Engineering Educators Bringing the World Together
2025 ASEE Annual Conference & Exposition
Palais des congrès de Montréal, Montréal, QC • June 22–25, 2025 ASEE
Paper ID #46896

Embodied Sensors and Digital Twins as an Introduction to Microprocessor Programming for Middle and High School Non-CS Majors

Ms. Leslie Bondaryk, The Concord Consortium

Leslie Bondaryk received the B.S. degree from the Massachusetts Institute of Technology, and the M.S. degree from the University of California, Santa Barbara, both in electrical engineering. She is currently the Chief Technology Officer with the Concord Consortium, Concord, MA, USA. Over her career, Ms. Bondaryk has introduced new technologies to educational research and publishing projects across computer science, mathematics, engineering, and sciences, including the first Web Calculus text, The Analytical Engine Online (PWS Publishing, 1998), and Schaum's Interactive Outline Series (McGraw Hill, 1994–2000). She was a defining force behind Mathcad software and the educational version Studyworks. She is the author of papers, articles, and book chapters on technology adoption in traditional classrooms, citizen science, and more recently on collaborative and AI supportive technologies in STEM software. Her research interests include data visualization, collaborative learning technologies, and novel STEM educational interfaces for formative learning and assessments.

Aaron Kyle, Duke University

Aaron Kyle is a Professor of the Practice in the Department of Biomedical Engineering at Duke University. Kyle teaches first-year design, medical device instrumentation, and capstone design courses at Duke. He leads the Outreach Design Education (ODE) Program, an NIH-funded set of activities focused on enhancing STEM identity in younger students through engineering design education. Over 1000 students in New York and North Carolina have been introduced to engineering design through ODE-associated pre-college programs and courses.

Kyle received his B.S. in Electrical Engineering from Kettering University ('02) and his Ph.D. in Biomedical Engineering from Purdue University ('07). He is a fellow of the AIMBE and BMES and serves on the BMES Board of Directors.

Ido Davidesco, University of Connecticut
Chad Dorsey
Bianca Montrosse-Moorhead, University of Connecticut

Bianca Montrosse-Moorhead, Ph.D., is a Professor of Research Methods, Measurement, and Evaluation at the University of Connecticut, where she also directs the Partnership for Evaluation and Educational Research (PEER) lab. As Co-Editor-in-Chief of New Directions for Evaluation and internationally recognized evaluation scholar, Bianca has dedicated her career to bridging the space between evaluation theory, research, and practice. Her scholarship encompasses a broad spectrum of contributions, from evaluating various educational and social programs using diverse methodologies to enhancing the professional training of evaluators worldwide.

Embodied Sensors and Digital Twins as an Introduction to Microprocessor Programming for Middle and High School Non-CS Majors

Abstract

Low-cost, accessible microelectronics and sensors embedded in a bioengineering curriculum are ideal for generating engineering interest and computational thinking proficiency in non-engineering high school courses and middle school electives. This kind of curriculum provides relatable, empathetic, real-world engineering challenges that engage non-engineering-focused and marginalized student communities. This paper describes recent curriculum and instrumentation updates to two curriculum units: (1) a novel bioengineering high school unit that challenges students to solve robotics control problems using electromyography sensor technology, an Arduino kit, and a collaborative dataflow programming language and (2) a middle school Internet of Things engineering unit that engages students with microcontroller sensor-based problem-solving scenarios. In the high school unit, students design a robotic gripper that opens and closes based on their own muscles' electrical activity. This model of a bionic arm integrates life science content, computational thinking skills, and engineering design. Students are scaffolded through stakeholder analysis, biological electrophysiology, microprocessor instrumentation, and programming concepts required to solve control and feedback problems. In the middle school unit, students design a feedback and control system using micro:bits, sensors, fans, heat lamps, and humidifiers to maintain ideal temperature and humidity conditions in a terrarium environment. In these units, students are motivated to solve computational thinking challenges based on achieving simulated or hands-on goals. In both curricula, students have the opportunity to create and test logic with programmable simulations—digital twins—of the hardware kits to evaluate the utility of twins in separating debugging concerns of algorithms vs. hardware. Specialty programming blocks help students deal with complex feedback issues beyond the scope of non-major students, such as hardware response time and proportional-integral-derivative control. The classroom experience revealed gains in students' self-efficacy in engineering design and improvements in ability to recognize key components of feedback-control systems. Class tests also revealed challenges associated with scaffolding both students and teachers at these grade levels and levels of experience or interest in computational subjects. Students struggled with algorithmic design in particular, which made it harder for them to complete the capstone projects in the curricula. There were also lessons learned about robust design and instrumentation of physical devices in classes that might only use them for a short period of time, posing hurdles for both students and teachers. Software affordances developed for programming and analyzing data from the devices and the observable moments of computational struggle are discussed.

Keywords

Flow Programming, Computational Thinking, Microprocessors, Digital Twin, Simulation

Background

In modern science, computational thinking (CT) methods have increasingly become an indispensable part of scientific inquiry [1]. To support students in modeling professional science practice in the classroom, researchers have operationalized CT for K-12 students [2], [3]. Integrating science with CT can deepen learning in both domains [4] and integrated STEM classes tend to produce better student acquisition of the Next Generation Science Standards (NGSS) crosscutting standards [5], [6]. Further, cross-curriculum introduction of engineering and computer programming is often a requirement for inclusion in middle and high school curricula, as many schools do not have dedicated time or resources for an engineering class [7], [8]. In the United States, for example, computer science and programming courses are only offered by 57% of high schools and are a graduation requirement in only 11 states. In 2024, only 6.4% of students across 41 states surveyed enrolled in a foundational computer science course [9].

Even when these classes exist, students from minoritized groups that are underrepresented in STEM are less likely to enroll, either because they do not perceive themselves as engineers [10], [11], or because they doubt the community value of engineering careers [12]. To provide better access to all students and to better prepare students to participate in engineering reasoning before and after graduation, we need more curricula that emphasize relatable, project-based engineering topics that require CT [8], [13]. Introduction of engineering and CT practices in curricula outside of engineering, particularly at the middle and high school level, requires phenomena centered on problems students are interested in solving [8]. Theoretical and empirical research suggests that personally relevant instruction will result in higher engagement and a greater likelihood of successful knowledge revision [14]. In addition, the practice of engineering when applied to complex problems often requires integrating knowledge and skills across STEM disciplines and skill sets, including empathetic design and stakeholder validation [15]. Cross-disciplinary, real-use-case curricula that integrate engineering and, in particular, CT, are critical to demonstrate the interdisciplinary nature of the engineering practice and provide CT material that is likely to be learnable and interesting to a broad cross-section of students.

