Micro:bit vs. Python: Students' Perceptions and Attitudes Toward Computing

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: There has been a growing interest in teaching computer science (CS) concepts to students at a younger age. Increasingly, block-based programming has been used in place of traditional text-based programming languages, like Python, in K-12 education. However, little empirical research has been conducted to compare the combination of the former and physical computing with the latter. This study aimed to address this gap by comparing the attitudes and perceptions of elementary school students in the two approaches in a six-week afterschool program. The findings from the experiment indicated that students' attitudes and perceptions toward computing were more positive when using physical computing. These findings suggest potential pedagogical implications and future research directions.

Keywords: physical computing, computational thinking, text-based programming

Introduction

Drawing on computer science (CS) concepts, computational thinking (CT) skills are essential for managing various aspects of our daily lives and navigating the world around us (Wing, 2006). Early involvement in computing can significantly promote students' future participation and diversity in CS (Ching et al., 2018; Margolis, 2010; Yardi & Bruckman, 2007). Block-based programming has been introduced as a method to engage young learners in CS (Bau et al., 2017). Its combination with physical computing has shown promising results in attitudes toward computing and learning gains (Veng et al., 2023; Kastner-Hauler et al., 2022). However, the comparison of this combination with traditional text-based programming remains relatively unexplored. This study compared these approaches with two groups of elementary school students; one group used Python, a text-based programming language, while the other group utilized block-based physical computing with Micro:bit . Micro:bit is a device with various functions that can be programmed via MakeCode's block-based language, aligned with CS concepts (e.g., data, variables) (Brennan & Resnick, 2012; Voštinár & Knežník, 2020). This comparison aimed to answer the following questions:

1. How do students' experiences and perceptions of using block-based physical computing differ from those using text-based programming?
2. How do the two approaches differ regarding students' attitudes toward the field of CS?

Literature Review

Early Engagement in Computational Thinking and Micro:bit

Papert introduced the concept of computational thinking (CT) in 1980, suggesting that interactions with technology could help develop new ways of thinking. Later, Wing (2006) coined the term "computational thinking" to specifically describe thinking skills that involve applying concepts fundamental to computer science (CS). CT is considered a vital 21st-century skill for students to navigate in today's technology-driven world (Voogt et al., 2015; Zhang & Nouri, 2019). Therefore, it is important to engage students in CT at a younger age to allow sufficient time for the skills to develop before tertiary education (Barr & Stephenson, 2011; Czerkawski & Lyman, 2015). One way to engage students in CT is through physical computing. One of the common physical computing devices is Micro:bit, a pocket-sized programmable device that has a built-in display, buttons, motion detection, light sensing, and Bluetooth (Ball et al., 2016). Incorporating Micro:bit in CS education can help with students' motivation, creativity, and learning gains (Cliburn, 2006; Sentance et al., 2017). Students reported feeling satisfied and enjoying learning with the devices (Vlahu-Gjorgievska et al., 2018) and appreciated their ease of use and usefulness for learning CS concepts and coding (Sentance et al., 2017).

Block-Based and Text-Based Programming

Developing CT skills can be difficult for novice programmers due to the struggle with abstract concepts, solution organization, and concrete programming concepts (e.g., debugging) (Lahtinen et al., 2005). The majority of errors made by novice programmers during coding include semantic (e.g., comparing the wrong variables) and syntax errors (e.g., missing quotes for strings) (Altadmri & Brown, 2015). Therefore, block-based programming is commonly introduced to overcome these challenges (Bau et al., 2017). Several studies have compared the affective outcomes (e.g., confidence) of block-based and text-based programming (e.g., Price & Barnes, 2015; Weintrop, & Wilensky, 2017), but there have been mixed results. For example, Price and Barnes (2015) found that there was no significant difference in learners' attitudes towards programming (e.g., perceived difficulty), whereas a different study showed that block-based programming improved such attitudes (Saito et al., 2017). Moreover, another comparative study indicated that students in block-based programming groups showed a higher level of interest in computing in the future, but there was no difference in their confidence or enjoyment (Weintrop & Wilensky, 2017). However, none of the studies incorporated physical computing. With the positive reported experience and outcomes from physical computing (Kastner-Hauler et al., 2022; Sentance et al., 2017), our study aimed to compare students' experience and attitudes when using text-based programming and the combination of block-based programming and physical computing with Micro:bit.

