Examining the Design and Outcomes of an After-School Physical Computing Program in Middle School

Sotheara Veng
University of Delaware
United States
sotheara@udel.edu

Chrystalla Mouza
University of Illinois Urbana-Champaign
United States
cmouza@illinois.edu

Lori Pollock
University of Delaware
United States
pollock@udel.edu

Abstract: This work examines the application of high-quality pedagogical practices in the design and implementation of an after-school physical computing program aimed at providing middle school students with access to computer science (CS) education. It subsequently examines how the program influenced students' learning of CS concepts and attitudes towards computing. The program was designed and implemented through a school-university partnership, and 66 middle school students voluntarily participated. There were two cohorts of students in the study. Results indicate that the program had a positive impact on students' understanding of CS concepts and a significant impact on attitudes towards computing among students in the second cohort. Implications are drawn for the design of informal after-school programs aimed at broadening participation in computing.

Keywords: Computer science education, physical computing, computational thinking

Introduction

According to the U.S. Bureau of Labor Statistics (2022) employment in the computation and information technology field is projected to rise by 13% by 2030. With the growing digital economy, the Computer Science for All (CS4All) initiative was introduced by the White House to help equip students with computational thinking (CT) skills and broaden participation in computing (Smith, 2016). Yet, data continue to demonstrate that women and minoritized students are not well represented in computing (Cuny, 2012). Some of the factors include low self-confidence, gender stereotypes, and negative perceptions towards computer science (CS) (Cohoon & Aspray, 2008).

Prior research has shown the importance of engaging students in computing at a young age as this can influence their participation and help diversify the field (Ching et al., 2018). Middle school is an important period for such engagement as in this stage of development, children make important decisions on their own identities and develop perceptions of their ability, influencing their future career options (Brickhouse et al., 2000; Tang & Cook, 2001). Moreover, the use of physical materials can help motivate students to engage in computing (Sentance et al., 2017; Voštinár & Knežník, 2020). Although there have been different programs utilizing physical computing as a way of motivating diverse student participation (e.g., Cápay & Klimová, 2019; Shahin et al., 2022), few of those are accompanied by efforts to examine associated impacts on student outcomes.

To foster middle-school student participation in computing and address issues of gender equity, we designed a 6- week after-school program through a partnership between a university and a local teacher. The program is aimed at providing students with access to high quality CS education through block-based programming on a platform called MakeCode with the use of Micro:bit (https://makecode.microbit.org) – a pocket-size, interactive

computer. The program is delivered by CS undergraduates enrolled in a service-learning course. This study reports on the program by exploring two research questions:

1. How can short-term after-school programs that utilize physical computing be implemented to broaden middle-school student participation in CS?
2. How does participation in such programs impact middle school students' learning of CS concepts and attitudes toward computing?

Literature Review

Computational Thinking (CT) and Physical Computing in K-12 Education

CT is defined as "an approach to solving problems in a way that can be implemented with a computer" (Barr & Stephenson, 2011, p. 51), and is recognized as an important skill set in the 21st Century due to its benefits in the field of CS and beyond (Gretter & Yadav, 2016). Therefore, it should not only be taught at the tertiary level but at all levels of education (Wing, 2006). One way to teach CT is through physical computing devices such as Raspberry Pi, Arduino, and Micro:bit, which are programmable by students. According to constructivist learning theory, students learn by actively constructing their knowledge (Ben-Ari, 1998). Built on constructivism, the term "constructionism" has been coined to emphasize that learning occurs when learners actively construct something tangible, such as a computer artifact (Papert & Harel, 1991). Students' perspectives toward physical computing have been quite positive. It was preferred to traditional screen-based devises as it is tangible and fosters more open design without many restrictions (Devine et al., 2019; Sentance & Schwiderski-Grosche, 2012).

Micro:bit

Micro:bit was used in our work due to its functionality. Micro:bit is a pocket-size codable device designed by the British Broadcasting Corporation (BBC) equipped with a processor, wireless connections, LED lights, a compass, a temperature sensor, and two programmable buttons (Ball et al., 2016). This device allows students to utilize novel inputs from their physical environment, such as motions and sounds, which may not be possible through coding alone. It can be programmed on a web-based coding environment called MakeCode. The block-based coding language used on MakeCode is aligned with CS concepts, such as data, variables, operators, loops, and if-then statements (Brennan & Resnick, 2012; Voštinár & Knežník, 2020). The use of block-based coding seeks to promote creativity and increase the participation of students from diverse groups in computing (Ball et al., 2016).