Bioengineering is an interdisciplinary field where technological solutions are designed to improve life quality, promoting empathetic inquiry and design. Bioengineering requires understanding of biology, physics, chemistry, programming, a variety of engineering fields, and practical constraints such as cost and societal adoption, making it fertile ground for finding motivational CT case studies [5]. Given the recent emphasis on microelectronics as a driver for much of 21st century technology and as a powerhouse for economic change in communities [16], it makes sense to introduce microelectronics, Internet of Things (IoT) topics, and CT into bioengineering classes. Doing so demonstrates how the ready availability and low cost of devices such as the Arduino or the micro:bit can make possible a whole host of transformative engineering solutions [17]. With low-cost electronics, K-12 students can be introduced to this technology and engage in engineering design that is meaningfully integrated with the life sciences [17]. Despite the growing body of work on the characteristics of interdisciplinary STEM curricula, additional research is needed to understand successful patterns for positive learning outcomes in integrated STEM programs, specifically, those programs that merge science and engineering with embodied signals and models, and particularly for non-majors [6], [18], [19]. The studies described here include continued refinement of two integrated STEM

curricular units, one for high school [20], [21] and one for middle school [22]. The high school curriculum introduces students to bioengineering in the context of microprocessors and control programming with expanded opportunities for modeling, algorithm development, and exploration of physical systems. The middle school curriculum is an IoT engineering unit that introduces middle school students to feedback mechanisms to control environmental conditions in a terrarium, offering lessons in systems design for both biological and sensor control systems.

A challenge in creating integrated engineering curricula is that the teachers and students in these disciplines are not, by definition, well-practiced in techniques of debugging either hardware or software [18], [19], [23]. In fact, this skill is something that continues to plague undergraduate engineering students [24]. Debugging is one of the primary competencies of CT [3], [4], and yet it is particularly challenging to isolate and identify problems when a system has both hardware and software components [24]. The benefits of embodied electronic systems for learning are at odds with the general challenges of instrumenting this kind of curriculum. One important technique for system isolation while debugging is the use of a digital twin, a programmable simulation that reflects a physical system [25]. The use of digital twins for separation of concerns in error correction is commonplace in the engineering industry [25], and has growing acceptance in undergraduate education, but it is unusual in K-12 [26]. Some digital twin systems are being explored as a way to serve educational communities that do not have ready access to hardware [26]; for analysis of systems too large, long, or complex to analyze in a classroom context [27]; or for systems that are not visible by other means [28]. Few are being used for the educational purpose of easing the transition to a hardware system that has its own set of issues beyond the challenge of logical problem solving.

curriculum

high school bioengineering curriculum

The high school curriculum unit has been developed and refined over several years to emphasize (1) the ways in which understanding of biology and physiology can impact engineering interventions, (2) the importance of a holistic approach to engineering and front-end design, and (3) the computational thinking skills required to make a functional design that meets all design requirements [21], [22]. Students are scaffolded through a case study about a teenage amputee who uses bionic arms. The curriculum design team chose this anchoring phenomenon to motivate the questions and activities that arise in each successive lesson to add coherence and connection across exercises [5], and as a meaningful case of empathetic engineering [15]. This anchor is used to motivate lessons on muscle electrical signals and skin conductivity, current, electrical sensors, and mechanical control. The 4 week unit was designed in conjunction with a pilot teacher using co-design methodology [29] to ensure it could be adopted into a typical STEM high school curriculum. The storyline instructional model uses students' questions to guide collaborative sensemaking about phenomena, and the students are asked to reflect at the end of each lesson on the incremental STEM knowledge they have gained.

As the curriculum progresses, students are challenged in a progressive scaffold of CT interactions to mimic the many functions of a human hand with a combination of sensors and programming. Students in later lessons are tasked with challenges that require them to think

about what sensors and feedback conditions are required to keep a robotic hand from crushing the object it holds. These practical exercises in sensor and feedback design are related closely back to their biological counterparts. Students simultaneously gain an appreciation both for the complexity and physics of human physiology and neurology, and the ways in which a practical understanding of microprocessor and mechanical engineering can be brought to bear to solve societal problems. There is evidence that these physical and modeling modalities are beneficial for establishing the learning connections across disciplines and to solidify the mechanics of the physical systems for students [30].

The curriculum is scaffolded in a Use -> Modify -> Create progression shown to be effective for engaging youth in CT in rich computational environments [8]. To gain experience with and confidence in the system before moving on to CT challenges, students experiment with functioning software/hardware systems before they are asked to change or design programs and sensors. This approach allowed for rapid uptake of the hardware and software tools. However, it created some friction in terms of students trying to use the initial examples for subsequent challenges without detailed analysis of whether the same approach was correct. We will explore this further in the Analysis section.

Students are also briefly scaffolded at the start of the curriculum to use the Dataflow programming language [23] and digital twins to turn digital light bulbs on and off. Two capstone CT challenges are presented at the end of each module. The first capstone challenge maps a sensor output to an actuated device: a muscle sensor must be used to successfully open or close a mechanical gripper. The second challenge refines the first by asking students to avoid crushing objects with the gripper. This requires students to think about constraints an engineer would apply to a simple system and introduces concepts such as monitoring signal values and additional control sensors, thresholding, and feedback. The curriculum was expanded to include sample programs throughout the lessons to give students practice with wiring program functions together, adjusting function parameters, and updating and changing logic.

middle school systems engineering curriculum

The middle school unit contains a number of analogous features and challenges, though it is significantly shorter in duration due to its need to integrate modularly into an existing elective curriculum. This two-week unit challenges students to learn the basics of sensing and control systems, then design and program a control system to successfully maintain ideal atmospheric conditions within a terrarium. The conditions for a successful terrarium require a knowledge of the growth conditions for both plants and a gecko in the system. Students gain an appreciation for the key components of control systems, including sensor-based inputs, actuated outputs, and feedback-based control loops. They also gain experience with coding and testing such systems using a remote IoT setup within the classroom,investigating central concepts related to IoT networking, signal processing, and network configuration and topologies.

hardware, software, and programmable simulations

embodied sensor kits

Both curricula use hardware kits and a software programming language that allows students to engage directly with sensors, including the ability in the high school unit to create motion from sensors attached to their own bodies. The kits for this unit contain a set of EMG sensors, a programmable mechanical gripper, and an Arduino and shield that can be used to power the sensors and communicate signals through the system. The middle school units contain micro:bits and sensors for humidity and temperature in conjunction with heat lamps, fans, and humidifiers contained within an integrated terrarium environment. All of these components are routed through Dataflow, a custom data pipeline programming language that allows students to easily connect with hardware components and introduce filtering and control logic as shown in Figure 1.

