 70% students from minoritized racial and ethnic groups, a majority of students who qualify for free and reduced lunch, and a large population of English-language learners – is committed to offering equitable access to computing opportunities. Teachers working with the project participated in ongoing professional learning to integrate the instructional units and the DaSH into their science classes (Gendreau Chakarov, *et al.*, 2020). Professional learning activities included collaboratively designing instructional units, modeling activities in these units, reflecting on implementation, and revising – all with the goals of supporting teacher learning, developing a teacher community, and iteratively refining instructional materials (Biddy, et al., 2021).

Participant

We observed and took ethnographic field notes for all ten participating middle school teachers as they taught the SIU during the 2019-2020 school year. We chose Gabrielle as a focal case because her practices stood out for multiple reasons. She was very articulate as she engaged with students in debugging the DaSH, making her thinking clear for her students as well as for analysis. Unlike other teachers, she often shared how much she struggled completing the computing tasks that she was asking them to do. Finally, her debugging approach with students seemed to follow a clear pattern that warranted further analysis. We saw an opportunity to learn her debugging pedagogy from her practice, particularly how she incorporated her expertise to support students in doing something she felt less confident in, and therefore decided to construct a case study, not as representative, but as an instrumental case (Stake, 1995). In doing so, we aim to understand the unique case (Stake, 1995) as we investigate Gabrielle's approach to debugging in her middle school classroom.

Gabrielle taught this instructional unit twice with two different 6th grade classes of 20 to 25 students (ages 11-12). This was Gabrielle's second year with the project. Before joining the project, Gabrielle had no prior programming experience. Gabrielle expressed during the professional learning workshops that she was a complete novice and often did not know what she was doing. Nevertheless, she always jumped into whatever physical computing project the professional learning group was working on and asked many questions of fellow teachers and researchers in order to create a successful program. She carried over this attitude into her teaching by frequently telling her students that she was still a beginner and learning along with them. This vulnerability proved to be a key part of her debugging pedagogy.

Data Sources

The majority of the case was developed from analyses of Gabrielle's classroom practice, which was audio and video recorded over the course of two classroom implementations with two distinct groups of 6th grade students with 45-minute class periods that met almost every day. This class period was a bonus period at the end of the day for a rotating group of students and therefore did not always meet depending on the schedule. The first implementation with one group of students was recorded over two weeks at the beginning of September 2019. Five class periods, approximately three hours and forty minutes, over the two weeks were recorded. The second implementation with a second group of students was recorded during two weeks in November and December. Eight class periods, totaling just over six hours, were recorded during this implementation. Recording during the second implementation focused on Gabrielle's interactions with small groups of students as they worked on the DaSHs. We reviewed the whole classroom video data corpus (approximately ten hours of classroom video), first logging all data where students worked on developing their DaSH systems, to characterize what we call her interactional grammar when interacting with students as they were debugging. The focal episodes presented in this study were recorded during one class period during her second implementation of the unit in November and December of 2019 (see Table 2). As this was Gabrielle's second implementation, she had begun to refine and revise her practices. Our process for sharing our findings for the case are bounded by this one class period for three reasons: 1) this was the first day of the SIU where students both wire and program one of the sensors, 2) there were extended periods where Gabrielle supported students in debugging and 3) Gabrielle's approach during this class period are reflective of her approach to debugging throughout the SIU.

Table 2.
Gabrielle's classroom video data

Implementation & student group	Month(s)	# of days	Total video recording length
1st	September 2019	5 days	~3.75 hours
2nd	November & December 2019	8 days*	~5.5 hours

*Focal episodes come from 7th day of 2nd implementation

Secondary sources of data were used to triangulate and confirm analysis of Gabrielle's practice. After an initial analysis of the classroom video data, we reviewed approximately three hours of video recorded discussions at related professional development sessions before and after the focal episode, concentrating on moments when Gabrielle shared her experience or discussed the unit. In addition, we reviewed two 45-minute reflective interviews with Gabrielle: one interview after her initial implementation and another after her second implementation. We used this information to learn more about Gabrielle's overall teaching philosophy and teaching practice.

Analytic Methods

We conceptualize learning to debug as a multimodal, embodied (Solomon, *et al*., 2020), and situated practice (Suchman, 1987 from Flood *et al*., 2020). Building on this sociocultural approach by "maintaining attention to context and relations among people" (Esmonde, 2017, p.7), the research team, made up of each of the authors, relied on interaction analysis (Hall and Stevens, 2016; Jordan and Henderson, 1995) to trace in-depth Gabrielle's contributions to interactions supporting students at small timescales. We began analysis with a corpus review of classroom video data of all ten teachers who taught the instructional unit during the 2019-20 school year, focusing on the lessons where students built and debugged their DaSHs. Table 3 shares our analytic steps to developing Gabrielle's case study starting from this phase.

Table 3.

The Steps of the data analysis process

Steps	Data Analysis	Data Sources
Step 1	Reviewed classroom data and chose Gabrielle as case study	Video recordings and field notes from ten classroom implementation of the Sensor Immersion Unit from 2019-2020
Step 2	Reviewed video of Gabrielle's implementation logging moments where she interacted with students while they were debugging	Twelve hours of video recordings from Gabrielle's two classroom implementations
Step 3	Conducted a fine-grained analysis of episodes from Gabrielle's implementation individually and as a research team	Video recordings of six debugging moments
Step 4	Created a flowchart and iterated through repeated viewings as a research team	Video recordings of six debugging moments
Step 5	Triangulated data individually and as a research team	Three hours of video recorded professional development discussions STEM experience surveys Two 45-minute reflective interviews after each implementation

After Gabrielle was chosen and both her implementations were reviewed, six episodes of her working with students (between 90s and three minutes) were selected for fine-grained analysis, transcribed (for talk and gesture) and analyzed across repeated viewings. All episodes selected for fine grained analysis were from Gabrielle's second implementation. Transcriptions (using modified transcription conventions from Jefferson, 2004) paid attention to talk, gesture, and interaction with the DaSH and instructional and informational materials at hand. These transcripts allowed us to perform fine-grained multimodal analysis of the interactions in Gabrielle's practice (Goodwin, 2018). Such adds rigor by attending to the dynamics between our research questions and interactions involving speech, gesture, and material resources. Episodes were then reviewed repeatedly by our research group together and independently, iterating on the transcript and a working depiction of Gabrielle's practice each time.

One researcher who participated in this level of analysis, Alex (author 2) supported Gabrielle during her time with the project and regularly visited her classroom during implementations. The other four researchers were involved in various stages of the project implementation and analysis but never worked directly with Gabrielle:

In individual and group analysis sessions (all five of us together) we traced patterns across episodes and focused on Gabrielle's moves to support students in deciding what to do next. Noting many of her debugging moments followed a similar discourse pattern during her second implementation, we then mapped her debugging interactional routines as a flowchart. Across many analysis sessions as a group of 5, we developed multiple iterations of the flowchart while repeatedly reviewing all six selected episodes. This included bringing individual conjectures to the group to iterate on. Further, we also constructed the three main components of Gabrielle's approach to working with students in these episodes, which are shared in the findings.

Once a working analysis was constructed, we triangulated with the secondary data sources (video of professional learning sessions, STEM experience surveys, and post-implementation interviews) to confirm our characterizations of Gabrielle's practice. We also reviewed classroom moments not selected for fine grained analysis in relation to the map of her debugging routines and our other findings.

To share our analysis in this paper, we selected two focal episodes to exhibit Gabrielle's practices at the scale of individual interactions to clearly articulate her coordination of talk, resources, and embodied moves. These episodes represent our findings from analysis of all of Gabrielle's debugging interactions with students across both implementations of the unit.

Both representative episodes described below come from a 55-minute class period during Gabrielle's second implementation of the unit where students were instructed to develop versions of the DaSH that use one sensor. For students, this entailed creating programs on Makecode (with scaffolded directions), wiring the sensors to the gator:bit to create their version of the DaSH (using diagrams), and testing and fixing any errors. The episodes are also successive, depicting Gabrielle's ability to shift between different students with different errors using a very similar approach. In just under six minutes, Gabrielle performed two supportive sets of interactions (each an episode) in quick succession with students who asked for help because their DaSHs were not performing as expected.