Significant progress has been made in investigating the potential of utilizing Micro:bit in CS education (e.g., Kalelioglu & Sentance, 2020; Videnovik et al., 2018). One area of literature focuses on the students' experiences using Micro:bit (Cápay & Klimová, 2019; Sentence et al., 2017; Videnovik et al., 2018; Voštinár & Knežník, 2020). It was shown that the tangibility of the device was perceived to be able to support students' learning (Cápay & Klimová, 2019; Sentance et al., 2017). Specifically, the ability to see and manipulate creations and engage physically with the device helped students understand programming better (Sentance et al., 2017) and provided an enjoyable learning experiences (Cápay & Klimová, 2019; Videnovik et al., 2018). The device also created opportunities for collaboration and creativity, which can influence students' motivation in learning CS (Sentence et al., 2017; Videnovik et al., 2018). Another body of literature investigates the teaching experience and practices of using Micro:bit to teach computing. Teachers' experience confirmed students' perspectives towards the tangibility of Micro:bit and its impact on student motivation. From the teachers' perspective, engaging with an actual device instead of a virtual simulation on other platforms (e.g., Scratch) could provide feedback that can be helpful for students' understanding (Kalelioglu & Sentance, 2020). Moreover, the tangibility can also be a great motivator for students as it brings the programming experience "more down to earth" (Kalelioglu & Sentance, 2020). In terms of teaching practices, using a mixture of different strategies was perceived to be effective (Kafai et al., 2014). Besides common strategies including demonstration, collaborative work, and verbal explanation (Kalelioglu & Sentance, 2020), teachers may make connections between Micro:bit and its usefulness in the real world (Sentance, Waite, Yeomans, et al., 2017).

Although this body of work provides important insights on student and teacher experiences. It does not examine the impact Micro:bit has on the development of students' attitudes towards computing, such as their confidence and their identities towards CS. Despite the potential of physical computing to broaden participation in

computing, the strong emphasis on programming, lack of social implications, and insufficient preparation may cause students to lose interest (Cuny, 2012). Moreover, students can have some misconceptions towards some CS concepts such as loops and variables (Grover & Basu, 2017). Therefore, it is important that learning experiences be prudently constructed to support student learning and broaden participation in computing. In our study, the after-school program was established through a school-university partnership paying close attention to high quality content and pedagogical practices described below. The study can benefit teachers in making decisions on how to establish high quality CS programs in K-12 classrooms, with an emphasis on the use of physical computational material.

Theoretical Framework

In this work, we utilize a theoretical framework previously utilized by the authors (Mouza et al., 2021), which is built around high-quality CS content and pedagogical practices to examine the design and delivery of CS programs in K-12 schools.

High-Quality Computer Science Content

Providing access to high-quality content is very important to broaden participation in CS as it can make CS more accessible for students. Such access refers to curricula and teaching practices in class (Lewis et al., 2019). High-quality CS curricula contain components related to big ideas of CS, such as creativity, data, abstraction, programming, and algorithms (College Board, 2017).

These curricula may also require teaching practices intended to engage students with CS, one of which is using metaphors to teach CS concepts. The use of metaphors helps students to better structure their ideas and think more clearly and directly about abstract concepts (Diéguez, 1988). Since CS concepts are abstract in nature, using metaphors, with which students are familiar, to connect with new programming concepts can foster their understanding (Hui & Umar, 2011; Sorva et al., 2013). Moreover, in order to write computer programs accurately, students must engage in precise, attentive, and well-directed thinking (Heintz et al., 2016). One approach to utilizing familiar metaphors is through a series of classroom activities, called “CS Unplugged.” CS Unplugged refers to activities developed to introduce CS concepts to students without the use of computers or any other digital devices (Bell et al., 2009). Unplugged activities have shown promising results in students’ attitudes in prior work (Gardeli & Vosinakis, 2017; Jiang & Wong, 2017).

Additionally, it is important to promote multiple solutions to a problem. This may be inconsistent with a more traditional way of learning, which often emphasizes one correct way to solve a problem. However, such traditional practices can have a negative impact on learner motivation and are contradictory to the practice of CS (Boaler, 2008; Shah et al., 2013). CS problems should foster flexible ways of forming multiple possible approaches to arriving at solutions (Ben-Ari, 1998; Shah et al., 2013).

High-Quality General Pedagogical Practices

Exploring constructivist pedagogical practices is important for enacting high-quality pedagogy in CS. Constructivists believe that learning is an active process of constructing knowledge and meaning and students do so through experiences and interactions with the world (Schunk, 2016). There are two perspectives stemming from constructivist epistemology: cognitive perspectives and sociocultural perspectives.