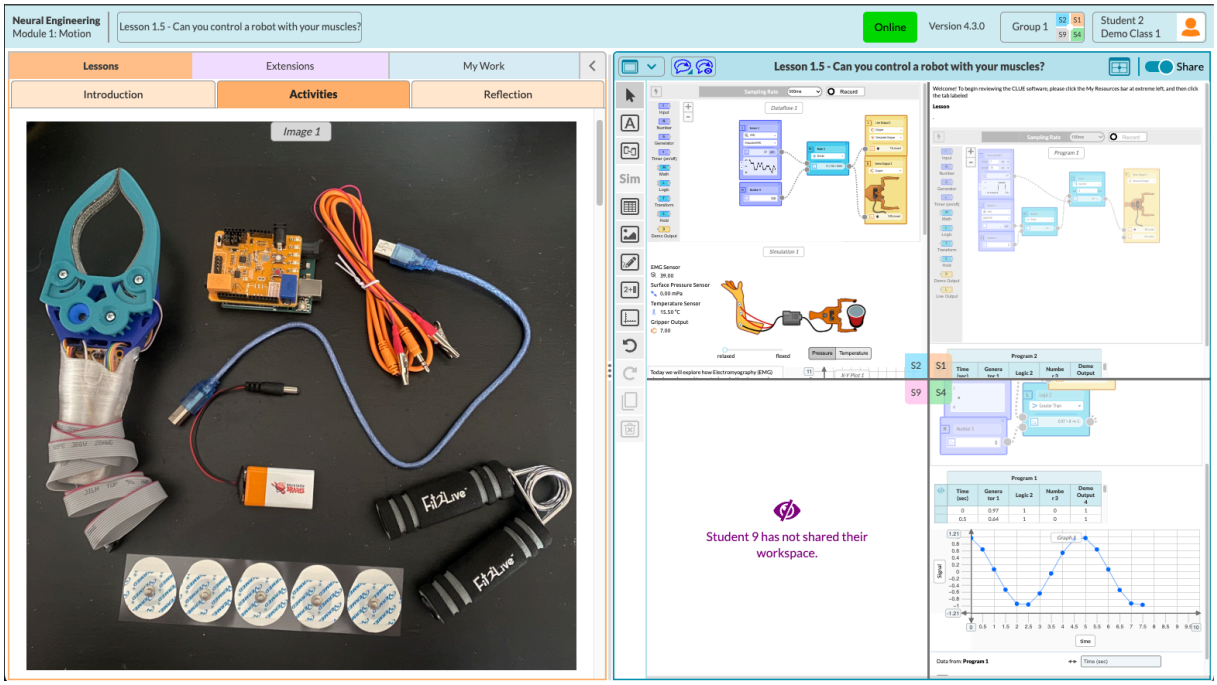

Figure 1. A screen capture of CLUE software showing (a) the inline curriculum (left) with a photograph of the hardware kit, and (b) WYSIWIS collaborative student view (right) where groups can compare and reuse solutions as they evolve. Note digital twins in the upper left quadrant workspace.

In this iteration of the high school kits, the gripper was redesigned to a more robust version (Figure 1a) capable of successfully gripping and holding common lightweight objects such as a cup. It has integrated pressure sensors and an easy to grip handle that both contains and controls the wiring through a ribbon cable to an Arduino shield. These updates improve the robustness of the gripper system and facilitate student experimentation with the device.

integrated STEM workspace

As shown in Figure 1, the curricula and all student (and teacher) work for both of these units is captured in a proprietary web software system known as the Collaborative Learning User Environment (CLUE) [22], [23], [31]. Students can work independently or together in this system, as shown in Figure 1b, and all elements of student work, teacher work, or the curriculum can be reused by dragging them into the active workspace. This allows students to springboard their own work from examples shown in the curriculum, or to collaborate on programming solutions and evolve them naturally without having to recreate them. The system is also instrumented with a detailed event and learning object logging system that allows researchers to track the propagation of ideas across the classroom.

To support the open-ended STEM work that must take place within these and similar integrated STEM curricula, the CLUE system supports a variety of tools in addition to the hardware-aware programming language. Students can write descriptions, create drawings, make or export tables of data from programs, make graphs, and write mathematical expressions. We also worked to make the recording and exporting of data from Dataflow programs easier. It is conceptually important for students to recognize that programming is distinct from program results, which capture a single experimental run in time. Data are, therefore, a reflection of the physical measurements in combination with a version of the program that processed them. A recording mechanism in Dataflow freezes the program to match a set of data. Programs can be unfrozen by clearing their data to return to a workable version of the code, which may then produce different results from a previous recording. This distinction between writing and debugging the code (tinker mode) and experimental recording (runtime mode) is deliberate and important for students to internalize as they learn to document and analyze lab work with programs [23].

the Dataflow language

Dataflow was developed in its present form in 2021 as a student-friendly sensor listening and actuator control language [23]. The language is constructed with affordances that help students understand how to connect components with little coaching. Dataflow reads, writes, and calculates values instantaneously while the student is programming, and each node contains a ‘minigraph’ that helps students see the data samples as they propagate through each programmatic operation. Dataflow operations are specified by tool buttons grouped by functionality to help students understand the type of actions that can be performed (Figure 2). Students can read sensor inputs, enter integer or float numbers, and generate signals such as sine waves. Value nodes have one output to indicate they can be connected to other blocks that receive and operate on values. Dataflow also contains Function nodes, including arithmetic math, logic, transform, and the new **Hold** operation. These are nodes with one or two input connectors and a single output connector. Finally, there are **Demo** and **Live** output nodes, which offer a choice of simulated or live actuated devices. The **Demo** output nodes were the original method of introducing a programmable digital twin, but because their impact on each other and on sensor inputs was missing, we elected to make an external simulation that mimicked the end-to-end embodied hardware more precisely.