Findings: A debugging pedagogy

During a professional development workshop that occurred as she was implementing the SIU with her second group of students, Gabrielle articulated her overarching pedagogical goal to support students in developing debugging skills and strategies. She described the importance of helping students conceptualize the environmental conditions that sensor data represent "so that as they are troubleshooting their systems, *they can figure it out on their own…*" (emphasis is ours). She later added that her group of students in her second implementation were not doing well with debugging on their own, something that surprised and challenged her. She echoed this in her reflective interview after the implementation, sharing that students were not good at, what she called, "struggling through it" (Interview, January 2020). Analyzing her moves at the moment-to-moment level below depicts how her goal of supporting students developing their own debugging skills – and at pushing through the struggles of debugging – unfolded in the classroom.

Working with students as they debugged their systems, Gabrielle tailored her contributions to each situation while also developing an interactional grammar, or a repeated approach, that began during her first implementation and continued during her second. In this section, we describe three foundational aspects that characterize Gabrielle's interactions with students, or the interactional grammar she developed: 1) a systematic routine including language and gestures that oriented students to possible issues in coordination with resources, 2) her regular overt affective responses to students navigating frustrations and joys, and 3) a consistent positioning of herself as a learner alongside her students of programming and debugging. These included offering specific kinds of resources for students to support their debugging including material resources (e.g., a color-coded wiring diagram) and relational resources (e.g., her own struggles with programming, emotional support and encouragement).

A multimodal systematic checking routine

Episode 1. This episode illustrates how Gabrielle's interactions with students were complex orchestrations of language, gestures, and resources to help them notice bugs and take action to fix them. In particular, Gabrielle found what she identified as an issue with the student's wiring and made specific moves to get a student (Emma) to notice it as well. Emma challenged Gabrielle's suggestion by demonstrating that the wiring was correct. Searching then for other sources of the error, Gabrielle eventually asked Emma to redownload the code in case the program was not actually on the micro:bit. Across this whole interaction, Gabrielle's talk and gestures are consistent with her typical practice for eliciting students to notice bugs and possible fixes.

At the beginning of the episode, Emma's micro:bit LED screen only showed numbers. She expected the DaSH to show either a smiley face or an X depending on the measured level of sound. She was stuck and asked her teacher to help. Stepping up to help, Gabrielle initially tested the DaSH, further assessing the expectation for it to show an X when she spoke loudly into the sound (microphone) sensor. Table 4 shows a transcript of the interaction, our interpretation of the transcript, and a few key images that illustrate Gabrielle's gestural work to help Emma notice a possible bug using the wiring diagram as a resource.

After testing the setup herself (Table 4, line 1), Gabrielle quickly moved through a software and connection check (lines 2 and 3), before checking the wiring. She clearly demonstrated her debugging process for Emma, first narrating what she was doing (i.e., "looked at our wires," line 6), then indicating when she found an issue with "uh oh" (line 7) and finally tapping the green wire connected to the gator:bit to indicate a wiring issue. Repeating her process for Emma, signified by saying "uh oh" again (lines 9 and 12), Gabrielle used exaggerated gestures to highlight what she was looking for and how she found a possible issue. She pointed directly to: 1) the green wire's connection on the gator:bit while saying "did you see this one" (line 9), then 2) the wiring diagram's green wire connecting the SDA ports on the gator:bit and sensor while saying "... there's an SDA with no green on it" (line 12). The gestures pictured with lines 9 and 12 and in Table 4 are successive environmentally coupled gestures (Goodwin, 2018) that perceptually bring together the wiring diagram and the physical setup to highlight a possible source of the issue and a resource for finding it. While the wiring eventually turned out not to be an issue (the other SDA port on the sensor was wired with the green wire), this move was indicative of Gabrielle's persistence – and ingenuity – for helping students develop a vision for noticing issues in hardware and software.

Table 4.

Episode 1 transcript

#	Speaker	Transcript	Interpretation
1	Gabrielle	Hello Hello Hello::: (1s) Hello Hello Hello::: Ok seven fifty then.	Gabrielle tests the sound sensor, then begins to read the code from the students computer.
2		Ok seven fifty then. Ok this code looks r::ight.	Confirms the code is correct.
3		(inaudible) Ok you downloaded this?	Asks if the program she is looking at is on the DaSH they just tested.
4	Emma (s)	Mmhm=	
5	Gabrielle	=And it's on your micro:bit. Ok.	
6		So then we looked at our wires. Ok Three-V-Three. ((picks up the sensor as Emma says something to her)) ((Drops sensor and picks it back up)) What.	Starts reviewing the wiring, particularly on the sensor.
7		Ut oh. ((puts sensor down and grabs the gator:bit connection; **Taps the SDL connection with the green wire with left index finger**)). ((grabs the wiring diagram)).	Notes she has observed an issue with "uh oh" and then proceeds to help Emma notice it by first **touching the wire she perceives as the issue**.
8		((tone changes)) All of a sudden my computer came on. Ok.	Addresses another student talking to her from across the room, indicated by tone change.
9		((tone changes back)) Ok uh oh. Now that's a tricky one. Do you see this one ((**touches SDA gator:bit connection**)). Ut oh.	Reiterates that she has noticed an issue ("uh oh") and **then points out the wire connected to the gator:bit's SDA port** (pictured).

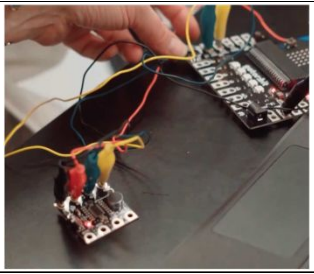
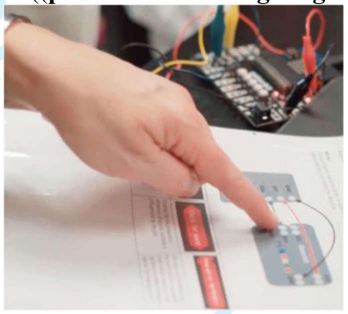
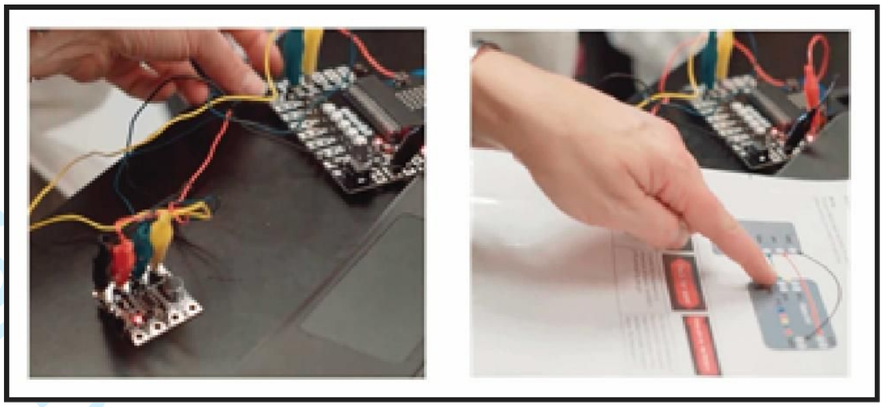
			
10		Your green's s-- wait are you environmental. Or sound.	Asks which sensor this is in order to see which **wiring diagram** to use in showing the student what to look at.
11	Emma (s)	Sound.	
12	Gabrielle	Ut oh. ((**points to the wiring diagram**))  Look the green says SDA to SDA. ((Picks up the sensor)) Did you do that. Oh no there's a SDA with no green in it. ((sets down sensor))	Gets Student 1 to notice what she sees as the issue by **pointing out the connection on the wiring diagram, modeling how to use the diagram in future debugging**. She shows the student that the green wire should be connected "SDA to SDA" from the gator:bit to the sensor. She then tells the student that the sensor has an SDA port without the green attached, therefore this must be the source of the issue.
13		Double check your wires again ok sweetheart. No big deal. It's not a big deal. Are you OK.	Directs student to check their wires and fix the issue. Offering some support, "no big deal."
14	Emma (s)	((picks up the sensor)) MMhmm.	
15	Gabrielle	((slight laugh)) are you frustrated.	Asks if the student is frustrated by their possible mistake.
16	Emma (s)	((turns sensor over, points to connection)). But there's and SDA right there.	Student shows Gabrielle that the green wire is connected to the other SDA port on the sensor (there are two).
17	Gabrielle	OhhK you're saying there's an SDA right there. Ok great. So then what's this one ((grabs gator:bit)) SDA. Ok you got me on that babe. You sure did. OK SCL SCL. SCL ((sets down the sensor))	Confirms that the wires are connected properly by checking the SCL connection as well.
18		OK well it looks like the wires are done right. Ok and it looks like you coding is done right. I'm gonna try something.	Tells student that the wires are connected properly.
…	…	…	…
30	Gabrielle	K so plug that in try and download it again. Cause your code looks correct babe.	Later, she directs the student to redownload the code onto the DaSH because everything looks right.