Deeply rooted in Piaget’s work (1972), cognitive perspectives center on how one learns by making connections between one’s prior knowledge and new information, by fitting new information into existing one, or adjusting the existing knowledge to accommodate new information. In class, students may come with prior knowledge and experience. Therefore, it is necessary that teachers can understand and track students’ progress in order to design practices appropriate for their level (Darling-Hammond et al., 2019). There are various methods through which CS instructors can assess their students, such as asking for a show of hands in response to a question, observing students' emotional reactions (e.g., frustration, joy, or engagement) while they work on coding tasks, and engaging in casual conversations with students as they code or debug their programs (Grover et al., 2020). By conducting formative assessments, teachers can provide feedback to students and make changes to instructional decisions (Black & Wiliam, 2009). The feedback given through formative assessment has a connection with

metacognition, which can assist in the self-regulation of learners and in teaching them how to effectively learn (Hudesman et al., 2013).

From a sociocultural perspective, rooted in Vygotsky's work (1978), the development of knowledge may occur through social interactions and collaborations. Such collaborations can be formed through contributing student pedagogy which promotes the act of contributing to peers' learning and valuing the contributions of other students (Hamer et al., 2012). Another form of collaborative work is called "Pair Programming", in which two students work together as a navigator and a driver with one computer to program (Denner et al., 2014). The driver directly writes the code, and the navigator evaluates the code and plans strategies to form solutions. Previous studies have indicated that when students engage in pair programming, they tend to exhibit enhanced competence in CS concepts and performance (Braught et al., 2011; Mendes et al., 2006) and experience an increased sense of enjoyment while programming (McDowell et al., 2003). Moreover, it is equally important to make lessons relevant to students' lives, interests, and aspects that are important to them (Ladson-Billings, 1995). This can help students develop a positive attitude toward computing and motivation to continue in the field (McGee et al., 2018).

Identity as a Computer Scientist

It is vital for students to identify themselves as capable of being computer scientists with dedication, time, and effort regardless of their gender and race. Their perception of themselves is strongly associated with how they engage in learning opportunities (Nasir & Cooks, 2009). There are practices that help students develop a positive self-perception in computing (Shah et al., 2013). First, students can be exposed to diverse groups of computer scientists. For example, there can be both male and female instructors in the class. This can help them develop a wider sense of who can be a computer scientist. Second, learning content can also be connected to students' identities outside of their classrooms as students tend to be engaged more in learning activities that are more connected to their own identities (Cobb & Hodge, 2002). For example, this connection can be fostered by providing students the freedom to design CS projects or artifacts of their own choice.

Methods

Settings and Participants

Our study was conducted as part of an after-school computing program at a middle school. The program is a collaboration among university faculty, CS undergraduates, and middle school teachers and students (Grades 4 to 6). Due to increased demand, there were two cohorts in the program. Each group met in a private classroom.

A total of nine CS undergraduates participated in the program by serving as instructors. They were supported by a group of volunteer parents. The undergraduates were enrolled in a college service-learning course focusing on engaging youth in computing (Mouza et al., 2016, 2021; Pollock et al., 2015). Prior to working with the middle school students, CS undergraduates met with faculty members and the middle school teacher to design lessons appropriate for the students' age group and CS experience and discuss teaching strategies that encourage participation in computing. CS undergraduates were also required to submit weekly journals reflecting on their teaching experience. The program was 6-weeks long, with each weekly session lasting 90 minutes and focusing on different CS concepts, including variables, conditionals, radio, loops, and while-loops. In the final session, students developed and showcased their own projects (see Appendix A). The content of the program was the same for both cohorts.

Data Collection

Observations of After-School Program. All sessions were observed and documented through field notes recording the CS concepts covered, description of teaching techniques and activities, interactions between undergraduates and middle school students, and materials used in the sessions.

Reflective Journals. Undergraduates submitted their weekly reflective journals after each teaching session. The 53 journal entries include narrative descriptions of their teaching and experience (e.g., what worked well, what did not work well), and recommendations for future sessions.

Middle school students' CS artifacts. Only original CS coding products were collected, which means that they did not include any remixed products or a product that was created using a tutorial. Evaluating students' products provided information regarding their learning of CS concepts and practices.
Pre/Post surveys for middle school students. A pre/post survey using a Likert scale format developed by Ericson and McKlin (2012) was used to examine changes in students' attitudes towards computing. The survey is organized around seven constructs (36 items) considered to encourage under-represented students to continue in computing, including (a) computing confidence, (b) computer enjoyment, (c) computer importance and perceived usefulness of computing, (d) motivation to succeed in computing, (e) computing identity and belongingness; (f) gender equity, and (g) intention to persist. The survey was administered at the beginning and the end of the program for each cohort.

