Programming Smart Objects: How Young Learners' Programming Skills, Attitudes, and Perception Are Influenced

Mazyar Seraj
Eindhoven University of Technology
Eindhoven, The Netherlands
m.seraj@tue.nl

Mauricio Verano Merino
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
m.verano.merino@vu.nl

Ebrahim Rahimi
The Open University of the Netherlands
Heerlen, The Netherlands
ebrahim.rahimi@ou.nl

Lina Ochoa Venegas
Eindhoven University of Technology
Eindhoven, The Netherlands
l.m.ochoa.venegas@tue.nl

Abstract

Programming literacy is crucial for current and future generations of young learners, irrespective of their career paths. Programming education is thus essential, making teaching methods and tools to be tailored to the target audience. In this context, contemporary visual programming environments, particularly block-based programming, have become instrumental in introducing programming concepts to young learners. Educational theories such as Constructionism advocate an approach centered on the learner to deepen and motivate learning. In computer science, these theories can be applied by providing hands-on experiences that connect computer science to real-life situations through the manipulation or construction of physical and tangible computational devices. This study explores the impact of creating a smart object for a smart home using block-based programming on young learners' attitudes and perceptions toward programming and their programming skills acquisition. An introductory programming workshop involved 28 8^{th} grade students from a secondary school constructing and programming a smart-lighting object in a smart home setting. Performance, attitude, and perception trajectories were assessed through repeated questionnaires. Our results indicate that constructing and programming a real-life smart object enhances learners' confidence and programming skills. This paper contributes to programming education literature by demonstrating the potential of block-based programming, specifically in the context of state-of-the-art smart technologies, to foster programming skills and develop positive attitudes and perceptions among learners.

CCS Concepts: • **Applied computing** → **Interactive learning environments**; *Computer-assisted instruction.*

Keywords: block-based programming, young learners, smart home, programming education, internet of things

ACM Reference Format:
Mazyar Seraj, Mauricio Verano Merino, Ebrahim Rahimi, and Lina Ochoa Venegas. 2024. Programming Smart Objects: How Young Learners' Programming Skills, Attitudes, and Perception Are Influenced. In *Proceedings of the 2024 ACM SIGPLAN International Symposium on SPLASH-E (SPLASH-E '24), October 24, 2024, Pasadena, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3689493.3689982

1 Introduction

The proliferation of computing applications has led individuals without a programming background to perform programming tasks as part of their work duties. In the United States, over 12 million individuals claim to perform some programming at work, and nearly 50 million rely on databases and spreadsheets [27]. Seeking ways to help individuals develop programming skills from a young age while keeping a positive attitude toward their learning process is essential. Nowadays, and aiming at the aforementioned goal, young learners often begin their programming learning journey with visual programming environments (e.g., block-based environments), designed to reduce syntactical errors and enable program authoring without extensive programming knowledge [23, 42, 47, 48]. These environments are prevalent in introductory programming courses [49, 51] and workshops [23, 42]. Additionally, block-based programming takes a step further by offering opportunities for learners to implement ideas into tangible objects and real environments, fostering hands-on experience [21, 23, 30, 43].

SPLASH-E '24, October 24, 2024, Pasadena, CA, USA

ACM ISBN 979-8-4007-1216-6/24/10
https://doi.org/10.1145/3689493.3689982

Constructionism advocates learning through constructing objects or artifacts [28]. In particular, it emphasizes learner-centred education by encouraging learners to understand abstract concepts through designing personal and meaningful artifacts [3, 13, 15, 26]. This approach in programming education has led to the widespread use of tangible objects and block-based programming (e.g., Scratch[1]) for young learners and inexperienced users [21, 23, 34, 35]. Previous research highlights the effectiveness of using block-based programming to introduce programming to young learners in various contexts, including mobile robots [21, 23, 30], computational textiles [13, 15, 34], and smart homes [14, 43].

We argue that smart homes can have an impact on young learners by incorporating objects they interact with daily, potentially leading to more meaningful experiences. However, the current state-of-the-art shows a limited exploration of utilizing block-based programming in educational environments to program tangible objects in the context of smart homes. Moreover, there is limited research on how programming such objects impact young learners' performance, attitudes, and perceptions of programming, which is key to facilitating a sustainable and comprehensive programming education [32, 44]. This paper addresses this gap by investigating how block-based programming in real-life smart homes influences young learners' programming skills, attitudes, and perceptions towards programming.

To evaluate the impact of constructing smart objects in smart home contexts using block-based programming on young learners, we conducted a 2-day non-formal programming workshop for 28 8^{th} grade students (ages 12–14). The workshop aimed to allow students to construct and program smart-lighting objects for integration into a smart home environment. Using a Block-Based Programming Environment (BBPE) based on Google Blockly [8], the goal was to enhance young learners' programming skills, attitudes, and perception toward programming by connecting the smart-lighting objects with real-life smart homes. The block-based programming application simplified programming complexity, helping learners grasp fundamental concepts and author programs. Learners used the programming application to connect a micro-controller to the smart home server and read data from various sources to construct the *smart-lighting object*. The path of learners' attitudes and perception toward programming based on repeated quantitative and open-ended qualitative questionnaires (based on Weintrop and Wilensky's study [51]) was examined at the programming workshop's beginning, middle, and end. Furthermore, we assessed their programming performance at the workshop's beginning and end using two computational thinking tests (based on Lewis' study [19]). Based on our results, we identified that using a real-life smart home positively impacts learners' programming performance and potentially their confidence toward programming tasks. However, current results do not provide clear conclusions about their interest, enjoyment, and perception of programming.