Figure 4
Gabrielle's environmentally coupled gestures with gator:bit (line 9) and wiring diagram (line 12)

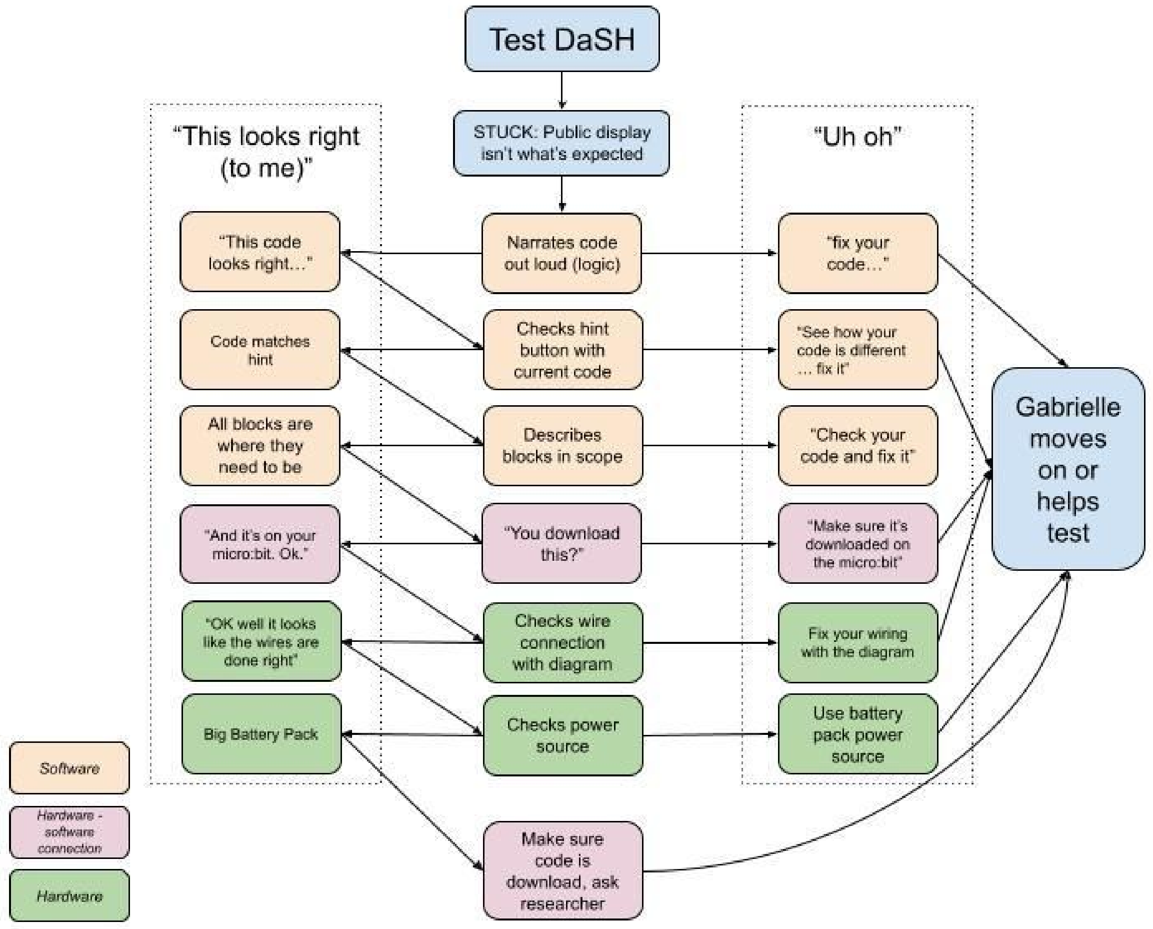


In this case study, we focus on how Gabrielle's beliefs about teaching and learning play out in her practices supporting students learning physical computing. This episode portrays the systematic and in-the-moment practices, or the interactional grammar, which stems from Gabrielle's debugging pedagogy. Consistently using highlighting embodied moves such as exaggerated gestures to involve a material resource (e.g., the wiring diagram, parts of the MakeCode tutorials), reading out the program out loud, or narrating her check of the hardware, Gabrielle did interactional work to get students to notice the bug while illustrating the process they could take to find future bugs. Such an interactional grammar struck a balance between telling students exactly what to do and letting them find the issues on their own further indicating a pedagogy where students learn through developing relationships to the materials –and to computing – by engaging in noticing practices for finding and fixing issues.

Gabrielle's persistent approach. Gabrielle's systematic approach to supporting students as they struggled to work with their DaSH is represented as a flowchart in Figure 5. Episode 1 depicts one path through this approach where Gabrielle assessed the code as correct (line 2), confirmed it was downloaded corrected onto the micro:bit (line 3), narrated a check of the wires (line 6), indicated there could be an issue with wiring (line 7-13), and later tells Emma to check the power because Emma showed the wiring was correct (line 30). Gabrielle approached her interactions with students in similar ways, progressing through this same order: narrating and asking about the software, then the software-hardware connections, then the hardware. Therefore, Figure 5 is a mapping of her interactional grammar across her second implementation with students.

Figure 5

Flowchart representing the interactional grammar of Gabrielle's debugging pedagogical practice


Test DaSH
STUCK: Public display isn't what's expected
"This looks right (to me)"
"Uh oh"
"This code looks right…"
Narrates code out loud (logic)
"fix your code…"
Code matches hint
Checks hint button with current code
"See how your code is different … fix it"
All blocks are where they need to be
Describes blocks in scope
"Check your code and fix it"
"And it's on your micro:bit. Ok."
"You download this?"
"Make sure it's downloaded on the micro:bit"
"OK well it looks like the wires are done right"
Checks wire connection with diagram
Fix your wiring with the diagram
Big Battery Pack
Checks power source
Use battery pack power source
Make sure code is download, ask researcher
Gabrielle moves on or helps test
Software
Hardware - software connection
Hardware

The "Test DaSH" box in Figure 5 was a repeated routine we observed when Gabrielle arrived at a student's work area. This involved testing the boundaries of the sensor by changing its environment – by yelling into the microphone sensor or breathing onto the CO2 sensor, for instance – and seeing how the DaSH system reacted. This initiation step often oriented Gabrielle and the students towards a shared understanding of how the DaSH was performing (e.g., LED array on the DaSH shows an X or an error code) in contrast to the student's expectation (e.g. expecting the LEDs array to show a number). Gabrielle then regularly started a series of checks represented by the steps in the middle column of Figure 5. At each step, Gabrielle often narrated what she was looking at, like she did in episode 1 (e.g., "So then we looked at our wires"). Her next move would often differ depending on whether she noticed an issue, or not. If it looked correct to her ("Ok the code looks ri:::ght" ; Table 1, line 2), she moved to the next check, represented by the arrow from the middle column to the left column back to the middle column and moving downward. If something seemed wrong, she indicated there was an issue with an utterance like "uh oh" (e.g., episode 1, lines 7 and 12) and then made interactional moves (gestures and utterances) to get the student to locate the bug and help them make a plan to fix it (e.g., Table 4 line 12-13 and Figure 4). She used the verbal indicator "uh oh" in five out of the six episodes we analyzed to indicate she had found a possible error.