Data Analysis

Classroom observations and reflective journals were analyzed using qualitative methods. Codes were generated for CS concepts and pedagogical practices as the observations and journals were read. Then, the codes were organized into codes and subcodes and paired with our theoretical framework as shown in Table 1 below.

Table 1. Coding Scheme

Codes	Subcodes
High-Quality CS Content	Using Metaphors to Teach CS Concepts
	Promoting Multiple Solutions to a Problem
High-Quality Pedagogical Practices	Tracking Student Progress
	Customizing Teaching Plans for Students
	Fostering collaborations
Identity as a Computer Scientist	Undergraduate Students and Parents as Role Models
	Encouraging Connections to Students' Identities

Data received from the pre/post surveys on students' attitudes towards computing were entered into a spreadsheet. A one-way ANOVA was conducted to assess the difference in pre-test results between the two cohorts. The calculation of means and standard deviations and paired sample t-tests were conducted on all seven constructs to measure the significance of the changes in students' attitudes before and after participation in the program. Finally, students' CS artifacts ($N = 61$) were evaluated using an evaluation scheme developed by Denner et al. (2012) and subsequently modified by Mouza et al. (2016). The evaluation scheme consists of three categories: (a) CS programming concepts represented in artifacts (i.e., data, conditionals. Radio, loops, operators); (b) code documentation and organization (e.g., naming variables); and (c) designing for usability (e.g., functionality).

Results

How can short-term after-school programs that utilize physical computing be implemented to broaden middle-school student participation in CS?

Findings indicate that CS undergraduates implemented both high-quality CS content and pedagogical practices while providing students with opportunities to develop their identities as computer scientists.

High-Quality CS Content: *Using Metaphors to Teach CS Concepts.* Undergraduates focused on fundamental concepts of physical computing, including variables, conditionals, loops, and radio. For instance, in Week 2, an undergraduate described teaching the concept of "Radio":

> The students were split into two groups. Each student was given a number. The students from one group had to send a message to the other group member with their matching numbers. If they picked the wrong number, the message would not be sent. This taught the students that you can only send messages to something if you are on the same wavelength.

High-Quality CS Content: Promoting Multiple Solutions to A Problem. In Week 1, students brainstormed ways to interact with imaginary friends. Micro:bits were then programmed into their virtual pet friends based on their

preferred quality of imaginary friends. According to observation data, students were given the freedom to customize the function of their Micro:bit pets as long as they utilize the input and variable functions on MakeCode.

High-Quality Pedagogical Practices: Customizing Teaching Plans According to Students' Progress. There were two main pedagogical practices used to support quality instruction: tracking student progress and customizing teaching plans for individual students. According to observation data, undergraduates assessed middle school students' understanding through interactions in class and analyses of their programs after each session using the evaluation scheme adapted from Denner et al. (2012) and Mouza et al. (2016). This type of formative assessment allowed them to adjust their lesson plans within the session or in subsequent sessions to meet students' needs.

High-Quality Pedagogical Practices: Fostering Collaboration. Observation data indicated that the undergraduates also modeled collaboration with a volunteer parent facilitator as well as modeled pair programming. They also established peer interactions in the sessions. An undergraduate student described a CS unplugged activity teaching the concept of "Conditionals": "I introduced the if/then game, each pair of students was given five note cards and got to take turns performing actions that reinforced the learning of if/then statements. The students worked in pairs, as drivers and navigators, for most of the lesson."

Identity as a Computer Scientist: Undergraduates and Parents as Role Models. There were gender representations. Both male and female undergraduates were selected. There were also volunteer parents of both genders and different races whose careers are in CS who also facilitated every session.

Identity as a Computer Scientist: Encouraging Connections to Students' Identities. In the final week of the program, students had the opportunity to develop their projects based on what they had learned in the program. According to observation data and student artifacts, a student, for instance, integrated her music background into her project by adding the function to play musical notes on her Micro:bit.

How does participation in such programs impact middle school students' learning of CS concepts and attitudes toward computing?

Learning of CS Concepts. Student learning of CS concepts and their uses are assessed by examining the presence of programming blocks in their final CS artifacts (i.e., Micro:bit programs; $N = 61$). There were 31 artifacts in Cohort 1 and 30 artifacts in Cohort 2. As indicated in Table 2, students performed similarly in both cohorts. Most students demonstrated competency with all CS concepts used in their programs; however, "Loops" had the lowest percentage of accuracy (86.36% in Cohort 1 and 85.71% in Cohort 2). Moreover, only a small number of projects (12.90% in Cohort 1 and 16.13% in Cohort 2) utilized the "Radio" concept, which emphasizes the ability to communicate among different devices through electromagnetic waves. When used, all "Radio" codes were utilized correctly. Besides CS Concepts, the artifacts were also assessed based on two dimensions: (a) code documentation and organization, and (b) design. The majority of the projects functioned as intended by the students and did not contain inappropriate or wrongly named variables, unnecessary scripts, or any glitches.