The remainder of the paper is structured as follows. Section 2 introduces our conceptual background and related work revolving around block-based programming and the use of tangible smart objects in programming education. In Section 3 and Section 4, we describe the study's research setting and its results, respectively. We wrap up the paper by presenting a discussion of the main results of the study in Section 5, and it conclusions in Section 6.

2 Background and Related Work

This section provides an overview of BBPEs and educational literature on tangible and smart objects and environments.

2.1 Block-based Programming Environments

Visual block-based programming has been used to enable inexperienced users and young learners to learn and author programs with little training [5, 11, 12]. In recent years, numerous BBPEs have been introduced for young learners to program on-screen animations [23, 35], mobile-based applications [38], micro-controllers [20, 31, 46], and other programmable tangible objects [23, 25]. These environments reduce the complexity of programming for learners by using visual blocks to generate code syntax. This approach reduces syntax errors and eases the manipulation of code structures, and, therefore, they are widely used to teach programming, in particular, to young learners [1, 2, 16, 17, 33, 50].

2.2 Tangible Smart Objects in Programming Education

One important feature in learning programming, especially for young learners, is to enable them to understand how programming and computer science are relevant to their daily life [6, 23, 44]. Rooted in Constructionism [28], programming tangible objects has a long history in education [4, 36], and meanwhile, countless programmable kits and computational textiles (e.g., wearable devices) are on the market and have entered into educational institutions [6, 23, 39, 52]. When following a constructionist approach, young learners learn by designing and constructing interactive and tangible objects that are personally meaningful to them [23, 37]. Thus, researchers and educators create introductory programming environments to support the acquisition of programming skills through designing and constructing tangible objects.

In the Computer Science Education (CSE) research community, several studies tried to investigate various forms of smart devices such as robots [21, 24, 26, 30], smart wearable devices [7, 9, 15, 34], and smart homes [14, 40] to motivate young learners and show them how modern technologies relate to their daily life. When it comes to constructing smart

[1] https://scratch.mit.edu

objects in smart home contexts—the focus of this article—previous studies have either investigated smart home topics that might attract female developers into the software industry [15, 43], or; develop block-based programming approaches (e.g., application, training session) to enable the programming of smart homes without assessing their impact on young learner's attitude and perception [42]. Thus, relatively little attention has been devoted to the potential impact of programming tangible objects in smart home contexts using BBPE on young learners' performance, attitude, and perception toward programming.

2.2.1 Mobile robot programming. Robots are one of the most common tangible and smart devices used for educational purposes. The literature reports that mobile robots are effectively used as a tool to generate positive interest and improve learning among young learners [21, 30, 32]. Paramasivam et al. [30] explored the use of mobile robots as a tool for suited reflection in elementary school programming courses. Martinez et al. [21] enabled preschool and elementary school learners to program and control the behavior of Arduino boards in the context of N6 robots. Furthermore, Przybylla and Romeike [32] presented creative learning environments for young students, using programmable kits (e.g., LEGO Mindstorms as a robotic toolkit) to offer a hands-on experience that can be used to develop a constructionist computer science curriculum with physical computing. The results show that learners were highly engaged in robot programming, and researchers successfully established confidence among the learners that robot programming is interesting. However, no information was provided on whether this approach enabled them to have a higher intention of learning programming.

2.2.2 Smart home programming. Another approach is to use smart homes and living labs to motivate young learners in learning programming and foster their interest in computer science. The main goal of using smart homes is to provide exposure to hands-on programming experiences in programming training sessions [10, 14, 43]. Thus, young learners can both participate in and experience new technologies which are adapted to technical equipment, as well as learn basic programming concepts in the context of smart homes. Katterfeldt and Dittert [14] reported on three co-design workshops where young female learners created ideas related to smart homes, and implemented them on a doll house. Still, a key challenge with the use of real-life smart homes is that young learners can only perform limited actions in the environment. For instance, Seraj et al. [42] used real-life smart homes as a medium to teach basic programming skills to young learners, where they were able to produce and deploy pieces of code that can be applied to smart homes. However, the range of activities was limited because they focused only on block-based programming to control the smart home itself, using written documents with work examples and instructional procedures. They did not involve students in constructing and programming tangible objects to integrate into the smart home. Hence, further research is warranted in this domain, particularly exploring how students can utilize block-based programming to construct and program smart objects suitable for integration into smart home systems.

Regardless of the specific approach, the open challenge lies in supporting young learners to grasp basic programming concepts and cultivate a positive attitude toward programming. Visual BBPEs, known for simplifying programming complexity for young learners [41, 48, 51], are insufficient on their own to connect programming with its impact on daily life. While smart environments demonstrate the relevance of programming to daily needs, they may not be fully accessible for young learners to create applications.