Once the student identified the bug, Gabrielle sometimes stayed to support them in fixing the bug and retesting the DaSH. At other times she moved on to support other students. The different moves to stay or leave is represented in the box furthest to the right in Figure 5. If no bug was identified (as in the focal episode shown in Table 4), Gabrielle would often offer a possible next step for the student to investigate before stepping away. Regularly, this extra step (the bottom-most box in Figure 5) was for the student to download the program onto the micro:bit again. If this did not work, Gabrielle would sometimes reach out to the researcher for help if they were in the room, or, if they were not, she would tell her students she was going to bring it up with a researcher at a later time.

Even though Gabrielle did not address each of the checks in the middle column explicitly every time she worked with students, the order was consistent. If the student still had their code up on the computer then she went to that first. If the code was not available because the student already deleted it, for instance, she checked if they correctly downloaded it to the hardware. Her series of questions and checks depicts a particular pedagogical approach to explicitly model debugging with the student in a clear, and repeated, pattern that they could use later on their own. She deployed this pattern in context-dependent ways. Therefore, we identified it as a local grammar of her debugging interactions.

Affective responses to students: caring for frustrations and joys

Gabrielle's work debugging with students also included affective moves that foregrounded her care for students' frustrations and joys. Her style of showing care, using affective language, as she supported students getting unstuck was different from other participating teachers because she often explicitly asked students how they were feeling while debugging. As other scholars have contended, learning to debug is an emotional process that can be improved through supporting youth in naming their emotional responses (Dahn and DeLiema, 2020; DeLiema *et al*., 2019; Morales-Navarro *et al*., 2021).

Gabrielle believed it was important to extend the length of the SIU for students to move through challenges and find success in programming (interview, October 2, 2019). Because she wanted her students to understand struggling was a natural part of the process and experience success, she approached students with great care. Caring moves were in-the-moment utterances and/or gestures that celebrated and supported students as they engaged in debugging practice, validating their frustrations when they struggled to get unstuck and encouraging excitement when the system worked.

Episode 1. Once Gabrielle helped Emma notice what she thought to be the bug (a wiring issue, Table 1 lines 12-14), Gabrielle set down the sensor and asked the student to try wiring it again (using the wiring diagram). She followed up with utterances: "are you OK" (line 13) and "are you frustrated" (line 15). Asking if Emma was feeling frustrated made Gabrielle's care for the student's emotions and her in-the-moment read of the student's disposition explicit. In this interaction, she also came to recognize her students' growing frustration: Emma believed that the wiring was correct and didn't understand why she would need to rewire. Gabrielle assuaged this sensed frustration when she agreed with Emma's assessment that the wiring was correct, admitting that she had been mistaken saying "Ok you got me on that babe. You sure did" (line 17). Gabrielle's utterances (lines 13 and 15) are an example of how she supported her students through locating potential errors in the original construction of the DaSH. However, because Gabrielle was mistaken in her initial assessment of the bug, she shifted her affective moves in supporting the students' successful wiring, re-affirming the correct writing by narrating her thinking aloud (line 17).

Episode 2. In a second episode, Gabrielle had an opportunity to celebrate debugging success with two students she worked with just after Emma. Sarah and Jessica were working at the same table supporting each other as they tinkered with separate DaSH setups. Gabrielle had previously worked with Jessica to find a wiring bug when she had deleted her program off her computer. Gabrielle asked her to program it again and call her over. When Jessica called Gabrielle back over, Jessica had reprogrammed her system with Sarah's help. The two students told Gabrielle when she first approached that their programs and wiring were the same but Sarah's setup was working and Jessica's was not.

Gabrielle worked with both of them to debug Jessica's DaSH. Together, they moved through the checking routine as shown in Figure 5 with the assumption that the program was correct because it was the same as Sarah's which was working. They then found two possible bugs and fixed them. The bugs were: 1) the number in the logic block was too high for the current context and 2) the sound switch on the gator:bit was turned off. Jessica and Gabrielle then tested the DaSH to see if they would get an X, as they expected, when the microphone sensor picked up loud sounds. As they tested it, the "X" came up and Gabrielle gasped loudly and excitedly, exclaiming "Finally! take a look." Here, her excitement drove Jessica and Sarah to take more notice of it working, and elicited joy in their achievement

Gabrielle regularly supported students' frustrations, as in episode 1, and also exuded joy in the collective accomplishment of successfully debugging the DaSH, like in episode 2. Her attunement to students' emotional and affective responses to debugging were important, especially because, as she expressed in her reflective interview, this class particularly had a hard time learning to deal with the struggles of programming. Amplifying joy when it worked, and recognizing frustration, worked together to support students in dealing with the emotional struggles of trying to get the physical computing system working.

Positioning as a learner

When facilitating debugging practices Gabrielle often made interactional moves that decentered her expertise by positioning herself as a learner along with her students. For example, her consistent use of inclusive language with students (e.g., "we" [Table 4; line 6]; and "our" in "K so we gotta look at our code" when working with Jessica) and moves to test the system with students, like in Episode 2, all positioned her as a collaborator who didn't have the answers but was learning and problem solving with students. Further, she also regularly mentioned to the whole class as well as students in small groups that she struggled with the programming side of this unit and had to get help herself regularly. When introducing the class to using MakeCode tutorials in her second implementation, she shared "I love them [color coding.]. Ms. Alex (author 2) designed this, and I feel like sometimes she designed it for people like me who are just clueless – which is me with coding. At the beginning I had no idea how to code. Now I can teach people how to code, which is cool."

Gabrielle often referred to the researcher/second author as someone with a greater wealth of expertise and experience. In episode 2, she talked about the skills of the researcher as they struggled to fix the bugs. Unsure about how to continue, she said "well you got me there you guys, this is one for Ms. Alex (author 2)." This kind of reference was consistent across both implementations of the unit when Alex was not in the room. Referring to Alex supported Gabrielle's positioning of being a learner working with students. Because Alex was her teacher. Gabrielle referenced this positioning in an interview Alex conducted after the second implementation. She shared that the tutorials helped her get a grasp of what the students needed to do for each programming task, in the moment, which empowered her when Alex was not in the room. Further she shared that "my experience with coding is not the best. Yeah, but I'm pretty good at supporting students" (Interview, January 2020).

Regularly positioning herself as a learner in relation to programming and physical computing during the Sensor Immersion Unit was a different approach than many other participating teachers. Yet, it did fit with Gabrielle's debugging pedagogy and her goal to push students to get better at struggling through it.

Discussion: Debugging with students as pedagogy

Gabrielle's practices supporting her students in developing debugging skills exemplifies how teachers can model key moves in the debugging process while using important resources to support them in noticing possible bugs and their fixes. The multimodal character of her checking routine, her affective supportive moves, and the way she positioned herself as a learner all demonstrate how Gabrielle approached this process of ***debugging with*** students, rather than debugging for them by expecting them to work it out on their own, or directly instructing them on what to do. Gabrielle's case shows how teachers new to computing can have pedagogical strengths conducive to supporting students in the debugging process. This also illustrates the potential in developing debugging pedagogies derived from the practice of teachers like Gabrielle.

When students needed support in debugging, noticing bugs was a multimodal interactional achievement (Goodwin, 2018; Keifert and Marin, 2018) between Gabrielle and her students. Learning to debug was a process of learning to notice, together, where Gabrielle pushed students to practice more authentic debugging processes. Students took more initiative about solving the problem on their own and Gabrielle provided scaffolds to keep the challenge level appropriate and ensure productive struggle rather than frustration. This helped students develop debugging skills and practices that are used in more complex computational situations (Flood *et al*., 2018).