Table 2. Students' Micro:bit Project Evaluation ($N = 61$)

Criterion 1: CS Programming – Utilizing CS Concepts				
	Cohort 1 ($N = 31$)		Cohort 2 ($N = 30$)	
Concepts	Percent Containing Block	Percent Worked as Intended	Percent Containing Block	Percent Worked as Intended
Data	93.55%	100.00%	90.32%	100.00%
Conditionals	38.71%	91.67%	41.94%	92.31%
Radio	12.90%	100.00%	16.13%	100.00%
Loops	70.97%	86.36%	67.74%	85.71%
Operators	67.74%	100.00%	64.52%	100.00%
Criterion 2: Code documentation and organization				
Criteria	Percent Containing Block		Percent Containing Block	
Naming variables	93.55%		90.32%	

No unnecessary statement	93.55%	90.32%

Criterion 3: Designing for usability

Criteria	Percent Containing Block	Percent Containing Block
Functionality	87.10%	90.32%

Attitudes Towards Computing. The ANOVA Test demonstrated that there were significant differences in the pre-test data between the two cohorts (e.g., computing confidence [$F(1, 64) = 34.487$, $p < .001$], computer enjoyment [$F(1, 64) = 66.420$, $p < .001$], motivation to succeed [$F(1, 64) = 6.418$, $p = 0.014$], and gender equality [$F(1, 64) = 41.079$, $p < .001$]). The students' attitudes towards computing between the two cohorts were analyzed separately as pre-test data differed significantly between the two cohorts in many constructs. Table 3 indicates that there were no significant changes in students' attitudes toward computing in Cohort 1. One explanation for this is that the after-school program may not have any effect on students' attitudes at all. The possible second explanation is that there is a ceiling effect, as students started in the program with a positive attitude towards computing. However, Table 4 showed drastically different results. The attitudes of students in Cohort 2 changed significantly after the program in all constructs illustrating medium to large effect sizes. Since pre-test data indicate that their attitudes towards computing were significantly less positive compared to students' attitudes in Cohort 1, it is possible that there is a ceiling effect for students in Cohort 1. In other words, students in Cohort 1 started the program with positive attitudes towards computing in most constructs (e.g., computer enjoyment), leaving little room for growth.

Table 3. Student Attitudes Toward Computing – Cohort 1 ($N = 36$)

Constructs	Pre Mean	Post Mean	Mean Difference	Paired t-test	Effect Size (Cohen's D)
Computing Confidence	3.792	3.824	0.032	.591	0.034
Computer Enjoyment	4.274	4.198	-0.0754	.182	-0.120
Computer Importance	4.310	4.361	0.051	.459	0.081
Motivation to Succeed	3.981	3.931	-0.051	.547	-0.054
Identity and belongingness	3.778	3.815	0.037	.579	0.041
Gender Equality	4.277	4.277	0	1	0
Intention to Persist	3.375	3.417	0.042	.652	0.037

*$p < .05$.**$p < .01$. (where 1 = least positive attitudes toward computing and 5 = most positive attitudes toward computing)

Table 4. Student Attitudes Toward Computing – Cohort 2 ($N = 30$)

Constructs	Pre Mean	Post Mean	Mean Difference	Paired t-test	Effect Size (Cohen's D)
Computing Confidence	2.422	4.300	1.877	<.001**	1.355
Computer Enjoyment	2.357	4.500	2.142	<.001**	1.496
Computer Importance	3.688	4.461	0.772	<.001**	0.736
Motivation to Succeed	3.411	4.172	0.761	<.001**	0.668
Identity and belongingness	3.355	4.177	0.822	.001**	0.616
Gender Equality	2.333	4.658	2.325	<.001**	1.464
Intention to Persist	3.100	3.916	0.816	<.001**	0.785

*$p < .05$.**$p < .01$. (where 1 = least positive attitudes toward computing and 5 = most positive attitudes toward computing)

Discussion

This study provides insight into the design of short-term after-school computing programs aimed at helping students acquire an understanding of CS concepts and develop positive attitudes toward computing. The results indicate that the undergraduates were able to make connections between their CS knowledge and pedagogy practices to provide instruction to middle school students in the afterschool program. The study indicates that partnerships between universities and K-12 schools are very important for teaching CS concepts (Cuny, 2012; Mouza et al.,

2021). Through the support of faculty members and middle school teachers, high-quality CS content, high-quality pedagogical practices, and CS identity development could be implemented in after-school CS programs, even the ones that are short-term, to help broaden student participation in computing.