In the 2024 version of Dataflow we added two new functions to accommodate algorithmic problems in the high school curriculum and connectivity to Microbits for the middle school

curriculum. To address the second capstone challenge in the high school curriculum, students needed to control the closing gripper to hold but not crush objects in a manner that mimics a human hand. Student programs needed to stop the gripper from closing too aggressively around a cup by setting a threshold for a pressure sensor in the gripper. The intended logic was "open and close the gripper according to the muscle sensor, but when the pressure is too high, back off to the last successful position." This is the kind of logic that would typically be written with multiple lines in a control loop, but crafting these loops is challenging from both a conceptual point of view [32] and practical standpoint. We introduced a **Hold** node that takes two inputs, the incoming sensor signal and a control input. When the control is on—in this case, the pressure is too great—the node holds the output equal to the input signal sample (or the previous one, or zero). When the control is off, the input is simply passed through. We attempted to design this functional node in such a way that it would match what the student might phrase in their head about what the control must do.

The second specialty function introduced was a **Ramp** function on the **Transforms** node. Significant timing delays exist between the servo in the gripper, the communication to and from the laptop, and the delay introduced by running code in a web browser. If students moved their arm quickly, either in physical or twin experiments, the time delay between exceeding the pressure threshold and sending the instruction to stop squeezing was too long to prevent an object from being crushed. A ramp allows students to moderate any signal that changes value quickly (defined as more than 10% change from sample to sample magnitude) to automatically be transitioned more gradually over 10 samples.

For the middle school curriculum, we introduced dedicated sensor output options for the IoT micro:bit-based sensor configurations used by students. These options contain channel selection capabilities enabling students to choose which channel a connected base station was communicating across in order to match a remote micro:bit receiver attached to one of multiple actuated terrarium setups located at another location in the room. By specifying the intended recipient, students can use the node and attached communicator bi-directionally, receiving input signals regarding the current environmental conditions within the terrarium and responding in kind by turning off or on actuated relays to control humidity and/or temperature within the terrarium.

digital twins

In previous implementations of both curricula, in particular the high school curriculum [20], [21], students struggled with distinguishing between issues caused by the hardware and issues caused by their own software solutions. This is also an issue for teachers of this curriculum, who are not well-versed in sensor systems or programming since they specialize in biology. To aid this issue, we introduced a set of programmable digital twins that would reliably behave as correctly instrumented hardware. Figure 2 shows Dataflow being used to program the digital twin sensor and gripper system developed for the high school unit.

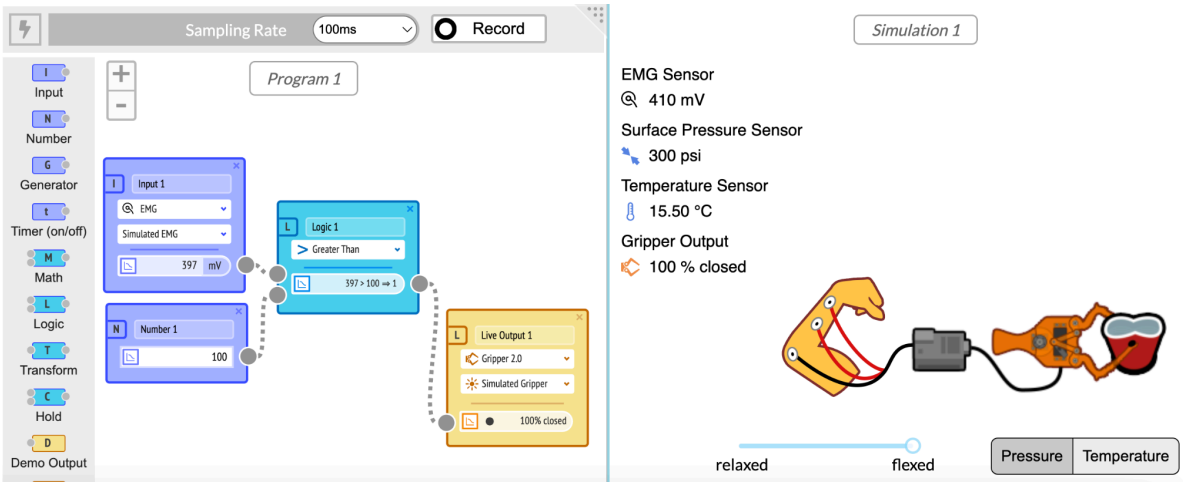

Figure 2. Dataflow programming (left) of a digital twin simulation (right) that allows debugging of the program. Dataflow functions are shown in the toolbar at the far left.

For the middle school unit, we developed a simulated terrarium environment that relays simulated measurements of temperature and humidity conditions, and the health of a virtual gecko. This digital twin also indicates visually when an actuator has been triggered, signaling to students that they are adjusting the temperature and humidity within the environment. An important aspect of this digital twin is its built-in buffering design, which simulates the delays in response of the terrarium environment to the actuated changes in temperature, driven by the heat lamp, or humidity, driven by the combination of the humidifier and fan. Incorporating these into the digital twin environment is important not only for added relevance and physical verisimilitude, but also because recognizing and accounting for such buffering and delay is an essential aspect in designing feedback-control systems.

Digital twins in these contexts provide many benefits to students and teachers, including: (1) *Separation of concerns.* Learners can test program logic without worrying about whether hardware is properly installed, wired, or powered. All of these conditions were observed in live classroom testing; (2) *Expanded access.* Even in classrooms with kits, typically not every learner has a kit, and sharing can make lessons chaotic. Digital twins bridge this gap by enabling learners without access to hardware [26]. This feature was critical in those cases where the hardware kits failed or were impossible to debug, an occurrence often related to power issues; and (3) *Expanded understanding.* Observing the parts of the system encoded or read from the twin helps learners gain understanding of the boundary between software and hardware [22].

research questions and class testing

RQ1: Can students conduct CT more efficiently with simulated hardware for program debugging?

RQ2: Can biology-based units successfully include CT affordances for feedback loops such that non-major students can code algorithms?

RQ3: Can simulated hardware setups assist students in gaining understanding of programming and control of physical systems?

methods

The high school study involved 215 high school students who participated in the CT unit as part of a biology course in the 2023-24 school year. The students were from five public schools in the northeastern United States. Most of them were in 11th (35%) or 12th (43%) grade. All students provided written assent and had permission from their caregivers to participate in the study. All study protocols were reviewed and approved by an Institutional Review Board.