Gabrielle's think-aloud process, in coordination with gesture, articulated her approach to debugging for the students, a key component of her pedagogical practice. Her approach consisted of making sense of what the bug is, then providing students opportunity to notice and correct bugs, and finally collaborating on a solution. Thus for Gabrielle, simply identifying stuck points as a teachable taxonomy for middle school students is not enough to support their debugging skills. Gabrielle's teaching practices and artful use of the DaSH do what Pea, Soloway, and Sphoror (1987) argued for: "going beyond… [the] primary emphasis on teaching syntax and semantics of programming languages to the goal-directed and planful aims of programming action" (p.25) to better help students develop programming expertise. Debugging cannot be taught as an isolated skill but must be developed over time through engagement with authentic activities where students encounter bugs as they engage in computing activities. This is especially true for physical computing systems like the DaSH because of the vast possibilities for issues. Gabrielle's interactions exemplify a focus on supporting students in developing a process for negotiating stuck points and finding fixes in software, hardware, and hardware-software connections.

As Rich and colleagues' (2019) wide survey of the field of debugging research in K-8 classrooms implicitly points out: the field has been focused on developing conceptualizations of how learners develop debugging skills, and has much less clarity on the role of teachers. The learning trajectory Rich and colleagues (2019) outline represents skills students gain as they progress from novice to more experienced programmer. The case study of Gabrielle offers a teacher-focused complement that depicts what it can look like to center the process of debugging – as a systematic, flexible, and emotional process – instead of on the product of a working system.

Gabrielle often worked through finding a bug with the students, in real time, instead of first trying to find it before helping her students. This is part of her larger epistemological practice of debugging with students. In contrast to prior work that focuses on the need for deep computer science knowledge to be able to teach debugging (McCauley *et al*., 2008), Gabrielle's lack of experience became a strength because she intentionally crafted a space where she was learning with students as she encouraged them to work through getting stuck.

Gabrielle exhibited in-the-moment affective moves that supported students' emotional responses (frustrations and joys) and pushed them to work through what they might see as mistakes or failures inevitable in building their own physical computing system. Her flexible navigation between supporting students in finding the sources of the issues and their emotional processes is an implicit example of what Dahn and colleagues (2020) articulate as "strategies that promote and sustain student-driven communication about the critical thinking and emotional processes that surround debugging" (p. 211). It is also genuine of her practice, part of her general caring classroom culture, and time efficient within a limited-time setting. She furthered the caring culture by allowing students to see her explicitly as a learner in programming and physical computing.

Her overall way of working with students models for them the power of being open to learning from getting unstuck in social, emotional, and conceptual ways. Students in her class learned to debug through engagement with their own work with the DaSH because they were supported. Gabrielle rejects the position of teacher as knowledge disseminator and, instead, embraces a position of facilitator, collaborator, and fellow learner. We thus characterize this pedagogical approach as ***debugging with her students*** to support their development of skills and strategies. This approach relied on her experience as an elementary and middle school science teacher, drawing from her expertise at supporting students relationally. To find success supporting students, she leaned on her strengths as an educator even though she had no programming or physical computing experience prior to joining our project.

Conclusions

Gabrielle's combination of being vulnerable as a learner, showing affective support, and modeling a systematic, multimodal, routine for finding bugs has important implications because programming with a supportive peer-community has the potential to promote social, emotional and identity development (Roque and Rusk, 2019). Gabrielle modeled supportive peer practices while helping them develop their own technical skills. This focus on social and relational practices of debugging has the potential to equitably impact students' identity formation in relation to programming (Huadong, 2019) and physical computing. Further, it is part of developing an equitable and inclusive learning environment where "every student has access to not only material resources (e.g., rich content and quality instruction) but to non-material resources (e.g., identities as computer scientists and peer relationships)," (Shaw, *et al*, 2020, p. 315) and teachers and students are positioned in more collaborative expertise-sharing ways.

The skills of debugging are, themselves, integral to computer programming practice (DeLiema *et al*., 2019) and thus a major component in developing computational thinking (Grover and Pea, 2012). Educators working with physical computing systems in a variety of learning environments require deft facilitation and orchestration skills that incubate learners' development of their own process for debugging while also helping them get their current DaSH working, often in a short amount of time. Learning from this case, future professional development of computer science educators – both in formal context like classrooms and less formal contexts like afterschool programs and libraries – should emphasize the multimodal and affective skills required in supporting students to debug, or what we call Gabrielle's debugging pedagogy. Thus, the design of professional development programs should foreground these skills as a complement to teachers developing technical expertise of the computing systems their students will work with.

In this way, Gabrielle's case illuminates the need for future research to understand how to support educators in developing facilitation skills as part of their own debugging pedagogy and how it plays out in the fast-paced moments of supporting learners as they tinker with their systems. Particularly, more research is needed on how educators new to physical computing rely on their own expertise and pedagogies to equitably support students to develop debugging skills and practices in the face of struggles and successes. Such generally underscores the importance of learning from educators practice by engaging, and finding expertise from, educators of various skills and confidence in computer science instruction.

Acknowledgments

We deeply thank Gabrielle, and her students for giving us the gift of thinking with their stories. We are also in debt to our ongoing partners at Gabrielle's district for continuing the Research Practice Partnership with SchoolWide Labs that made this possible. Further, thank you to the anonymous reviewers for their valuable feedback. This research was supported by the National Science Foundation (Award No. 1742053, No. 2019805 and No. 1742046) and the James S. McDonnell Foundation.

References

Ahmadzadeh, M., Elliman, D., and Higgins, C. (2005), "An analysis of patterns of debugging among novice computer science students", In *Proceedings of the 10th annual SIGCSE conference on Innovation and* *technology in computer science education*, pp.84-88.

Anastopoulou, S., Sharples, M., Ainsworth, S., Crook, C., O'Malley, C., and Wright, M. (2012), "Creating personal meaning through technology-supported science inquiry learning across formal and informal settings", *International Journal of Science Education*, Vol. 34 No. 2, pp.251–273.

Biddy, Q., Chakarov, A. G., Bush, J., Elliott, C. H., Jacobs, J., Recker, M., Sumner, T., & Penuel, W. (2021). A Professional Development Model to Integrate Computational Thinking Into Middle School Science Through Codesigned Storylines. Contemporary Issues in Technology and Teacher Education, 21(1), 53–96.

Blikstein, P. (2013), "Gears of our childhood: Constructionist toolkits, robotics, and physical computing, past and future", *Proceedings of the 12th International Conference on Interaction Design and Children*, pp.173–182.

Blikstein, P. and Moghadam, S. (2019), "Computing education: Literature review and voices from the field", S. Fincher and A. Robins (Eds.), *The Cambridge Handbook of Computing Education Research,* Cambridge University Press, Cambridge, pp.56-78.

Brennan, K. and Resnick, M. (2012), "New frameworks for studying and assessing the development of computational thinking", 25.

Buechley, L., Eisenberg, M., Catchen, J., and Crockett, A. (2008). The LilyPad Arduino: Using Computational Textiles to Investigate Engagement, Aesthetics, and Diversity in Computer Science Education. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 423–432. https://doi.org/10.1145/1357054.1357123

Buxton, C. A. (2010), "Social problem solving through science: An approach to critical, place-based, science teaching and learning", *Equity and Excellence in Education*, Vol. 43 No. 1, pp.120–135.

Coburn, C. E., Penuel, W. R., and Geil, K. E. (2013), "Research-practice partnerships: A strategy for leveraging research for educational improvement in school districts", William T Grant Foundation.

Cole, M. (1996). *Cultural psychology: A once and future discipline* (6. printing). Belknap Press of Harvard Univ. Press.

Dahn, M. and DeLiema, D. (2020), "Dynamics of emotion, problem solving, and identity: Portraits of three girl coders", *Computer Science Education*, Vol. 30 No. 3, pp.362–389.

Dahn, M., DeLiema, D., and Enyedy, N. (2020), "Art as a point of departure for understanding student experience in learning to code", *Teachers College Record*, Vol. 122 No. 8, pp.1–42.

DeLiema, D., Dahn, M., Flood, V. J., Asuncion, A., Abrahamson, D., Enyedy, N., and Steen, F. (2019), "Debugging as a context for fostering reflection on critical thinking and emotion", E. Manalo (Ed.), *Deeper Learning, Dialogic Learning, and Critical Thinking,* (1st ed.). Routledge, pp.209-228.