3 Research Setting

This study experimentally explores the following Research Question (RQ): *How does the construction of a smart device in the context of a smart home, using a block-based programming environment influence young learners' programming skills, attitudes, and perception toward programming?* To answer our RQ, we conducted a 2-day non-formal programming workshop, incorporating two computational thinking tests and three questionnaires. The workshop was extra-curricular and independent of their regular curriculum[2]. We evaluate learners' performance using Lewis' approach [19], focusing on basic computational concepts: variables, loops, and control-flow statements (conditions and logical operators). Computational thinking tests were conducted at the beginning (PreCTT) and end (PostCTT) of the workshop. Additionally, we conduct three questionnaires to assess young learners' attitudes and perceptions toward programming. Administered at the beginning (PreQ), middle (IntermediateQ), and end (PostQ) of the workshop, the questionnaires utilized a 5-point Likert Scale and open-ended questions. In accordance with Weintrop and Wilensky's study [51], learners' attitudes in all questionnaires were measured in terms of confidence, interest, and enjoyment in future programming learning opportunities.

3.1 Overview of the Smart Home

The smart home considered in this study has 60 m^2 and serves as an automated living lab, featuring smart elements like a height-adjustable sink, wardrobe suggesting outfits, voice recognition, and smart mirror and fridge. Designed for trial living, it caters to the elderly and those with physical or cognitive impairments, equipped with actuators in charge of translating electrical signals into physical events (e.g., doors

[2]Students did not receive any grades based on their performance during the workshop.

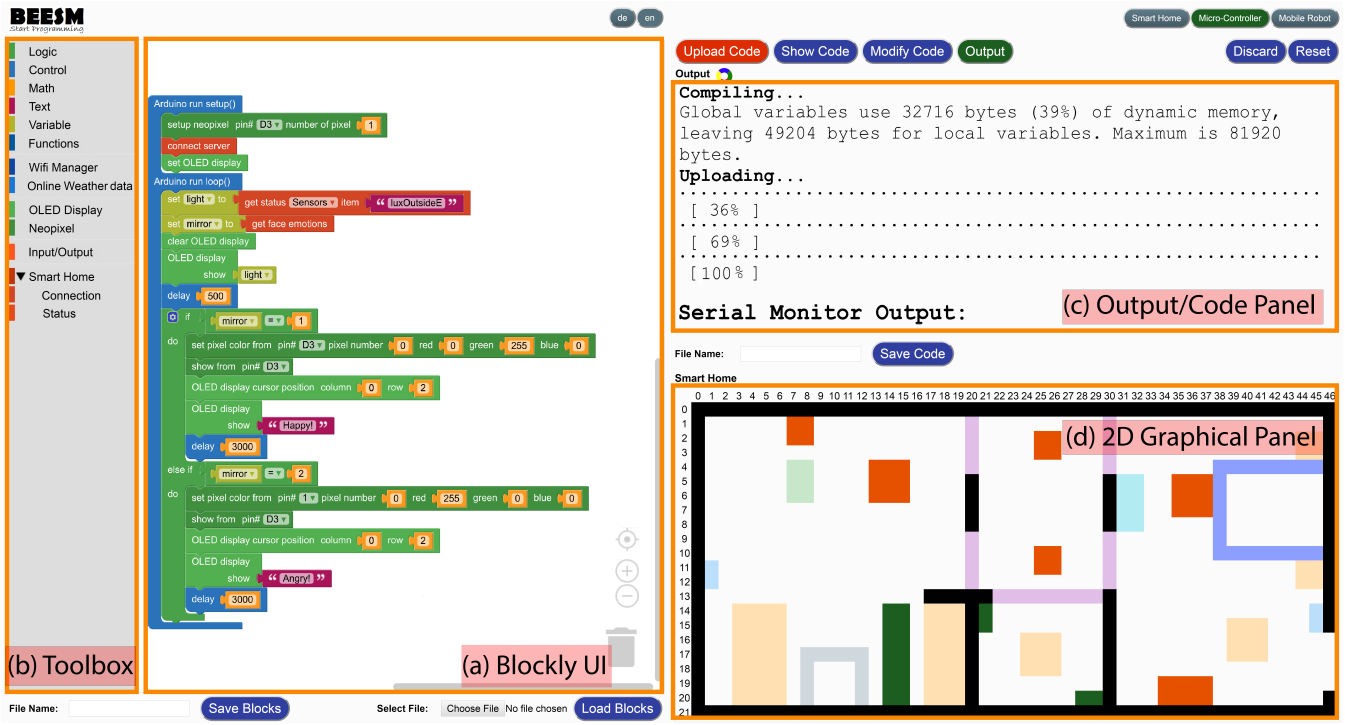


Figure 1. Screenshot of the programming application interface.

and lights) and sensors responsible for translating physical events into electrical signals (e.g., lighting and temperature). Remote control is facilitated via a RESTful HTTP interface of the smart environment. In our programming workshop, students connect to the smart home's server using a WeMos D1 mini board. Afterward, students can construct a *smart-lighting object* by reading generated data and programming *lights* and an *OLED display* to react to such data. We choose WeMos D1 mini board for its compact Wi-Fi functionalities enabling easy prototyping, and OLED display for showing numerical and string values.