Research was conducted by review of classroom observation notes, student work, including student programs, activity log files, and surveys. Student work files were analyzed in response to the two capstone curriculum efforts for quality and correctness of programs, repetition of demonstrated programs vs. new/creative program construction, and text descriptions of program operation.

student high school algorithm results

Many students reused the program presented in the first CT capstone lesson. This program naively compares the level of the EMG signal to a threshold: if the EMG is greater than the threshold, the gripper closes, and vice versa. This produces a binary output (on/off) or open/closed setting for the gripper rather than a full mapping of muscle contraction to percentage closed for the servo. Most student programs across all classes replicated this program identically when attaching the EMG sensor with the gripper without changes or exploration. The result is that the gripper was either open or closed entirely with no intermediary subtlety that maps to the degree of muscle clench. The resulting behavior of the digital twin and the hardware appears buggy and laggy in this scenario, both because of the natural delay introduced by feeding signals in and out of the computer, and because neither the digital nor the physical servo can respond instantaneously, so an ‘open’ or ‘close’ instruction will take a few hundred milliseconds to reach its fully open or closed position, regardless of what the student is doing with their arm.

At the end of the first capstone lesson, students were asked to elaborate on the program and the hardware system, describe its good and poor properties, relate it to the realistic task of creating a robotic arm, and propose enhancements. From the analysis of student documents, 32% of the students made note of the jerky, laggy response of the system, and 16% of the students were able to connect that behavior to the logic of the program suggested in the curriculum. Below are some examples of statements made by students referencing the limitations of the program.

> *“the robotic gripper starts open and if the EMG signals reach past a certain threshold (like say 30 or something), the robot gripper will close”*
>
> *“I would make it so that the thing that receives my electrical signal, measures it (the measurement would determine how hard or much to contract, since if I contracted harder or more, the EMG read a bigger signal), then send a proportionally sized signal to the gripper, so that it matches the amount and strength of contraction in my hand.”*

“Too sensitive. The threshold is only one number and the arm responds to the slightest adjustment in grip. This can lead to it open and closing wildly if movements aren't overtly deliberate.”

“goes from 0 to hundred - no in between”

“not direct identical movement”

“has limited movement or use, since it can only either fully open or close and doesn't have a range”

“the gripper didn’t match the flexing of the arm”

“does not have a variety of movement like the arm, only opens and closes”

“With regular movement, the gripper will immediately open and close, not hold one action”

While a third of the students noticed the jerky, laggy motion created by the program, and one sixth of them could describe the deficiency of the program, none of the students changed the logic of the program or showed any other evidence that they knew how to fix the issue they had rightly observed. A handful of students had programs or additional programming nodes that fed continuous signals into the gripper, either the digital twin or the physical hardware, but none made the leap to change the algorithm.

In the second capstone challenge, a program mapping the full range of motion from the arm to the gripper was demonstrated within the curriculum by dividing the EMG signal by its maximum value, thereby setting the range of values smoothly between 0 and 1. The new functions for ramps (slow transition) and holds (feedback logic) were scaffolded, with prompts that asked students to analyze and experiment with the behavior of each of these functions. Students were not specifically instructed how the functions were to be used. Creating a program to grip but not crush requires the normalized mapped input and a second pressure sensor with a threshold to stop the EMG signal when the gripper has closed too far. If the gripper is to reopen, a second detection must determine whether the EMG indicates the arm is opening, and this is used to turn the **Hold** function off.

Of the documents reviewed, 62% of students were able to correctly use the **Hold** and **Ramp** functions to create a program that closed but did not crush the cup. This logic was completed using the digital twin in most of those cases. There appears to be some correlation between whether the teacher in the class had created an example program that showed this construction and the success of the students, but the evidence is circumstantial and would require further investigation to establish order of creation of programs with respect to the timing of the teacher demonstration.

Every one of the programs in the more complex challenge was accompanied by a digital twin simulating the arm and the gripper. While most students did eventually make their programs work with hardware, classroom observation and historical development of programs showed that the twin was an easier way to get the logic working before progressing to the hardware.

At the conclusion of the second capstone lesson, students were again asked to reflect upon the success of the program and what they would do differently. Only one student showed logic that opened the gripper after having it close without crushing the cup. Among all other students, none acknowledged that additional logic was required to reopen the gripper once it closed. The general principle of creating a threshold for a pressure sensor was well understood by the 62% of students who made a program with a feedback loop. The programs sometimes did not achieve the intended result but in principle the logic was sound, and these students were further able to describe what they were trying to achieve.

> *"Use the less to make the gripper NOT crush the cup"*
>
> *"the hold block can open and close the gripper when off but when its on it keeps it open stopping the code from going through it"*
>
> *"when i tried to grab the plastic cup, it crushed since it had a lot of pressure in the hand and there was no limit so i had to put a hold block so that way, it can limit the amount of pressure so that way the cup doesn't crush while holding it"*
>
> *"At first when i tried to grab the plastic cup, the cup got crushed. To fix the problem i added a hold block so when it gets to a certain amount of pressure the sensor tells the simulation to not crush the cup."*
>
> *"When I tried to grab the plastic cup, it would stop before the cup was completely crushed. If the cup got crushed, you need to add a hold block that stops sending pressure once it's at a certain pressure."*

student and teacher high school hardware results

The hardware system, sensors, Arduino, shield, and gripper were successful in so far as all students were able to get the gripper to successfully open and close in a straight pass-through scenario in the first capstone exercise. The moment of success in class was observed to be typically jubilant—the embodied reaction of the hardware is very satisfying. The project is also multi-component, which makes it a good team exercise.

Hardware debugging was challenging for students, teachers, and researchers who were observing in the classroom. When the system was wired correctly with fresh batteries, things tended to work. If, however, a connection was missing (failing to connect power to the gripper or a loose connection) or if the battery was inadequately charged, the resulting system errors presented as if they were programming problems. Students reported that their EMG signal stopped reading, or that the gripper would work according to the program momentarily and then fail. These issues were reflective of a lack of power but there was no instrumentation to demonstrate that issue either in the hardware or software.

middle school results

The middle school unit, performed in two iterations a year apart in spring of 2023 and 2024, was included as a module within a seventh grade engineering class at a suburban northeastern middle school. The first implementation included 35 students in a version without a digital twin. In this version of the unit, only three or four hardware setups were available across a classroom, and individual students could not test their control programs on a live control setup without

recreating them on or porting them to a computer connected to a dedicated microcontroller. As a result, only a handful of students were able to successfully generate and execute a program that operated a live control setup. Further, during the times when other students were programming or troubleshooting live-control programs, the majority of students in a group were unable to view meaningful output from their programs, causing lack of engagement and limited ability to create programs that reinforced the unit's goal concepts of sensing, control, and feedback.