Regarding students' learning, overall, students performed well in developing a project by using the CS concepts that they had learned in the program. However, data indicate that the concept used with the least accuracy is "Loops" although the accuracy rate is still quite high (86.36% for Cohort 1 and 85.71% for Cohort 2). Perhaps, more time and effort should be dedicated to learning this concept as it can be the one that students often struggle with (Grover & Basu, 2017). Moreover, there was only a small number of students employing the "Radio" concept in their projects although this was used correctly in those projects. It is possible that many students were not comfortable or confident with using the concept, or they just simply did not prefer using the concept for their projects as it may require multiple devices to showcase the functionality of the code.

Further, findings suggest that the program had no statistically significant impact on students' attitudes towards CS in Cohort 1. Since students in Cohort 1 started the program with positive attitudes towards CS, it is probable there was a ceiling effect for students in Cohort 1. Moreover, as students in Cohort 2 developed significantly positive attitudes towards CS, it is likely that the program, indeed, could produce a significant effect on students' attitudes. The effect was also more significant compared to a prior study on a similar after-school program that used Scratch, another block-based programming environment (Mouza et al., 2016). One possible reason can be the students' strong positive attitudes at the beginning of that program compared to students' attitudes in Cohort 2 of this study. It is also possible that students' positive attitudes decrease as a program progresses due to its longer duration (12 weeks in the prior study instead of 6 weeks in the current study). Another explanation could be the presence of the physical computing device – Micro:bit, as the device has been shown to be a great motivator, and its tangibility helps develop students' interest and understanding (Voštinár & Knežník, 2020).

Conclusion

This work demonstrates the potential of establishing after-school CS programs through partnerships between K-12 schools and universities. In the study, with the support of university faculty, middle school educators, and CS undergraduates, the program was able to provide high-quality CS content and pedagogy. At the end of the program, most students were able to employ CS concepts in their final projects accurately. The program also had a significant impact on attitudes toward computing among students who started with less positive attitudes. Future work should incorporate a control group in the study to improve the reliability and validity of the results. Moreover, it can be beneficial to investigate the potentially different impacts of screen-based computing and physical computing on student learning of CS concepts. Finally, it may be worthwhile to examine the use of the "Radio" concept in students' projects due to its lack of utilization in both cohorts. This study is significant because it examines various approaches for supporting K-12 CS teachers and for increasing the participation in computing. It provides valuable insights into the design and implementation of effective after-school physical computing programs and can inform the development of future CS education initiatives in K-12 settings.

Acknowledgements

This work was funded by grants from the National Science Foundation: Award #1639649 and #1923483.

References

Ball, T., Protzenko, J., Bishop, J., Moskal, M., De Halleux, J., Braun, M., Hodges, S., & Riley, C. (2016). Microsoft touch develop and the bbc micro: bit. *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, 637–640.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads*, *2*(1), 48–54.

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology*, *13*(1), 20–29.

Ben-Ari, M. (1998). Constructivism in computer science education. *Acm Sigcse Bulletin*, *30*(1), 257–261.

Black, P., & Wiliam, D. (2009). Developing the theory of formative assessment. *Educational Assessment, Evaluation and Accountability*, *21*(1), 5–31. https://doi.org/10.1007/s11092-008-9068-5

Boaler, J. (2008). Promoting 'relational equity' and high mathematics achievement through an innovative mixed-ability approach.