3.2 Block-based Programming Application

In this study, we utilize BEESM [41, 45] to generate Arduino code for programming the WeMos D1 mini board connected to our smart home. The resulting code is generated by combining blocks, which are at the top level of the program. When the learner runs the program, the generated code is directly uploaded into the micro-controller. To enhance the user interface for our target students, we modify BEESM to have a full vision of four panels (see Figure 1): the *Block Panel* with a block workspace and categories (see Figure 1a and Figure 1b), the *Code Panel* displaying the generated code (see Figure 1c), the *Output Panel* for program output, errors, and the compile/upload process (see Figure 1c), and the *2D Graphical Panel* presenting a view of the smart home's status (see Figure 1d).

3.3 Study Design and Data Collection Strategy

The PreQ comprises eight 5-point Likert scale questions (Q1–Q8), two "*yes-no*" questions (Q9 and Q10), and two open-ended questions (Q11 and Q12). It records learners' attitudes toward programming, incorporating Likert Scale questions from Weintrop and Wilensky's study [51] (considered Q1 to Q6) with additional questions relevant to our research. Learners' prior experience with block-based programming and micro-controllers is captured through the "*yes-no*" questions. Two open-ended questions inquire about learners' intentions for the programming workshop and their perception of computer programming (see Table 1).

The IntermediateQ consists of (1) a 5-point Likert scale question (Q1) measuring learners' perception of using a tangible object and making it smart; (2) eight Likert scale questions (Q2–Q9) measuring attitudes toward programming; and (3) two open-ended questions (Q10–Q11) for learners' workshop feedback. The attitudinal questions mirror those in the PreQ, with slight wording changes in two questions: "*do you think you will be successful in this workshop?*" to "*do you think you were successful in this workshop?*", and "*would you like to learn how to program?*" to "*would you like to learn more about programming?*" (see Table 1).

The PostQ includes all IntermediateQ questions and two inquiries about learners' gender and age. Notably, the question "*do you think it is useful if you program a real object? (e.g., the LEDs light up)*" is changed to "*do you think it would*

Table 1. Questionnaires

Pre Questionnaire (PreQ)		Intermediate Questionnaire (IntermediateQ)	
#	Question	#	Question
Q1	Do you think you are good at programming?	Q1	* Do you think it is useful to program a real object? (e.g., the LEDs light up)
Q2	Do you think you will be successful in this workshop?	Q2	Do you think you are good at programming?
Q3	Do you think programming is fun?	Q3	Do you think you were successful in this workshop?
Q4	Do you like programming?	Q4	Do you think programming is fun?
Q5	Are you excited about this workshop?	Q5	Do you like programming?
Q6	Do you think programming is difficult?	Q6	Are you excited about this workshop?
Q7	Are you interested in programming?	Q7	Do you think programming is difficult?
Q8	Would you like to learn how to program?	Q8	Are you interested in programming?
Q9	Have you ever worked with a micro-controller (e.g. Arduino)?	Q9	Would you like to learn more about programming?
Q10	Have you ever worked with a block-based programming environment?	Q10	* What did you particularly like about the first workshop day?
Q11	I find programming ...	Q11	* What did you particularly dislike about the first workshop day?
Q12	What do you think of this workshop?		

be useful if you programmed a real smart object? (e.g., perform actions based on sensor information)". Additionally, the open-ended questions "*what did you particularly like/dislike about the first workshop day?*" are changed to "*what did you particularly like/dislike about the workshop?*" in the PostQ (see * in Table 1).

We assess learners' performance in basic computational concepts through PreCTT and PostCTT, each comprising a computational thinking exam with two gap-filling and seven open-answer questions.[3] Each question carries one point, except Q6, which had two points for mentioning both LED colors, making the maximum score for each test 10 points. Learner responses are independently evaluated by two researchers for consistency. Q1 and Q6 test understanding of "*variables*," Q2, Q3, and Q9 examine "*control-flow statements*," while Q4, Q5, Q7, and Q8 focused on "*loops*." PreCTT and PostCTT are slightly different. The variations in PreCTT and PostCTT ensure careful consideration of block programs and accurate responses. The open-ended responses are subjected to analysis by two researchers independently, utilizing an open-coding technique. Each researcher identifies and categorizes themes within the responses. Following an initial round of coding, the researchers engage in a collaborative discussion to reconcile any discrepancies and reach an agreement on the final coding labels. This iterative process ensures the validity and reliability of the coding framework employed in the analysis.

3.4 Participants

A total of 28 8^{th} grade students (22 boys, 6 girls; ages 12–14) from a secondary school participated in our 2-day non-formal programming workshop. During the workshop, we focused on learning to program using BEESM for smart objects in the context of smart homes. Prior to the workshop, we communicated with the school teacher and headmaster, outlining the project's objectives and seeking consent for student participation. Participation was voluntary, with students selected based on interest, teacher's suggestions, and parental consent from those enrolled in programming-related courses. The teacher facilitated communication and brought interested students to the workshop. All data collection and analysis were conducted anonymously and confidentially.

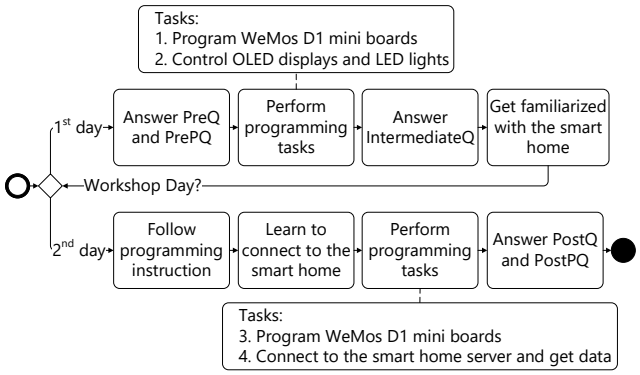


Figure 2. Procedure of the programming workshop.