DesPortes, K. and DiSalvo, B. (2019), "Trials and tribulations of novices working with the Arduino", *Proceedings of the 2019 ACM Conference on International Computing Education Research*, pp.219–227.

Devine, J., Finney, J., de Halleux, P., Moskal, M., Ball, T. and Hodges, S. (2018), "MakeCode and CODAL: Intuitive and efficient embedded systems programming for education", *Proceedings of the 19th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems*, pp.19–30.

Esmonde, I. (2016). Power and Sociocultural Theories of Learning. In I. Esmonde & A. Booker (Eds.), *Power and Privilege in the Learning Sciences: Critical and Sociocultural Theories of Learning* (pp. 6–27). Taylor & Francis.

Fields, D. A., Searle, K. A., and Kafai, Y. B. (2016), "Deconstruction kits for learning: Students' collaborative debugging of electronic textile designs", *Proceedings of the 6th Annual Conference on Creativity and Fabrication in Education*, pp.82–85.

Flood, V. J., DeLiema, D., Harrer, B. W., and Abrahamson, D. (2018), "Enskilment in the digital age: The interactional work of learning to debug", J. Kay and R. Luckin (Eds.), *Rethinking Learning in the Digital Age: Making the Learning Sciences Count*, 13th International Conference of the Learning Sciences (ICLS) 2018, Volume 3, London, UK: International Society of the Learning Sciences, pp.1405-1406.

Fitzgerald, S., Lewandowski, G., McCauley, R., Murphy, L., Simon, B., Thomas, L., and Zander, C. (2008), "Debugging: finding, fixing and flailing, a multi-institutional study of novice debuggers", *Computer Science Education*, Vol. 18 No.2, pp.93-116.

Gendreau Chakarov, A., Biddy, Q., Hennessy Elliott, C., & Recker, M. (2021). The Data Sensor Hub (DaSH): A Physical Computing System to Support Middle School Inquiry Science Instruction. *Sensors*, *21*(18), 6243. https://doi.org/10.3390/s21186243

Gendreau Chakarov, A., Biddy, Q., Jacobs, J., Recker, M., & Sumner, T. (2020). Opening the Black Box: Investigating Student Understanding of Data Displays Using Programmable Sensor Technology. *Proceedings of the 2020 ACM Conference on International Computing Education Research*, 291–301. https://doi.org/10.1145/3372782.3406268

Goodwin, C. (2018), *Co-operative action*, Cambridge University Press, Cambridge, UK.

Grover, S. and Pea, R. (2013), "Computational thinking in K–12: A Review of the state of the field", *Educational Researcher*, Vol. 42 No. 1, pp.38–43.

Haduong, P. (2019). "I like computers. I hate coding": A portrait of two teens' experiences. Information and Learning Sciences, 120(5/6), 349–365. https://doi.org/10.1108/ILS-05-2018-0037

Haduong, P. and Brennan, K. (2019), "Helping K–12 teachers get unstuck with Scratch: The design of an online professional learning experience", *Professional Development*, Vol. 7.

Hall, R. and Stevens, R. (2016). "Developing approaches to interaction analysis of knowledge in use", A. A. DiSessa, M. Levin, and N. J. S. Brown (Eds.), *Knowledge and interaction: A synthetic agenda for the learning sciences*, Routledge.

Hardy, L., Dixon, C. and Hsi, S. (2020), "From data collectors to data producers: Shifting students' relationship to data", *Journal of the Learning Sciences*, Vol. 29 No. 1, pp.104–126.

Jayathirtha, G., Fields, D., and Kafai, Y. B. (2020). Pair Debugging of Electronic Textiles Projects: Analyzing Think-Aloud Protocols for High School Students' Strategies and Practices While Problem Solving. Gresalfi, M. and Horn, I. S. (Eds.), The Interdisciplinarity of the Learning Sciences, 14th International Conference of the Learning Sciences (ICLS) 2020, 2, 1047–1054.

Jayathirtha, G., and Kafai, Y. B. (2020). Interactive Stitch Sampler: A Synthesis of a Decade of Research on Using Electronic Textiles to Answer the Who, Where, How, and What for K--12 Computer Science Education. ACM Transactions on Computing Education, 20(4), 1–29. https://doi.org/10.1145/3418299

Jefferson, G. (2004), "Glossary of transcript symbols with an introduction", *Pragmatics and Beyond New Series*, Vol. 125, pp.13–34.

Jordan, B. and Henderson, A. (1995), "Interaction analysis: Foundations and practice", *Journal of the Learning Sciences*, Vol. 4 No. 1, pp.39–103.

Kafai, Y. B., DeLiema, D., Fields, D. A., Lewandowski, G., and Lewis, C. (2019), "Rethinking debugging as productive failure for CS education", *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pp.169–170.

Kafai, Y. B., Lee, E., Searle, K., Fields, D., Kaplan, E., and Lui, D. (2014), "A crafts-oriented approach to computing in high school: Introducing computational concepts, practices, and perspectives with electronic textiles", *ACM Transactions on Computing Education*, Vol. 14 No. 1, pp.1–20.

Kafai, Y., Hutchins, N., Snyder, C., Brennan, K., Haduong, P., DesPortes, K., DeLiema, D., Aalst, O. W., Flood, V., Fong, M., Fields, D., Gresalfi, M., Brady, C., Steinberg, S., Franklin, D., Eatinger, D., Coenraad, M., Palmer, J., Weintrop, D., … Bulalacao, N. (2020), "Turning bugs into learning opportunities: Understanding debugging processes, perspectives, and pedagogies", Gresalfi, M. and Horn, I. S. (Eds.), The Interdisciplinarity of the Learning Sciences, 14th International Conference of the Learning Sciences (ICLS) 2020, Vol. 1, Nashville, Tennessee: International Society of the Learning Sciences, pp.374-381.

Kapur, M. (2008), "Productive failure", *Cognition and Instruction*, Vol. 26 No. 3, pp.379–424.

Kapur, M. and Bielaczyc, K. (2012), "Designing for productive failure", *Journal of the Learning Sciences*, Vol. 21 No. 1, pp.45–83.

Keifert, D.T., Marin, A.M., (2018), A Commentary on Charles Goodwin's "Co-Operative Actionn" for Learning Scientists. Cognition and Instruction 36, 171–187. https://doi.org/10.1080/07370008.2018.1460845

Kim, C., Yuan, J., Vasconcelos, L., Shin, M., and Hill, R. B. (2018), "Debugging during block-based programming", *Instructional Science*, Vol. 46 No. 5, pp.67–787.

Kinnunen, P. and Simon, B. (2010) 'Experiencing Programming Assignments in CS1: The Emotional Toll', in Proceedings of the Sixth International workshop on Computing Education Research. pp. 77–86.

Ko, A.J., Myers, B.A., (2003), Development and evaluation of a model of programming errors. Proceedings of the *IEEE Symposium on Human Centric Computing Languages and Environments*. pp. 7–14.

Ko, A. J. and Myers, B. A. (2005), "A framework and methodology for studying the causes of software errors in programming systems", *Journal of Visual Languages and Computing*, Vol. 16 No. 1–2, pp. 41–84.

Lewis, C.M. (2012), The importance of students' attention to program state: a case study of debugging behavior. In *Proceedings of the eighth International workshop on Computing Education Research* (pp. 127-134).

McCauley, R., Fitzgerald, S., Lewandowski, G., Murphy, L., Simon, B., Thomas, L., and Zander, C. (2008), "Debugging: A review of the literature from an educational perspective", *Computer Science Education*, Vol. 18 No. 2, pp.67–92.

Miyake, N. (1986). Constructive Interaction and the Iterative Process of Understanding. Cognitive Science, 10(2), 151–177. https://doi.org/10.1207/s15516709cog1002_2

Morales-Navarro, L., Fields, D. A., and Kafai, Y. B. (2021). Growing Mindsets: Debugging by Design to Promote Students' Growth Mindset Practices in Computer Science Class. In Proceedings of the 15th International Conference of the Learning Sciences-ICLS 2021. International Society of the Learning Sciences

Meyers, E. M. (2019). Guest editorial. Information and Learning Sciences, 120(5/6), 254–265. https://doi.org/10.1108/ILS-05-2019-139

Nasir, N. S., & Hand, V. M. (2006). Exploring sociocultural perspectives on race, culture, and learning. *Review of Educational Research*, *76*(4), 449–475.