Theoretical Framework

We situated our work within the concepts of computational thinking, constructionism, and structuration. Computational thinking (CT) refers to "the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (Cuny et al., 2010, p. 1). It draws on CS concepts, which encompass various cognitive processes, such as reasoning at different levels of abstraction and problem-solving skills (Wing, 2006). There have been different approaches to teaching CT (e.g., Denning, 2017; Folk et al., 2015), one of which draws on constructivism where students learn by actively constructing their knowledge (Ben-Ari, 1998). Constructionism, built on constructivism, posits that learning happens by creating tangible outputs. (Papert & Harel, 1991). Physical computing, drawing from constructionism, focuses on creating tangible and programmable objects. Prior studies highlighted its benefits in enhancing students' motivation, and creativity (Sentance et al., 2017). Moreover, receiving tangible and immediate feedback may benefit students' engagement and motivation (Scherer et al., 2020). Structuration reflects the dynamic relationship between knowledge tools and understanding (Wilensky & Papert, 2010). Studying these evolving systems helps us understand their influence on students' learning experiences and attitudes. Therefore, based on the aforementioned frameworks, we aimed to explore students' attitudes and perceptions when using two different systems: block-based physical computing with Micro:bit and text-based coding with Python.

Methods

Context and Participants

The study was conducted as part of an after-school program, a joint effort between a university and an elementary school. Students, consisting of fourth and fifth graders, were randomly assigned into two groups: Group A: a block-based physical computing group (*n* =30), and Group B: a text-based programming group (*n* = 32). Two CS undergraduates, enrolled in a college course on engaging youth in computing (Mouza et al., 2016, 2021; Pollock et al., 2015), served as instructors. CS undergraduates met with faculty and teachers to plan lessons and discuss strategies to foster computing participation. The six-week program consisted of 90-minute sessions covering various CS concepts (see Table 1), with instructors rotating between groups biweekly. Each session consisted of (1) an unplugged activity, in which CS concepts were taught through an interactive activity without using a computer, and (2) a programming activity where Group A programmed their Micro:bits to solve a problem and Group B developed screen-based programs. The program is designed for students to develop positive self-perception in computing through exposure to diverse instructors and parents in the CS field and by linking learning content to students' identities, such as asking them to design their projects in the final session (Shah et al., 2013).

Table 1. Overview of the After-School Program's Content

Week	Content	CS Unplugged	Programming	
			Group A (Micro:bit)	Group B (Python)
1	Input and Variables	Brainstorming ways to interact with an imaginary friend	Create a pet hamster	Create a pet turtle
2	Conditionals	If/then notecard game	Rock, Paper, Scissors Game	Rock, Paper, Scissor Game
3	Radio	Guess the frequency number	Send radio wave messages	Create program to send a message to your friends
4	Loops	Over and over instruction	Shake the Micro:bit to activate the countdown	Create a countdown timer
5	While-loop	Musical Chair	Create the Hot Potato Game	Create a number guessing game
6	Review	Connecting Coding to everyday lives	Students get to develop their own code	

Data Collection

Data were collected through four main sources:

(a) **Observations of After-School Program:** Five sessions were observed, documenting covered CS concepts, teaching techniques, interactions, and materials used.

(b) **Pre/Post surveys for elementary school students**: A pre/post survey in a Likert scale format developed by Ericson and McKlin (2012) was used to examine changes in students' attitudes toward computing (e.g., computing confidence; computer enjoyment)

(c) **A focus group interview with elementary school students**: A semi-structured interview was conducted with six students to understand their experiences and perceptions of each approach.

Data Analysis

(a) **Quantitative Data:** Data on students' computing attitudes were analyzed using R Studio. Paired t-tests were used to measure students' attitude changes after the program and compare the pre-test results of the two groups, which showed that there were no statistically significant differences between the pre-test results of the two groups. ANCOVA was conducted to compare attitudes between block-based physical computing and text-based computing groups, controlling for pre-test data and gender.

(b) **Qualitative Data:** The interview transcript was coded using content analysis, suitable for phenomena with limited literature or knowledge (Hsieh & Shannon, 2005). Generated codes were organized into main and subcodes as shown in Table 2 below. Data from in-class observations (*n* = 5) with students from both classes were analyzed for triangulation purposes.