British Educational Research Journal, *34*(2), 167–194. https://doi.org/10.1080/01411920701532145
Braught, G., Wahls, T., & Eby, L. M. (2011). The case for pair programming in the computer science classroom. *ACM Transactions on Computing Education (TOCE)*, *11*(1), 1–21.
Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada*, *1*, 25.
Brickhouse, N. W., Lowery, P., & Schultz, K. (2000). What kind of a girl does science? The construction of school science identities. *Journal of Research in Science Teaching: The Official Journal of the National Association for Research in Science Teaching*, *37*(5), 441–458.
Cápay, M., & Klimová, N. (2019). Engage your students via physical computing! *2019 IEEE Global Engineering Education Conference (EDUCON)*, 1216–1223.
Ching, Y.-H., Hsu, Y.-C., & Baldwin, S. (2018). Developing computational thinking with educational technologies for young learners. *TechTrends*, *62*(6), 563–573.
Cobb, P., & Hodge, L. L. (2002). A relational perspective on issues of cultural diversity and equity as they play out in the mathematics classroom. *Mathematical Thinking and Learning*, *4*(2–3), 249–284.
Cohoon, J., & Aspray, W. (2008). *Lost in translation: Gender and high school computer science*. Mit Press.
College Board. (2017). *CS Principles*. https://apcentral.collegeboard.org/courses/ap-computer-science-principles/course
Cuny, J. (2012). Transforming high school computing: A call to action. *ACM Inroads*, *3*(2), 32–36.
Darling-Hammond, L., Flook, L., Cook-Harvey, C., Barron, B., & Osher, D. (2019). Implications for educational practice of the science of learning and development. *Applied Developmental Science*, *24*(2), 97–140. https://doi.org/10.1080/10888691.2018.1537791
Denner, J., Werner, L., Campe, S., & Ortiz, E. (2014). Pair programming: Under what conditions is it advantageous for middle school students? *Journal of Research on Technology in Education*, *46*(3), 277–296. https://doi.org/10.1080/15391523.2014.888272
Devine, J., Finney, J., de Halleux, P., Moskal, M., Ball, T., & Hodges, S. (2019). MakeCode and CODAL: Intuitive and efficient embedded systems programming for education. *Journal of Systems Architecture*, *98*, 468–483.
Diéguez, J. L. R. (1988). Curriculum de José Luis Rodríguez Diéguez. *Enseñanza & Teaching: Revista Interuniversitaria de Didáctica*, *23*, 223–240.
Ericson, B., & McKlin, T. (2012). Effective and sustainable computing summer camps. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, 289–294.
Gardeli, A., & Vosinakis, S. (2017). Creating the computer player: An engaging and collaborative approach to introduce computational thinking by combining 'unplugged'activities with visual programming. *Italian Journal of Educational Technology*, *25*(2), 36–50.
Gretter, S., & Yadav, A. (2016). Computational thinking and media & information literacy: An integrated approach to teaching twenty-first century skills. *TechTrends*, *60*(5), 510–516.
Grover, S, Sedgwick, V., & Powers, K. (2020). Feedback through formative check-ins. In S Grover (Ed.), *Computer Science in K-12: An A to Z Handbook on Teaching Programming.* Edfinity.
Grover, S., & Basu, S. (2017). Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and boolean logic. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 267–272.
Hamer, J., Sheard, J., Purchase, H., & Luxton-Reilly, A. (2012). Contributing student pedagogy. *Computer Science Education*, *22*(4), 315–318. https://doi.org/10.1080/08993408.2012.727709
Heintz, F., Mannila, L., & Färnqvist, T. (2016). A review of models for introducing computational thinking, computer science and computing in K-12 education. *2016 IEEE Frontiers in Education Conference (FIE)*, 1–9. https://doi.org/10.1109/FIE.2016.7757410
Hudesman, J., Crosby, S., Flugman, B., Issac, S., Everson, H., & Clay, D. B. (2013). Using formative assessment and metacognition to improve student achievement. *Journal of Developmental Education*, *37*(1), 2.
Hui, T. H., & Umar, I. N. (2011). Does a combination of metaphor and pairing activity help programming performance of students with different self-regulated learning level?. *Turkish Online Journal of Educational Technology-TOJET*, *10*(4), 121–129.
Jiang, S., & Wong, G. K. W. (2017). Assessing primary school students' intrinsic motivation of computational thinking. *2017 IEEE 6th International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, 469–474.
Kafai, Y. B., Lee, E., Searle, K., Fields, D., Kaplan, E., & 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 (TOCE)*, *14*(1), 1–20.
Kalelioglu, F., & Sentance, S. (2020). Teaching with physical computing in school: The case of the micro: bit. *Education and Information Technologies*, *25*(4), 2577–2603.
Ladson-Billings, G. (1995). But that's just good teaching! The case for culturally relevant pedagogy. *Theory Into Practice*, *34*(3), 159–165.
Lewis, C. M., Shah, N., & Falkner, K. (2019). Equity and diversity. In S. A. Fincher & A. V. Robins (Eds.), *The Cambridge Handbook of Computing Education Research* (pp. 481–510). Cambridge University Press. https://doi.org/10.1017/9781108654555.017
McDowell, C., Werner, L., Bullock, H. E., & Fernald, J. (2003). The impact of pair programming on student performance,