Norooz, L., Mauriello, M. L., Jorgensen, A., McNally, B., and Froehlich, J. E. (2015), "BodyVis: A new approach to body learning through wearable sensing and visualization", *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp.1025–1034.

O'Dell, D. H. (2017), "The debugging mind-set", *Communications of the ACM*, Vol. 60 No. 6, pp.40–45.

Papert, S. (1993), *Mindstorms: Children, computers, and powerful ideas* (2nd ed). Basic Books.

Pea, R. (1986), "Language-independent conceptual 'bugs' in novice programming", *Journal of Educational Computing Research*, Vol. 2 No. 1, pp.25–36.

Pea, R. D., Soloway, E., and Spohrer, J. C. (1987), "The buggy path to the development of programming expertise", *Focus on Learning Problems in Mathematics,* Vol. 9 No. 1, pp.5–30.

Penuel, W. R., Riedy, R., Barber, M. S., Peurach, D. J., LeBouef, W. A., and Clark, T. (2020), "Principles of collaborative education research with stakeholders: Toward requirements for a new research and development infrastructure", *Review of Educational Research*, Vol. 90 No. 5, pp.627–674.

Peppler, K. (2013), "STEAM-powered computing education: Using e-textiles to integrate the arts and STEM", *Computer*, Vol. 46 No. 9, pp.38–43.

Rich, K. M., Strickland, C., Binkowski, T. A., and Franklin, D. (2019), "A K-8 debugging learning trajectory derived from research literature", *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pp.745–751.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. and Kafai, Y. (2009), "Scratch: programming for all", *Communications of the ACM*, Vol. 52 No. 11, pp.60-67.

Roque, R., and Rusk, N. (2019). Youth perspectives on their development in a coding community. Information and Learning Sciences, 120(5/6), 327–348. https://doi.org/10.1108/ILS-05-2018-0038

Sentance, S., Waite, J., Yeomans, L., and MacLeod, E. (2017), "Teaching with physical computing devices: The BBC micro:bit initiative", *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*, pp.87–96.

Shaw, M. S., Fields, D. A., and Kafai, Y. B. (2020). Leveraging local resources and contexts for inclusive computer science classrooms: Reflections from experienced high school teachers implementing electronic textiles. Computer Science Education, 30(3), 313–336. https://doi.org/10.1080/08993408.2020.1805283

Solomon, A., Bae, M., DiSalvo, B., and Guzdial, M. (2020). Embodied Representations in Computing Education: How Gesture, Embodied Language, and Tool Use Support Teaching Recursion. In Gresalfi, M. and Horn, I. S. (Eds.), The Interdisciplinarity of the Learning Sciences, 14th International Conference of the Learning Sciences (ICLS) 2020, Volume 4 (pp. 2133-2140). Nashville, Tennessee: International Society of the Learning Sciences.

Stake, R.E., (1995), The art of case study research. Sage Publications, Thousand Oaks.

Uppal, G. (2020), "Exploring and understanding pupils' lack of perseverance and autonomy with debugging in computing", *Research in Teacher Education*, Vol. 10 No. 1, pp.12-16

Vessey, I. (1985). Expertise in debugging computer programs: A process analysis. International Journal of Man-Machine Studies, 23(5), 459–494. https://doi.org/10.1016/S0020-7373(85)80054-7

Vygotsky, L. (1978). Mind in Society: The Development of Higher Psychological Processes (Revised ed. edition). Harvard University Press.

Warner, J. R., Fletcher, C. L., Torbey, R., and Garbrecht, L. S. (2019, February). Increasing capacity for computer science education in rural areas through a large-scale collective impact model. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education, pp.1157-1163.

Webb, D. C. (2010), "Troubleshooting assessment: An authentic problem solving activity for IT education", *Procedia: Social and Behavioral Sciences*, Vol. 9, pp.903–907.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., and Wilensky, U. (2016), "Defining computational thinking for mathematics and science classrooms", *Journal of Science Education and Technology*, Vol. 25 No. 1, pp.127–147.

Witherspoon, E. B., and Schunn, C. D. (2019). Teachers' goals predict computational thinking gains in robotics. Information and Learning Sciences, 120(5/6), 308–326. https://doi.org/10.1108/ILS-05-2018-0035

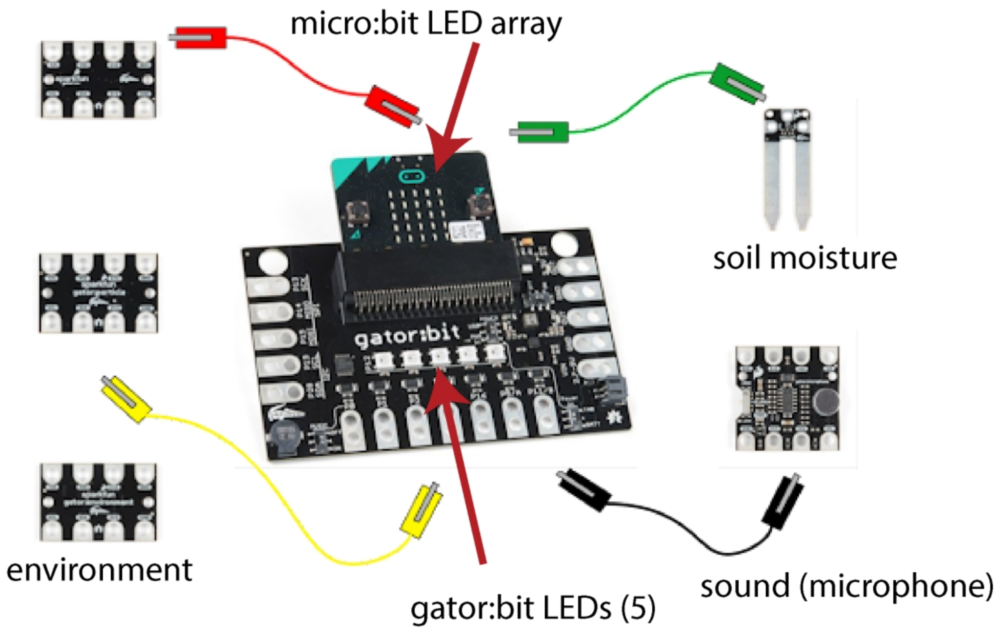


Figure 1. The DaSH is composed of a micro:bit (top center), gator:bit (middle), and a set of alligator clippable sensors that can measure different environmental conditions such as temperature, noise level, or UV level. The micro:bit and gator:bit support the display of information through onboard LEDs and a speaker.

193x135mm (300 x 300 DPI)

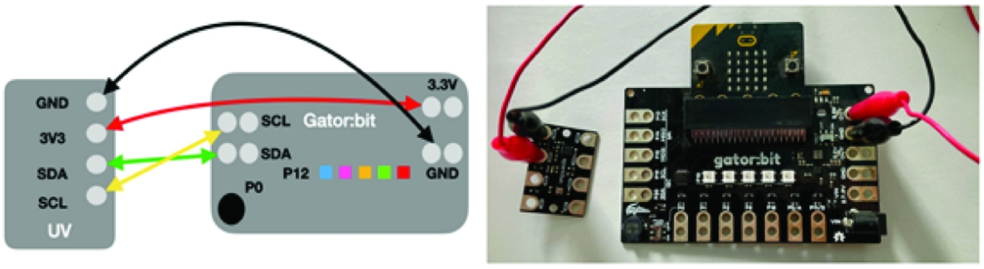


Figure 2. The image on the left shows the wiring diagram used by students to wire the UV sensor. The image on the right shows a common issue that students encounter where they have only wired the power wire (red) and ground wire (black)

378x100mm (300 x 300 DPI)