The school, teacher, and parents were informed about study protocols, with parental consent obtained for students' participation. German Research Center for Artificial intelligence (DFKI) provided all necessary equipment, including computers and components, as well as the smart home. All participating students had prior experience with a BBPE (Scratch) as part of their curriculum. Six students (all girls) indicated that they worked with a micro-controller (Arduino board) in another one-day extra-curricular workshop, while the remaining 22 students (all boys) did not have prior experience with it. Notably, neither Scratch nor Arduino is employed in our workshop.

3.5 Procedure

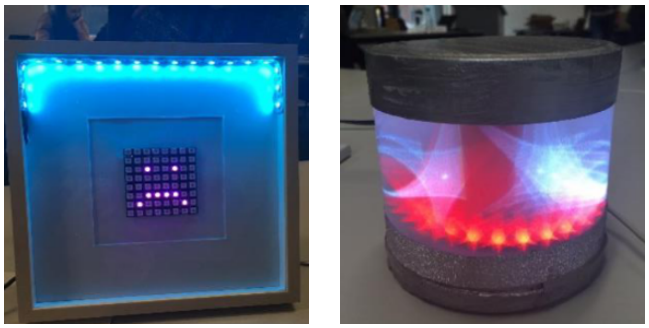
The workshop was conducted by an instructor directly hired by the research center funding the study. Each daily session of the six-hour workshop included a 90-minute break. The participants were allowed to make their self-made teams of size 2 or 3. As a result, 10 teams were formed by the students themselves. It is noteworthy that all female students opted to team up together, while male students grouped with each other. Students worked collaboratively on the programming tasks within their formed teams. All students individually completed PreQ, IntermediateQ, PostQ, PreCTT, and PostCTT. Oral explanations and prepared slides were employed daily, complemented by supplementary documents detailing components and programming blocks. These materials served as aids, reducing instructor effects.[4] The description of the topics and activities covered during the workshop are as follows (see Figure 2):

[3] See supplementary material at https://figshare.com/s/375671e99b54c2008a7d

[4] We make our prepared slides and other supplementary documents available at https://github.com/projekt-smile/Smartes-Stimmungslicht

First Day: Each student began the session by completing the PreQ and PreCTT assessments. Following this, they received an overview of block-based programming, which they would utilize to accomplish a series of programming tasks. These tasks guided them through the creation of a *smart-lighting object* using data generated by the smart home system. Next, students were introduced to the block-based programming application. Each group was supplied with basic components including WeMos boards, LED lights, OLED displays, cables, and mini breadboards, along with concise instructions for connecting and assembling them. In the initial programming task, students learned how to program the WeMos board to manage an OLED display and display alphanumeric values in various cursor positions. Following this, for the second programming task, they were requested to program the WeMos board to control LED lights using blocks linked to the RGB coloring model, enabling them to change the lights' colors. Students were asked to explore corresponding blocks in the programming application and then to fill in the IntermediateQ. The session concluded with an introduction to the smart home environment, where all objects such as lights, doors, and sensors, along with their respective functionalities, were explained to help students identify different smart items and understand their functionalities in the smart home.

Second Day: At the beginning of the second day, each student group was tasked with presenting and sharing the workings of WeMos boards, OLED displays, and LED lights with their peers and other groups. Additionally, a review of various smart items within the smart home environment was provided. In addition to the RGB coloring model, using WeMos boards and OLED displays, the basic computational concepts introduced in this session were variables, loops, and control-flow statements. For the third programming task, students acquired the skills to develop programs for WeMos boards, facilitating their connection to the smart home server, data retrieval, and variable assignment. Subsequently, as part of the fourth programming task, students started programming the WeMos board to control the behavior of OLED displays and LED lights in response to the data generated by the smart home. Following this, students formed small groups of 2–3 members and brainstormed ideas, envisioning how they could design artifacts with LED lights and displays to be embedded in the smart home ecosystem. They were required to think of how the LED lights and displays in their object should communicate and react to the data generated by different items in the smart home. Furthermore, students designed the layout of the artifacts for their *smart-lighting object* and implemented their project functionality utilizing the programming concepts in WeMos boards. The smart-lighting objects and the number of groups that made each object are: (i) communicating with the smart mirror–four groups, (ii) communicating with the light and temperature sensors–3 groups, (iii) reacting to the status of lights and doors–2 groups, and (iv) reacting to the status of TV volume–1 group. Figure 3a shows an example of the communication with the smart mirror, where learners program the microcontroller to display different colors and information on the *smart-lighting object* based on facial expressions detected by the mirror. Also, Figure 3b demonstrates the constructed *smart-lighting object* reacting to the status of lighting sensors. By the conclusion of the second day, students completed the PostQ and PostCTT assessments. The workshop wrapped up with a presentation detailing the characteristics and functionalities associated with each lighting object.