Table 2. Coding Scheme

Codes	Subcodes
Grammatical Complexity	Syntax error; Frustration
Tangibility	Physical device feedback; Sharing results with others
Assistance and Support	Debugging support from instructors; Peer mentoring; Feeling of competence and confidence
Sense of Belonging	Opportunity for exploration and creativity

Results

RQ1: How do students' experiences and perceptions of using block-based physical computing differ from those using text-based programming in the program?

Findings indicate that there were differences in students' experiences and perceptions in the two classes.

Grammatical Complexity

Students found Python coding challenging due to syntax errors, which hindered their creative work. One student shared their Python experience, "I'm still confused about it. The grammar is really hard to get right …. It is really hard when I really want to make something." Observations revealed that students in Python class encountered more errors, asked for help frequently, and often expressed frustration.

Tangibility

Students in the Micro:bit class enjoyed the immediate results and tangibility of their projects. They felt satisfaction in sharing their work with others. One student said, "Microbit, instead of ….. on screen, it could also be in the Micro:bit, it was really fun to play …. and show it to friends and adults."
Another student also said, "Like you put an LED to show up on the Micro:bit….like for the magic 8 ball. It makes me feel kind of good."

Observations confirmed that Micro:bit students frequently showcased their projects, while Python students, despite occasionally sharing their codes with peers, typically spent more time on their computers.

Assistance and Support

Students in both classes valued instructor support for debugging codes. Python class students felt guilty asking for help repeatedly due to the lack of other resources. One student said, "[The instructors] told you where the bug is... But sometimes, I feel bad when I ask them a lot... There is no video for us to watch." Students from both groups also sought help from peers. One student said, "There were people around me who knew more so sometimes I asked them for help."

Sense of Belonging

Python class students questioned their competence and future success, despite recognizing Python's practicality. One student said, "Even if you mess up one little thing in Python, it won't work, because they really want you to get all the grammar correct. I probably need more time, but I don't know if I am going to be good….". These students also felt they lacked enough time to explore the codes and complete their projects due to Python's complexity. One student shared, “So I never really got to finish it and I never got time to either. [Be]cause like it's hard on Python."

RQ2: How do the two approaches differ regarding students’ attitudes toward the field of CS?

The pair t-test results indicated that students’ attitudes towards computing significantly improved in both groups, except for identity and belongingness and intention to persist for those in Group B (see Tables 3 and 4). When comparing the two groups by controlling for pre-test data and gender, the ANCOVA result suggested that block-based physical computing has a more statistically significant effect on all constructs of students’ attitudes, except gender equality and intention to persist (see Table 5). The result on gender equality was expected as programming environments should not have any effect on such students’ attitudes. Interestingly, the result also indicated that female students showed a higher intention to persist (*p*<.05) in studying CS than males, after accounting for class types and pre-test scores.

Table 3
Student Attitudes Toward Computing – Group A (Micro:bit) (*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<.001, ** p<.01, *p<.05. (where 1 = least positive attitudes toward computing and 5 = most positive attitudes toward computing)

Table 4
Student Attitudes Toward Computing – Group B (Python) (*n* =32)

Constructs	Pre Mean	Post Mean	Mean Difference	Paired t-test	Effect Size (Cohen’s D)
Computing Confidence	2.515	3.312	0.797	.004**	0.558
Computer Enjoyment	2.888	3.911	1.022	.001**	0.673
Computer Importance	3.536	3.91	0.406	.64	0.34
Motivation to Succeed	3.1	3.636	0.536	.013*	0.467
Identity and belongingness	3.083	3.646	0.563	.35	0.389
Gender Equality	2.046	4.273	2.23	.001**	1.276
Intention to Persist	2.883	3.328	0.445	.55	0.353

***p<.001, ** p<.01, *p<.05. (where 1 = least positive attitudes toward computing and 5 = most positive attitudes toward computing)

Table 5
Comparing Student Attitudes Toward Computing between Python and Micro:bit Group