perception and persistence. *25th International Conference on Software Engineering, 2003. Proceedings.*, 602–607.
McGee, S., McGee-Tekula, R., Duck, J., McGee, C., Dettori, L., Greenberg, R. I., Snow, E., Rutstein, D., Reed, D., & Wilkerson, B. (2018). Equal outcomes 4 all: A study of student learning in ECS. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 50–55.
Mendes, E., Al-Fakhri, L., & Luxton-Reilly, A. (2006). A replicated experiment of pair-programming in a 2nd-year software development and design computer science course. *ACM SIGCSE Bulletin*, *38*(3), 108–112.
Mouza, C., Marzocchi, A., Pan, Y.-C., & Pollock, L. (2016). Development, implementation, and outcomes of an equitable computer science after-school program: Findings from middle-school students. *Journal of Research on Technology in Education*, *48*(2), 84–104.
Mouza, C., Sheridan, S., Lavigne, N. C., & Pollock, L. (2021). Preparing undergraduate students to support K-12 computer science teaching through school-university partnerships: Reflections from the field. *Computer Science Education*, 1–26.
Nasir, N. S., & Cooks, J. (2009). Becoming a hurdler: How learning settings afford identities. *Anthropology & Education Quarterly*, *40*(1), 41–61.
Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism*, *36*(2), 1–11.
Piaget, J. (1972). *Psychology and epistemology: Towards a theory of knowledge* (Vol. 105). Penguin Books Ltd.
Pollock, L., Mouza, C., Atlas, J., & Harvey, T. (2015). Field experiences in teaching computer science: Course organization and reflections. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 374–379.
Schunk, D. H. (2016). *Learning theories an educational perspective* (seventh ed). Pearson.
Sentance, S., & Schwiderski-Grosche, S. (2012). Challenge and creativity: using. NET gadgeteer in schools. *Proceedings of the 7th Workshop in Primary and Secondary Computing Education*, 90–100.
Sentance, S., Waite, J., Hodges, S., MacLeod, E., & Yeomans, L. (2017). "Creating Cool Stuff" Pupils' Experience of the BBC Micro: bit. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 531–536.
Sentance, S., Waite, J., Yeomans, L., & MacLeod, E. (2017). Teaching with physical computing devices: the BBC micro: bit initiative. *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*, 87–96.
Shah, N., Lewis, C. M., Caires, R., Khan, N., Qureshi, A., Ehsanipour, D., & Gupta, N. (2013). Building equitable computer science classrooms: Elements of a teaching approach. *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, 263–268.
Shahin, M., Gonsalvez, C., Whittle, J., Chen, C., Li, L., & Xia, X. (2022). How secondary school girls perceive Computational Thinking practices through collaborative programming with the micro: bit. *Journal of Systems and Software*, *183*, 111107.
Smith, M. (2016, January 30). *Computer Science For All*. Office of the Press Secretary, The White House.
Sorva, J., Karavirta, V., & Malmi, L. (2013). A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education (TOCE)*, *13*(4), 1–64.
Tang, M., & Cook, E. P. (2001). Understanding relationship and career concerns of middle school girls. In *Educating young adolescent girls* (pp. 223–240). Routledge.
United States Bureau of Labor Statistics. (2022). *Computer and Information Technology Occupations*. https://www.bls.gov/ooh/computer-and-information-technology/home.htm#:~:text=Employment in computer and information,add about 667%2C600 new jobs.
Videnovik, M., Zdravevski, E., Lameski, P., & Trajkovik, V. (2018). The BBC micro:bit in the international classroom: Learning experiences and first Impressions. *2018 17th International Conference on Information Technology Based Higher Education and Training (ITHET)*, 1–5.
Voštinár, P., & Knežník, J. (2020). Education with BBC micro:bit. *International Journal of Online and Biomedical Engineering (IJOE)*, *16*(14). https://doi.org/10.3991/ijoe.v16i14.17071
Vygotsky, L. S., & Cole, M. (1978). *Mind in society: Development of higher psychological processes*. Harvard university press.
Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35.

Appendix A. Overview of the After-School Program

Week	Lesson Content	Week	Lesson Content
Week 1	Input & Variables - Identifying students' level of CS experience - CS Unplugged: Brainstorming ways to interact with an imaginary friend - Micro:bit: Create a pet hamster	Week 4	Loops - CS Unplugged: over and over instruction - Micro:bit Programming: Shake the Micro:bit to activate the countdown
Week 2	Conditionals - CS Unplugged: If/then notecard game - Micro:bit: Rock, Paper, Scissors Game	Week 5	While-loop - CS Unplugged: Musical Chair - Micro:bit Programming: Create Hot Potato Game
Week 3	Radio - CS Unplugged: Guess the frequency number - Micro:bit: Send radio wave messages	Week 6	Review - Connecting Coding to everyday lives - Micro:bit: Students get to develop their own code