(a) Interacting with smart mirror **(b)** Reacting to the status of sensors

Figure 3. Samples of constructed "smart-lighting object" by the students.

4 Results

This section is divided into three parts addressing young learners' programming skills, attitudes, and perceptions. First, results from an analysis of the students' responses to the computational thinking tests, PreCTT and PostCTT, are presented in Section 4.1. Second, results from an analysis of the PreQ, IntermediateQ, and PostQ questionnaires are presented in Section 4.2, looking at the students' confidence, enjoyment, and interest in future programming learning opportunities. Finally, in Section 4.3, students' perception of constructing and seeing the impact of their programs on a tangible smart object is reported. Central tendency and variability statistics of all results are presented and summarized in Table 2 (performance) and Table 3 (attitudes and perception). In particular, we introduce minimum and maximum values, as well as the first quartile (Q1), median (Q2), third quartile (Q3), mean, and standard deviation (SD).

4.1 Programming Skills

In Table 2, we present the results obtained by the participants in PreCTT and PostCTT. Both the mean and median (M) reveal an increase in the overall performance of the group, however, a paired-sample t-test (assuming normality) is required to verify the significant difference between the distributions. To ensure the data adheres to normality

Table 2. Participants' programming performance based on PreCTT and PostCTT (Q: Quartile and SD: Standard Deviation)

	Min	Q1	Q2	Q3	Max	Mean	SD
PreCTT	0.00	2.00	3.50	5.00	8.00	3.36	2.22
PostCTT	0.00	2.75	5.00	6.00	9.00	4.43	2.64

assumptions and that the pair differences exhibit an approximately normal distribution, we conducted the Shapiro-Wilk test considering the residuals between the two distributions and a significance level $alpha = 0.05$. The results of the test ($W = 0.97 \mid p = 0.56$) suggest that data distributions of the PostCTT and PreCTT results behave like a normal distribution given that $p > alpha$ ($0.56 > 0.05$). We now conduct a paired-sampled t-test with a significance level $alpha = 0.05$. The results ($t(28) = 2.69 \mid p = 0.012$) show that students performed significantly better in PostCTT compared to the PreCTT as $p < alpha$ ($0.012 < 0.05$).

4.2 Attitudes Toward Programming

This section looks at the students' attitude in terms of confidence (Section 4.2.1), interest (Section 4.2.2), and enjoyment (Section 4.2.3) based on the results obtained from the PreQ, IntermediateQ, and PostQ questionnaires. The results are presented in Table 3. In particular, we aim to identify if there is a significant difference in the students' attitudes at different moments of the workshop. The following analysis relies on the Friedman test, a non-parametric test well-suited for small sample sizes involving more than two distributions (three questionnaires), to assess potential significant differences among them. We consider a significance level $alpha = 0.05$. In case the Friedman test indicates a significant difference among the distributions, we perform a post-hoc Nemenyi test to determine which specific distributions differ from each other. We also consider the Benjamini-Hochberg correction to control for False Discovery Rate (FDR) when dealing with multiple tests by adjusting p-values (denoted p_{hb}).

4.2.1 Confidence. With respect to students' confidence, we measured their responses to the three questions in PreQ (Q1, Q2, and Q6), IntermediateQ (Q2, Q3, and Q7), and PostQ (Q2, Q3, and Q7). Concerning the question of whether they think that they are *Good at programming*, the Friedman test indicates a significant difference among the distributions given that $p < alpha$ with $p = 0.014$. We then perform the Nemenyi test between distribution pairs. However, for all three comparisons p-values were greater than the chosen alpha—that is, $p = 0.306$ ($p_{bh} = 0.459$) between PreQ and IntermediateQ, $p = 0.082$ between PreQ and PostQ ($p_{bh} = 0.247$), and $p = 0.761$ between IntermediateQ and PostQ ($p_{bh} = 0.761$). With respect to the question of whether they think that they will be/were *Successful in the workshop*, in general, students indicated a medium level of confidence in all three questionnaires ($M = 4$). The Friedman test indicates no significant difference between the datasets ($p = 0.241$). Concerning the *Programming is difficult* questions, the students experienced a medium level of difficulty in programming throughout the workshop ($M = 3$). The Friedman test indicates no significant difference among the distributions ($p = 0.113$).

Table 3. Participants' attitude and perception based on PreQ, IntermediateQ, and PostQ (Q: Quartile and SD: Standard Deviation)