Constructs	Unstandardized B	Standardized Coefficient Beta	Sig.	Effect Size r^2	Gender Sig.
Computing Confidence	0.994	0.246	<.001***	0.189	.949
Computing Enjoyment	0.571	0.167	.001**	0.156	.899
Computer Importance	0.476	0.155	.003 **	0.148	.374
Motivation to Succeed	0.4674	0.208	.028 *	0.131	.943
Identity and Belongingness	0.507	0.234	.035 *	0.049	.940
Gender Equality	0.373	0.210	.082	0.028	.514

Intention to Persist	0.359	0.225	.117	0.258	.003**

****p*<.001, ** *p*<.01, **p*<.05. (where 1 = least positive attitudes toward computing and 5 = most positive attitudes toward computing)

Discussion

Our study contributes to the growing research on teaching methods and programming environments in CS education. Unlike previous studies focusing solely on block-based or text-based programming (e.g., Grover & Basu, 2017; Weintrop, 2019) or the comparison between the two (e.g., Price & Barnes, 2015; Weintrop & Wilensky, 2017), our work incorporated physical computing in the comparison and in-depth analysis of how these different environments may influence students' attitudes and perceptions toward computing. Our findings show that students have more positive perceptions of block-based physical computing than those of text-based programming due to the likelihood of errors and tangibility. Moreover, the complexity of text-based programming may play a negative role in students' sense of belonging and intention to persist in the CS field. In contrast, block-based programming has a higher impact on students' attitudes toward computing, except for gender equality and intention to persist. Similar pre-test attitudes in both groups suggest ceiling effect is unlikely for the Python group. The findings are inconsistent with prior works comparing text-based programming and block-based programming as the studies did not show a significant change in computing confidence and enjoyment (Weintrop & Wilensky, 2017) or no difference at all (Price & Barnes, 2015). The addition of physical computing, appreciated for its tangibility by students or unplugged activities, could be the factor causing these differences, given their promising results in prior studies (Gardeli & Vosinakis, 2017; Jiang & Wong, 2017). The findings can help guide decisions about elementary school CS education programming environments. Future studies could explore larger samples over longer periods to understand students' attitudes and the impact of introducing physical computing in text-based environments.

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

References

Altadmri, A., & Brown, N. C. (2015, February). 37 million compilations: Investigating novice programming mistakes in large-scale student data. *Proceedings of the 46th ACM Technical Symposium On Computer Science Education* (pp. 522-527). https://doi.org/10.1145/2676723.2677258

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? *Inroads, 2*(1), 48–54.

Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable programming. *Communications of the ACM*, *60*(6), 72–80. https://doi.org/10.1145/3015455

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

Ben-Ari, M. (1998). Constructivism in computer science education. *ACM SIGCSE Bulletin*, *30*(1), 257–261. https://doi.org/10.1145/274790.274308

Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking.* [Conference presentation]. 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada.

Ching, Y.-H., Hsu, Y.-C., & Baldwin, S. (2018). Developing computational thinking with educational technologies for young learners. *TechTrends*, *62*(6), 563–573. https://doi.org/10.1007/s11528-018-0292-7

Cliburn, D. (2006). Experiences with the LEGO Mindstorms throughout the undergraduate computer science curriculum. *Proceedings Frontiers in Education 36th Annual Conference*. https://doi.org/10.1109/fie.2006.322315

Cuny, J., Snyder, L., & Wing, J.M. (2010). *Demystifying computational thinking for non-computer scientists*. Unpublished manuscript in progress, referenced in http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Czerkawski, B. C., & Lyman, E. W. (2015). Exploring issues about computational thinking in higher education. *TechTrends, 59*(2), 57–65. https//doi.org/10.1007/s11528-015-0840-3

Denning, P. (2017). Computational thinking in science. *American Scientist*, *105*(1), 13. https://doi.org/10.1511/2017.124.13

Ericson, B., & McKlin, T. (2012). Effective and sustainable computing summer camps. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, 289–294. https://doi.org/10.1145/2157136.2157223

Folk, R., Lee, G. R., Michalenko, A. C., Peel, A., & Pontelli, E. (2015, October 21). GK-12 DISSECT: Incorporating computational thinking with K-12 science without computer access. *Frontiers in Education Conference*. https://doi.org/10.1109/fie.2015.7344238

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. https://doi.org/10.17471/2499-4324/910

Grover, S., & Basu, S. (2017, March 8). Measuring student learning in introductory block-based programming. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. https://doi.org/10.1145/3017680.3017723