The second iteration of the curriculum included a digital twin of the terrarium setup with virtual sensors and actuated fans, lights, and humidifiers pictured in Figure 3. In this case, students could create, test, and troubleshoot their programs and experience the key concepts of sensing, control, and feedback without the need to wait for access to a hardware setup. While some students still struggled with programming, in the second iteration 73% of students created a program that engaged the digital twin with at least one control or sensor block. The majority of these programs included correctly functioning control logic, with one third of students completing a fully functional control scheme accounting for feedback-based control of the intended parameters.

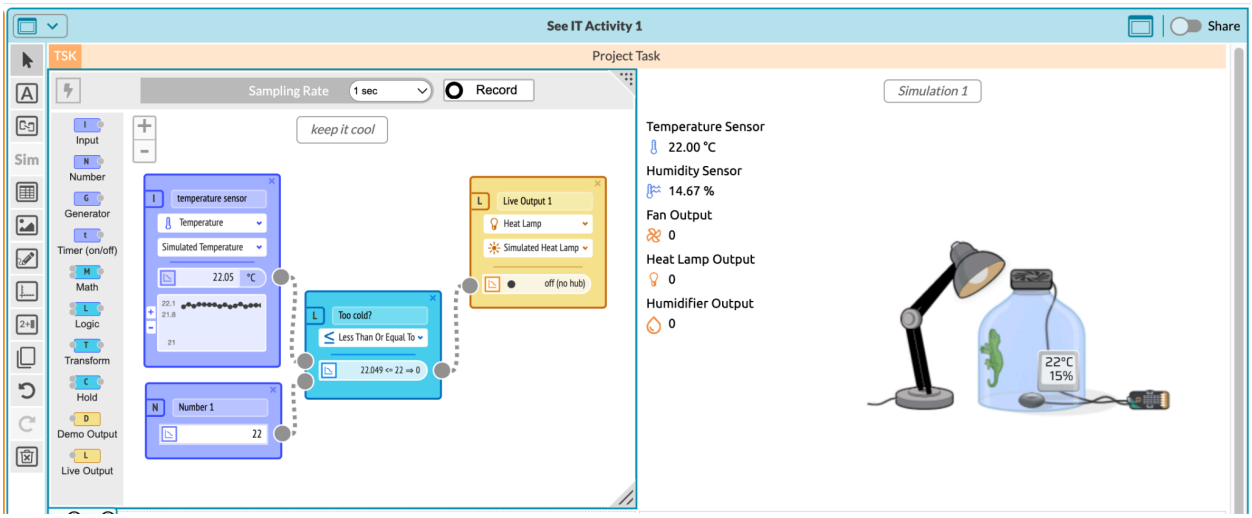

Figure 3. The terrarium simulation controlled by a Dataflow program to monitor temperature.

From the standpoint of the overall curricular experience, the digital twin appeared to offer a more optimal instructional flow, enabling students to prepare, test, and debug programs in advance of working with the hardware setup. Upon encountering the hardware setup, students appeared to have a more complete understanding of the control and feedback mechanisms involved and could more easily transition to engagement with the mechanics of the hardware.

analysis

A full thematic analysis of the high school student documents, action logs, and associated focus group and observational data is still ongoing. Following Saldaña's [33] two-cycle coding to thematize the data, we started with an initial coding strategy. In this paper, the coding focuses on successful programming outcomes and a stated understanding of the intended function and deficiencies of programs.

What is apparent from the class work is that students seem to understand the general steps required by a digital system in order to achieve an engineered outcome, but they are unable in some cases to put those ideas into practice. They seem to be able to trace and reuse logic that is presented to them, but do not expand upon that logic even though the tools required to do so are readily available. They are also unable to successfully debug programs that are failing. There was also very little exploration of different logical solutions or the functioning of the program tools unless specifically instructed in the curriculum. We speculate that some time needs to be built in for students to construct simpler programs of their own devising so they can become comfortable with the tools.

Specifically, students need more scaffolding around the values and logic required to manipulate a quasi-continuous physical device such as the gripper. We chose to make this device have a range between 0-1. It is unclear if a range of 0-100 would have been easier or harder to understand. Note that output blocks were introduced with a binary lightbulb, which specifically maps 0 to off and 1 to on, with no intermediary values. This first impression seemed to stick with students until they were prompted to experiment more completely with the gripper and a known input such as a sine wave using the signal block.

Although the normalization of the EMG signal into a 0-1 range to run the gripper was covered in detail, it was not justified or elaborated. Many students were unable to understand why a program that feeds the EMG signal directly to the gripper closes but never opens it (the EMG signal has a baseline of 39, so it never reaches 0). Likewise, understanding that there are two conditions that must be reached simultaneously to successfully open and close a gripper (pressure not too high, EMG signal not too low) is a level of computational design that appeared beyond these students. Students who are programming microprocessors and associated sensors and actuators must learn to translate between physical values and arbitrary thresholds, such as the one above, and the normalized, quantized values that function in a program. This concept does not appear to be natural or apparent to the students in these classes.

Embodied engineering is important for motivation. Our findings, based on classroom observation, student focus groups, student programs, and student comments in their work, indicate that students were highly engaged by the authentic engineering design process, and that the simulated digital twin of the hardware setup was necessary to their success [34].

When teachers played with and presumably demonstrated programs (which we could tell from their own documents), the results in the student work were far superior to that of their peers in classes without teachers who knew how to program. Student work was typically very consistent with the teacher's work without attempting different logical paths. Students seemed to be following patterns dictated by the teacher and programs were very uniform.

Many students commented on the jerkiness of the resulting hardware and how it did not reasonably respond to their hand motions or that they had to squeeze very hard to make the gripper close. In classroom observations where a more experienced teacher or team member was present, students who were tutored in the mapping of sensor dynamic range to the range of the gripper positions were easily able to understand the concept, but needed to be shown the technique. The same is true of the full control logic program. Students seem able to understand

the flow of logic and are able to describe it, but creating or modifying it does not seem within their grasp (or grip) under the current scaffolding or tools.