Question	Questionnaire	Min	Q1	Q2	Q3	Max	Mean	SD
Confidence								
Good at programming	PreQ	1.00	3.00	3.00	3.25	5.00	3.11	0.90
	IntermediateQ	1.00	3.00	4.00	4.00	5.00	3.43	0.90
	PostQ	1.00	3.00	4.00	4.00	5.00	3.54	0.73
Successful in the workshop	PreQ	2.00	3.75	4.00	4.00	5.00	3.75	0.69
	IntermediateQ	2.00	3.75	4.00	5.00	5.00	4.07	0.92
	PostQ	3.00	4.00	4.00	5.00	5.00	4.07	0.70
Programming is difficult	PreQ	2.00	2.00	3.00	4.00	5.00	3.22	1.07
	IntermediateQ	1.00	2.00	3.00	4.00	5.00	2.93	1.21
	PostQ	1.00	2.50	3.00	4.00	5.00	3.26	1.14
Interest								
Interested in Programming	PreQ	2.00	4.00	5.00	5.00	5.00	4.43	0.78
	IntermediateQ	2.00	4.00	5.00	5.00	5.00	4.36	0.85
	PostQ	2.00	4.00	5.00	5.00	5.00	4.32	0.93
Learn how to program	PreQ	3.00	4.00	5.00	5.00	5.00	4.43	0.68
	IntermediateQ	2.00	4.00	4.00	5.00	5.00	4.25	0.87
	PostQ	1.00	3.75	5.00	5.00	5.00	4.18	1.14
Enjoyment								
Programming is fun	PreQ	3.00	4.00	5.00	5.00	5.00	4.61	0.62
	IntermediateQ	3.00	4.00	5.00	5.00	5.00	4.64	0.61
	PostQ	3.00	4.00	5.00	5.00	5.00	4.57	0.68
Like programming	PreQ	3.00	4.00	5.00	5.00	5.00	4.68	0.54
	IntermediateQ	2.00	4.00	5.00	5.00	5.00	4.46	0.86
	PostQ	3.00	4.00	5.00	5.00	5.00	4.54	0.73
Excited about the workshop	PreQ	3.00	3.00	4.00	4.50	5.00	3.89	0.79
	IntermediateQ	1.00	3.50	4.00	5.00	5.00	3.89	1.03
	PostQ	2.00	4.00	4.00	5.00	5.00	4.15	0.80
Perception								
Programming a Smart Object	IntermediateQ	1.00	4.00	5.00	5.00	5.00	4.30	1.05
	PostQ	2.00	4.00	4.00	5.00	5.00	4.26	0.80

4.2.2 Interest. To measure students' interest in programming, they responded to the two questions in PreQ (Q7 and Q8), IntermediateQ (Q8 and Q9), and PostQ (Q8 and Q9). Most students showed a high tendency ($M \geq 4$) toward programming and learning how to program from the beginning until the end of the workshop for all questionnaires. The Friedman test shows no significant result among the distributions throughout the workshop, $p = 0.657$ for *Interested in programming* and $p = 0.404$ for *Learn how to program.*

In the open-ended question "*I find programming...*" in PreQ, 26 students associated programming with a positive attitude, except one who called it "*boring*", and one did not give any answer. Most of students (16) found it "*very interesting*", "*interesting*", or "*fascinating*"; others responded: "*cool*", "*great*", "*good*", etc. Four students additionally reasoned that programming allows them to be creative; for example "*good*

and interesting, because you can let your creativity run freely". This question was not asked again in IntermediateQ and PostQ and it was replaced by the two questions regarding the programming workshop. This introduces a shift in the questionnaire structure and prompts further exploration into the evolving attitudes toward programming over time.

4.2.3 Enjoyment. With respect to students' enjoyment, we measured their responses to the three questions in PreQ (Q3–Q5), IntermediateQ (Q4–Q6), and PostQ (Q4–Q6). In all questionnaires, students indicated a high level of enjoyment for programming in terms of *Programming is fun*, and *Like programming* ($M = 5$). The Friedman test shows no significant statistical differences among the distributions throughout the workshop, $p = 0.761$ for *Programming is fun* and $p = 0.815$ for *Like programming*. When the students were asked whether they are/were *Excited about the workshop*, they reported a middle level of excitement for all questionnaires ($M < 4$). The Friedman test indicated no significant result was obtained among the groups ($p = 0.410$).

Students were also required to respond to the open-ended question "*what do you think of this workshop?*" in PreQ. In IntermediateQ and PostQ, students were asked what they like about the first day and about the workshop in general, respectively. In PreQ, 10 students did not answer the question or wrote that they do not know it. Among the remaining 18 students, three mentioned that they "*appreciate the workshop offer*", and three were "*excited*" or "*interested*" about the workshop. Other 12 students addressed different thoughts; for instance, "*I think it is good*", "*it sounds interesting to me*", "*the importance of programming for their future*", or "*having insights into smart homes*".

Students' expectations toward the end of the workshop appeared positive. 25 students answered the question in IntermediateQ. Working with "*LED lights*" or "*displays*" was mentioned by eight students, and "*programming*" was addressed by seven of them. The scope of action (e.g., opportunities to be creative) was mentioned by three; other answers (7) were "*fun*", "*everything*", or "*the instruction and explanation*". In PostQ, five students mentioned the "*scope of freedom in working*", five "*programming*", seven "*constructing or decorating the tangible objects*", and four addressed the "*smart home*". Other students (5) wrote other statements, such as the "*the programming application*" (2), "*explanation*" (2), or "*everything*" (1); two students did not answer.

In IntermediateQ and PostQ, students were asked what they did not like about the first day and about the workshop in general, respectively. In IntermediateQ, six students complained about the "*length*" of the workshop. They also pinpointed there was too much "*repetition*", resulting in a "*boring*" setup. Among the rest of the students, six complained about "*too much explanation*", and that they would like to "*experiment more*". In PostQ, six students mostly complained about the "*missing variety of tasks*"; for example, they indicated that it is better to divide the programming tasks, or provide more tasks. Likewise, two students still were not happy with "*too much explanation*" and two complained about "*programming*".