Hsieh, H. F., & Shannon, S. E. (2005). Three Approaches to Qualitative Content Analysis. *Qualitative Health Research*, *15*(9), 1277–1288. Sage. https://doi.org/10.1177/1049732305276687

Jiang, S., & Gary W.K. Wong. (2017, December 1). Assessing primary school students' intrinsic motivation of computational thinking. *2017 IEEE 6th International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*. https://doi.org/10.1109/tale.2017.8252381

Kastner-Hauler, O., Tengler, K., Sabitzer, B., & Lavicza, Z. (2022). Combined effects of block-based programming and physical computing on primary students' computational thinking skills. *Frontiers in Psychology*, *13*. https://doi.org/10.3389/fpsyg.2022.875382

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, *37*(3), 14-18.

Margolis, J. (2010). *Stuck in the shallow end: Education, race, and computing*. MIT Press.

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. https://doi.org/10.1080/15391523.2016.1146561

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. https://doi.org/10.1080/08993408.2021.1970435

Papert, S. A. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic books.

Papert, S., & Harel, I. (1991). Situating constructionism. In I. Harel (Eds.), *Constructionist learning*. Cambridge, MA: MIT Media Laboratory.

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. https://doi.org/10.1145/2676723.2677286

Price, T. W., & Barnes, T. (2015, August 9). Comparing textual and block interfaces in a novice programming environment. *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*. https://doi.org/10.1145/2787622.2787712

Saito, D., Washizaki, H., & Fukazawa, Y. (2017). Comparison of text-based and visual-based programming input methods for first-time learners. *Journal of Information Technology Education: Research*, *16*(1), 209-226.

Scherer, R., Siddiq, F., & Viveros, B. S. (2020). A meta-analysis of teaching and learning computer programming: Effective instructional approaches and conditions. *Computers in Human Behavior, 109*. https://doi.org/10.1016/j.chb.2020.106349

Sentance, S., Waite, J., Hodges, S., MacLeod, E., & Yeomans, L. (2017, March 8). "Creating cool stuff": Pupils' experience of the BBC Micro:bit. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. https://doi.org/10.1145/3017680.3017749

Shah, N., Lewis, C. M., Caires, R., Khan, N., Qureshi, A., Ehsanipour, D., & Gupta, N. (2013). Building equitable computer science classrooms. *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, 263–268. https://doi.org/10.1145/2445196.2445276

Veng, S., Mouza, C. & Pollock, L. (2023). Examining the Design and Outcomes of an After-School Physical Computing Program in Middle-School. In E. Langran, P. Christensen & J. Sanson (Eds.), *Proceedings of*

Society for Information Technology & Teacher Education International Conference (pp. 2316-2326). New Orleans, LA, United States: Association for the Advancement of Computing in Education (AACE).

Vlahu-Gjorgievska, E., Videnovik, M., & Trajkovik, V. (2018). Computational thinking and coding subject in primary schools: Methodological approach based on alternative cooperative and individual learning cycles. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)* (pp. 77-83). IEEE.

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, *20*, 715-728. https://doi.org/http://dx.doi.org/10.1007/s10639-015-9412-6

Voštinár, P., & Knežník, J. (2020). Education with BBC Micro:bit. *International Journal of Online and Biomedical Engineering (IJOE)*, *16*(14), 81. https://doi.org/10.3991/ijoe.v16i14.17071

Weintrop, D., & Wilensky, U. (2015, June 21). To block or not to block, that is the question. *Proceedings of the 14th International Conference on Interaction Design and Children*. https://doi.org/10.1145/2771839.2771860

Weintrop, D., & Wilensky, U. (2017). Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education*, *18*(1), 1–25. https://doi.org/10.1145/3089799

Weintrop, D. (2019). Block-based programming in computer science education. *Communications of the ACM*, *62*(8), 22-25. https://doi.org/10.1145/3341221

Wilensky, U., & Papert, S. (2010). Restructurations: Reformulating knowledge disciplines through new representational forms. *Proceedings of the Constructionism 2010 Conference*.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35.

Yardi, S., & Bruckman, A. (2007). What is computing? *Proceedings of the Third International Workshop on Computing Education Research - ICER '07*. https://doi.org/10.1145/1288580.1288586

Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education*, *141*, 103607. https://doi.org/10.1016/j.compedu.2019.103607