The middle school curriculum participants completed a pre-post test including an open response item querying them about how to create a setup to control an environment for a pet lizard to keep its conditions within a set of parameter values. In both iterations of the curriculum, students were more likely to describe scenarios involving feedback mechanisms in the post-test than the pre-test. Only 12% in the first iteration and none of the students in the second iteration included feedback concepts on the pre-test. One third of the students in both iterations included scenarios incorporating feedback concepts on the post-test. The effects of the digital twin iteration of the curriculum are most apparent in the depth of students' responses. In the second iteration, 24% of the student responses included complete and functional descriptions including quantitative references with the inclusion of sensing inputs, actuated outputs, feedback mechanisms, and timing.

conclusion

Integrated CT STEM courses are challenging to construct because of the difficulty of choosing subjects that lend themselves both to science standards and programming or modeling and because of the challenge of scaffolding all concepts in a curriculum that can be taught in a few weeks. In the high school curriculum we have achieved significant engagement of students with the biology of muscle motion and the methods by which engineers might engage in helping people with motor problems. In the middle school curriculum we are similarly engaging students with a desktop terrarium and pet care scenario. However, students are unlikely to reach a facility with algorithm design without additional scaffolding in these curricula. In particular, CT solutions to life science examples require a nuanced understanding of which values of what size and type are required at any point in an embodied or twinned control program. Students must know how to navigate the divide between the values that represent a physical device or system and the binary or at least quantized values that function in a program. To successfully teach these concepts we would need to expand the curriculum and include practice with them. We expect to extend this part of the work with a future grant incorporating AI to step in at critical moments to help move these concepts forward even when the teacher in the classroom is unfamiliar with them.

In addition to updates to incrementally introduce coding conventions and control concepts, we should also refine the hardware to report various connection issues, particularly low or no power states so they can be debugged. This awareness would need to be reflected in the digital twin as well so students could gain experience with wiring and hardware debugging in a reliable environment.

Student struggles with debugging and creating programs from scratch further suggests that for non-major classes, some programming affordances are critical for getting students past the stumbling blocks that prevent them from engaging fully with a lesson, and that scaffolding around unfamiliar concepts such as thresholding and feedback is necessary. These affordances include both convenience functions within the programming language, hardware and integrated feedback about common issues, and digital twins to separate debugging concerns. There is

additional work to be done to find a suitable split between raw logical commands and domain-specific programming functions that obscure the logic but may help with CT.

We have only started experimenting with digital twins and associated microprocessor hardware systems in the classroom, but we have many questions about what is important vs. what is distracting. For example, as seen in Figures 2 and 3, we have endeavored to make our twins not only codable and measurable but also animated and physically reflective of the 'kit' in the classroom. Digital twin implementations in other projects (e.g., [26], [27], [28]) are often oriented around visualizations or models to allow statistical or measurement reflection on CT. So far, our twins are constrained to the choices we have made as programmers and designers about what can be changed and what is displayed. More thought work is required to define the most successful features for educational digital twins.

Students who engaged with this material found it interesting and relevant based on feedback in their documents and comments in classroom observation and from focus groups. The curricula definitely achieved the goal of providing motivation for engaging with future STEM and CT activities. These findings suggest that biological systems coupled with microprocessor-driven sensor technology can provide an effective context for integrating life sciences and engineering. The methodology is worth pursuing, but more work remains to make the curriculum more practical for teaching the mechanics of programming in a wider classroom context. The existing curriculum and software can be reviewed and assigned from https://learn.concord.org/neural-engineering.

Acknowledgments

This material is based on work supported by the National Science Foundation under grant nos. DRL-2101615 and DRL-2054079. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

The authors would like to thank the Concord Consortium software developers, in particular Michael Tirenin for user interface design, and Joe Bacal, Teale Fristoe and Scott Cytacki for design and implementation of the digital twin interface to Dataflow.

References

[1] P. B. Henderson, T. J. Cortina, O. Hazzan, and J. Wing, "Computational thinking," in I. Russell & S. Haller (Eds.), *Proceedings of the 38th ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '07)*, pp. 195-196https://doi.org/10.1145/1227310.1227378

[2] S. Grover and R. Pea, "Computational thinking in K-12: A review of the state of the field," *Educational Researcher*, vol. 42, no. 1, pp. 38–43, 2013.

[3] D. Weintrop, E. Beheshti, M. Horn, K. Orton, K. Jona, L. Trouille, and U. Wilensky, "Defining computational thinking for mathematics and science classrooms," *J Sci Educ Technol, vol.* 25, pp. 127–147, 2016. https://doi.org/10.1007/s10956-015-9581-5

[4] U. Wilensky and W. Stroup, "Networked gridlock: Students enacting complex dynamic phenomena with the HubNet architecture," in B. Fishman & S. O'Connor-Divelbiss (Eds.), *Fourth International Conference of the Learning Sciences.* Erlbaum, 2000, pp. 282-289.

[5] B. Reiser, M. Novak, and T. McGill, "Coherence from the students' perspective: Why the vision of the framework for K-12 science requires more than simply 'combining' three dimensions of science learning," Commissioned for the Board on Science Education Workshop Instructional Materials for the Next Generation Science Standards, 2017. Available: https://sites.nationalacademies.org/DBASSE/BOSE/DBASSE_180249. [Accessed Jan. 3, 2023].

[6] G. Roehrig, E. Dare, E. Ring-Whalen, and J. Wieselmann, "Understanding coherence and integration in integrated STEM curriculum," *Int. J. STEM Educ.* vol. 8, no. 2, 2021. https://doi.org/10.1186/s40594-020-00259-8

[7] M. Mataric, "Robotics education for all ages," In *Proc. AAAI Spring Symposium on Accessible, Hands-on AI and Robotics Education,* 2004.

[8] I. Lee, F. Martin, and K. Apone, "Integrating computational thinking across the k-8 curriculum," *ACM Inroads,* vol. 5, no. 4, pp. 64-71, 2014. DOI: 10.1145/2684721.2684736

[9] Code.org, CSTA, and ECEP Alliance, "State of computer science education 2024," Accessed from https://advocacy.code.org/stateofcs/ Jan. 12, 2025.

[10] M. Papastergiou, "Are computer science and information technology still masculine fields? High school students' perceptions and career choices," *Computers & Education,* vol. 51, no. 2, pp. 594–608, 2008.

[11] J. Margolis, *Stuck in the Shallow End: Education, Race, and Computing.* Cambridge, MA: MIT Press, 2008.