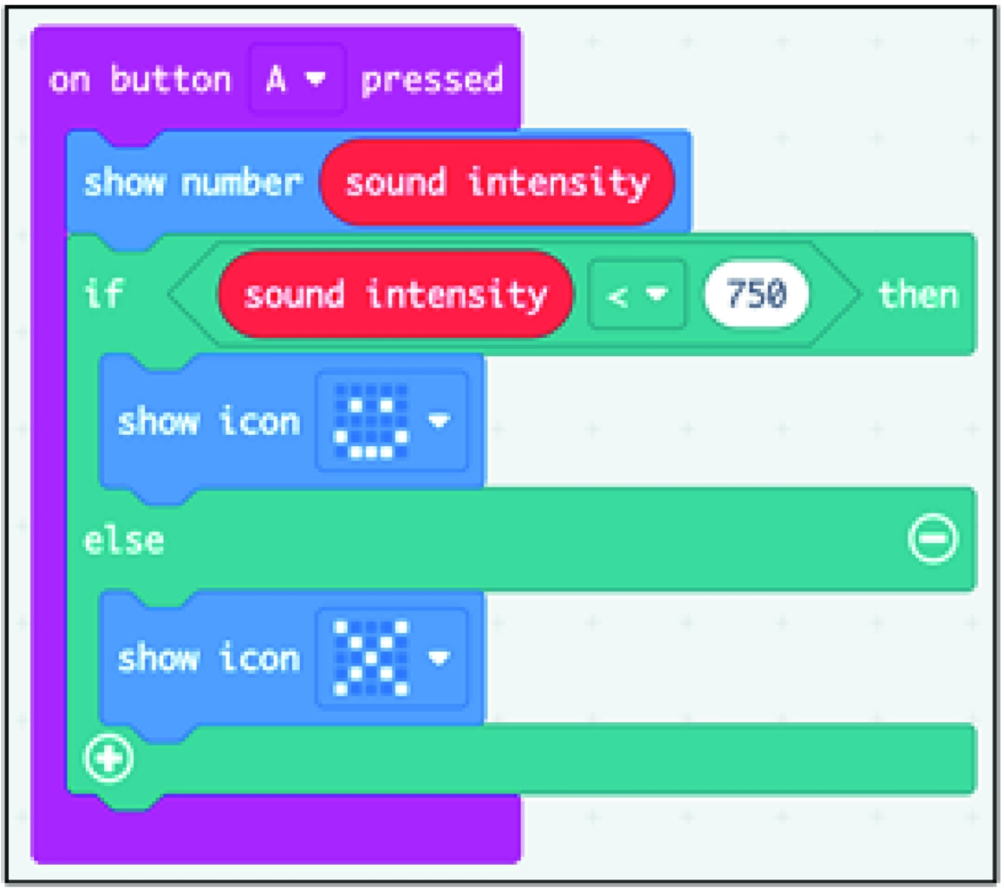

```
on button A ▾ pressed
    show number sound intensity
    if sound intensity < ▾ 750 then
        show icon ▾
    else
        show icon ▾
```

Figure 3. This code snippet represents a stuck point for students where the value displayed from the show number block is not the same value as used by the conditional logic statement.

132x117mm (300 x 300 DPI)

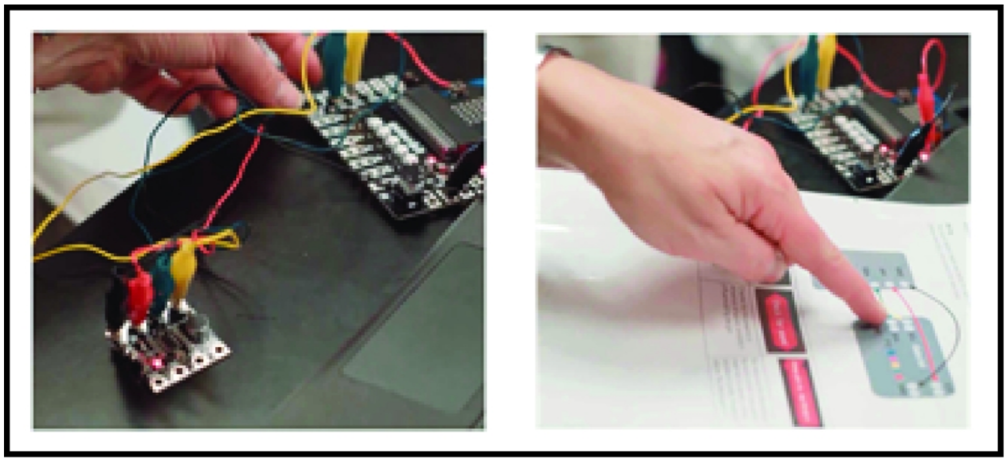


Figure 4. Gabrielle’s environmentally coupled gestures with gator:bit (line 9) and wiring diagram (line 12)

190x86mm (300 x 300 DPI)

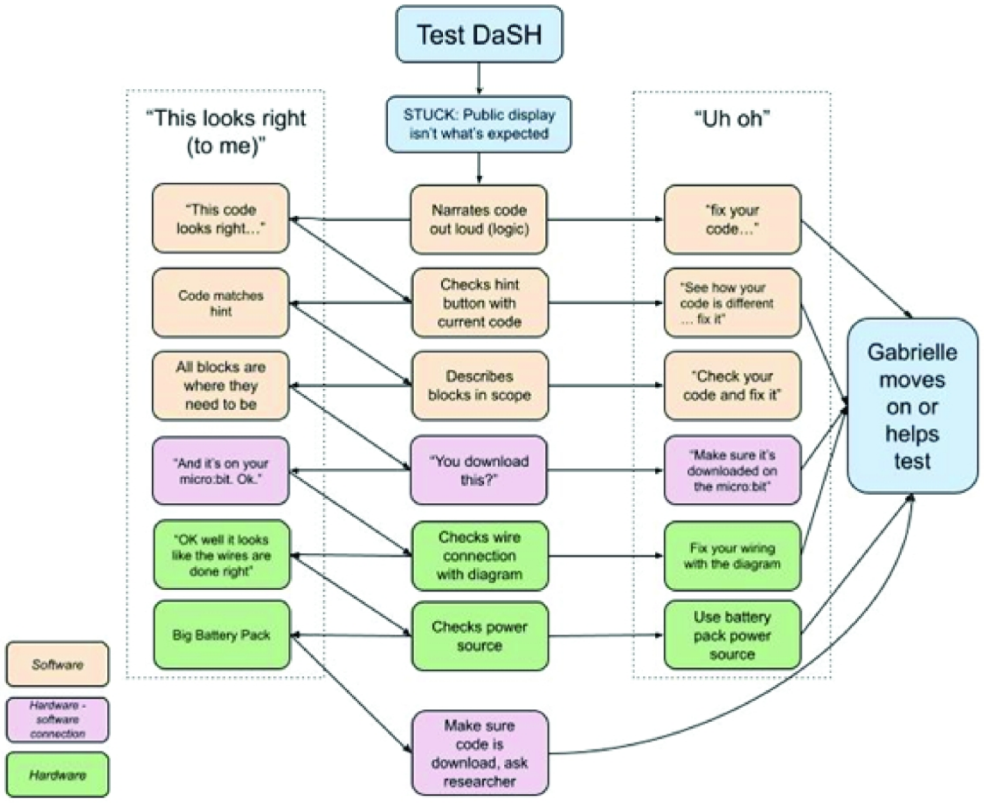

Test DaSH
STUCK: Public display isn't what's expected
"This looks right (to me)"
"Uh oh"
"This code looks right…"
Code matches hint
All blocks are where they need to be
"And it's on your micro:bit. Ok."
"OK well it looks like the wires are done right"
Big Battery Pack
Narrates code out loud (logic)
Checks hint button with current code
Describes blocks in scope
"You download this?"
Checks wire connection with diagram
Checks power source
Make sure code is download, ask researcher
"fix your code…"
"See how your code is different … fix it"
"Check your code and fix it"
"Make sure it's downloaded on the micro:bit"
Fix your wiring with the diagram
Use battery pack power source
Gabrielle moves on or helps test
Software
Hardware - software connection
Hardware

Figure 5. Flowchart representing the interactional grammar of Gabrielle's debugging pedagogical practice

209x173mm (300 x 300 DPI)