[12] A. B. Diekman, E. R. Brown, A. M. Johnston, and E. K. Clark, "Seeking congruity between goals and roles: A new look at why women opt out of science, technology, engineering, and mathematics careers," *Psychological Science,* vol. 21, no. 8, pp. 1051-105, 2010. https://doi.org/10.1177/0956797610377342

[13] F. Levy, and R. Murnane, *Dancing with Robots: Human Skills for Computerized Work* Washington, DC: Third Way NEXT, 2013. http://www.thirdway.org/publications/714

[14] J.A. Dole and G.M.Sinatra, Reconceptalizing change in the cognitive construction of knowledge. *Educational Psychologist*, vol. 33, no. 2–3, pp. 109–128, 1998. https://doi.org/10.1080/00461520.1998.9653294

[15] J. Walther, M. Brewer, N. Sochacka, and S. Miller, “Empathy and engineering formation,” *Journal of Engineering Education*, vol. 109, no. 1, pp. 11-33, Jan 2020. DOI: 10.1002/jee.20301.

[16] National Science Foundation. “Celebrating two years of ‘CHIPS and science,’ 2024,” https://nsf-gov-resources.nsf.gov/files/CHIPS_and_Science_2_year_fact_sheet.pdf?VersionId=5xBOdwtAPnUeVLBSbzWAtuORXUgcj9Et, Accessed Dec. 10, 2024.

[17] G. J. Gage, “The case for neuroscience research in the classroom,” *Neuron*, vol. 102, no. 5, pp. 914-917, 2019. https://doi.org/10.1016/j.neuron.2019.04.007.

[18] A. Chakarov, Q. Biddy, J. Jacobs, M. Recker, and T. Sumner, “Opening the black box: Investigating student understanding of data displays using programmable sensor technology,” *Proc. 2020 ACM Conf on Intl Computing Education Research (ICER '20).* Association for Computing Machinery, New York, NY, pp. 291–301, 2020. https://doi.org/10.1145/3372782.3406268

[19] M. Lewis, A. Holloman, B. Hernández-Cuevas, and C. Crawford, “Exploring computational thinking perspectives in black communities with physiological computing,” *Black Issues in Computing Education (BICE),* pp. 27-32, 2024.

[20] T. Aldemir, I. Davidesco, S. Kelly, N. Glaser, A. M. Kyle, B. Montrosse-Moorhead, and K. Lane, “Investigating students’ learning experiences in a neural engineering integrated STEM high school curriculum,” *Educ. Sci.* vol. 12, no. 10, pp. 705, 2022. https://doi.org/10.3390/ educsci12100705.

[21] M. K. Coburn, I. Davidesco, A. M. Kyle, J. Bacal, and L. Bondaryk, “Neural Engineering Collection Page”, retrieved Feb. 21, 2025 from https://learn.concord.org/neural-engineering.

[22] C. Dorsey, “Systems engineering: Design challenges for the internet of things,” *@Concord*, Spring 2023. Retrieved Jan. 14, 2024 from https://concord.org/newsletter/2023-spring/systems-engineering-design-challenges-for-the-internet-of-things/

[23] L. Bondaryk, S. Hsi, and S. Van Doren, "Probeware for the modern era: IoT dataflow system design for secondary classrooms," *IEEE Transactions on Learning Technologies*, vol. 14, no. 2, pp. 226-237, 2021. doi: 10.1109/TLT.2021.3061040.

[24] S. Zacher, “Digital twins for education and study of engineering sciences,” *Intl Journal on Engineering, Science and Technology*, vol. 2, no.2, pp. 34-42, 2020. https://doi.org/10.46328/ijonest.40

[25] S. Boschert and R. Rosen, “Digital twin — the simulation aspect.” In: P. Hehenberger, and D. Bradley. (eds) *Mechatronic Futures*. Springer, Cham,2016. https://doi.org/10.1007/978-3-319-32156-1_5

[26] O. F. Caribo, L. Sibomana, J. C. Byungura, B. K. Asingwire, and C. Niyizamwiyitira, "Digital twins applications in STEM education: Challenges and implementation opportunities in developing countries," *2024 IEEE Digital Education and MOOCS Conference (DEMOcon),* Atlanta, GA, USA, 2024, pp. 1-6. doi: 10.1109/DEMOcon63027.2024.10747951

[27] C. Lore, H. S. Lee, A. Pallant, C. Connor, and J. Chao, “Integrating computational thinking into geoscientific inquiry about volcanic eruption hazards and risks,” *Int J of Sci and Math Educ* vol. 22, pp. 1173–1195, 2024. https://doi.org/10.1007/s10763-023-10426-2

[28] A. Samarapungavan, L. A. Bryan, C. Staudt, B. Sapkota, H. E. Pinto, H. E., J. M. Broadhead, and N. Kimball, “Using technology‐mediated inquiry to help young learners reimagine the visible world through simple particle models.” *Journal of Research in Science Teaching,* vol. 60, no. 2, pp. 390-422, 2023.

[29] J. Roschelle, W. Penuel, and N. Shechtman, “Co-design of innovations with teachers: Definition and dynamics,” In S.A Barab, , K.E. Hay, and D.T. Hickey (Eds.), in *Proceedings Intl Conference of the Learning Sciences: Indiana University 2006.* vol. 2, pp. 606-612.

[30] P. Blikstein, “Bifocal modeling: A study on the learning outcomes of comparing physical and computational models linked in real time,” In *Proceedings of the 14th ACM international conference on Multimodal interaction (ICMI '12).* pp. 257–264. https://doi.org/10.1145/2388676.2388729

[31] L. Bondaryk and C. Dorsey, “Aligning teacher facilitation tools with pedagogies in a real-time environment for mathematics team learning.” in L.O. Campbell, R. Hartshorne, and R.F. DeMara, (Eds), *Perspectives on Digitally-Mediated Team Learning. Educational Communications andTechnology: Issues and Innovations.* Springer, Cham, 2021. https://doi.org/10.1007/978-3-030-77614-5_1

[32] S. Grover, R. Pea, and S. Cooper, “Designing for deeper learning in a blended computer science course for middle school students,” *Computer Science Education,* vol. 25, no. 2, pp. 199–237, 2015.

[33] J. M. Saldaña, , 3rd ed.; SAGE Publications: London, UK, 2016.

[34] I. Davidesco, B. Montrose-Moorehead, and A. Kyle, “Fostering computational thinking through engineering design activities in a high school biology course,” *Proc. of the ICLS, Helsinki, Finland,* in press.