In summary, the opportunity to work on personally meaningful projects within the smart home, as well as programming, did not alter the high level of enjoyment among the students. However, via the intermediate and post-questionnaires, we identify points of improvement when it comes to the workshop length, the instruction style (e.g., time for explanation versus practice), and the variety of programming tasks and their corresponding difficulty level.

4.3 Perception Toward Programming

Concerning programming a tangible object (Q1 in IntermediateQ and PostQ), most students expressed a high level of usefulness ($M = 5$ in IntermediateQ and $M = 4$ in PostQ). No significant difference occurred among the distributions. In terms of these scores, most students indicated a high usefulness level for programming a tangible object in IntermediateQ and programming a real smart object in PostQ.

5 Discussion

In this study, young learners utilized block-based programming to grasp basic computational concepts and apply them to construct a personally meaningful *smart-lighting object* within a smart home. In the following sections, we discuss how this study sheds light on how this constructionist approach might influence young learner's programming performance, attitude, and perception, becoming a stepping stone to filling in the current research gap. Particularly, we delve into the impact of this experience on learners' acquisition of programming skills, attitudes, and perceptions toward programming. Further, the limitations of this study and future lines of research will be outlined.

5.1 Impact on Programming Skills

After comparing the differences between the pre-workshop and the post-workshop results, we identified a statistically significant difference in the students' scores. These results show potential for the use of the construction of smart objects in programming skills acquisition. This is further reinforced by the students' positive feedback towards the conclusion of the workshop. They expressed appreciation for the freedom to work on programming, delve into smart home concepts, and engage in constructing tangible smart objects.

Our main takeaway message is that experiencing programming in the realm of smart technologies can allow young learners to understand programming better, significantly improving students' programming skills. This study focuses on fundamental programming skills including variables and assignment statements, `for` and `while` loops, and control-flow statements like the `if` statement. Our results cannot

directly attribute the acquisition of such skills to programming real objects in smart homes or the construction of smart objects. Further research is thus required to identify the correlation between block-based programming in real-life smart homes and the young learner's performance, learning attitudes, and perception towards programming. This can be implemented by, for instance, introducing a control and experimental group and accounting for significant differences between them at different points during the young learners' learning path rather than focusing only on the students' improvement after the intervention (via the smart-objects programming workshop).

Arguably, programming real objects in smart homes can establish a collaborative learning environment where students co-design and create tangible objects, using their design and construction activities as a vehicle for learning [22]. This argument is in line with findings from other studies [16] and the premises of educational theories and models such as Constructionism [29] and learning by design [18]. These theories advocate the indispensable role of design and making activities in deepening students' conceptual understanding and providing a meaningful context for their learning.

5.2 Impact on Attitudes and Perception Toward Programming

When it comes to young learners' attitudes toward programming, our findings indicate a general *interest* and *enjoyment* in programming among students before, during, and after the workshop. Qualitative findings suggest that enjoyment among students stems from the creative construction of the *smart-lighting object* and the application of their newly acquired programming skills to program it. Conversely, when it comes to *confidence*, students perceived programming as somehow challenging throughout the workshop with a mild (non-significant) increase in confidence as the study progressed. These results align with their computational test performance, which demonstrated a significant improvement between the beginning and end of the workshop.

These attitudes are similar to the ones reported in [15, 34], where computational textiles construction activities can increase students' confidence and enjoyment in dealing with technology and working with electronics and programming. However, confidence decreased at the end of the second day compared to the first day of the workshop, coinciding with increased perceived difficulty in programming.

Further studies are thus required to assess the correlation between difficulty and students' attitudes—in terms of confidence, interest, and enjoyment—and perceptions. Moreover, there are no conclusive positive results when it comes to evaluating young learners' perceptions of programming when using these learning contexts. This might be related to the relatively short timeframe of the workshop (i.e., 2 days), making it difficult to observe any significant change in the attitude and perception of the students.

5.3 Limitations and Future Work

In this study, we identified five main limitations. Addressing these can outline future lines of research within this context. The first limitation is the gender disparity among participants, which was beyond the researchers' control due to the school teacher's recruitment. To mitigate this, we consider that researchers must be involved in the recruiting process in our future work, ensuring that efforts are made to target girls. Additionally, selecting a gender-balanced sample should be prioritized. A similar limitation is that all participants were from the same school and already interested in learning programming, potentially impacting the results related to attitudes and perception. Therefore, we seek to address it in our future work by offering rewards and incentives to students who are not necessarily interested upfront in programming. The third similar limitation concerns the small sample size (28 8^{th} grade students), which can limit the generalizability of our findings. While we tend to expand the scope of this work, having a larger sample size is our primary concern and should be addressed in future iterations.

The fourth limitation concerns the limited diversity of programming tasks and the workshop's duration. Moreover, due to high costs, the practicality of constructing smart objects in the classroom is limited to a few privileged educational contexts. Comparing the effects of using a smart home versus more affordable artifacts like robots or smart textiles in different educational contexts would also be an intriguing avenue for exploration. Lastly, the fifth limitation arises from the short interval between completing the pre-, intermediate, and post-questionnaires, all containing almost the same questions (e.g., identical titles and numbers). This proximity in time and similarity in questions may cause students to recall their responses from the pre- or intermediate questionnaires when completing the post-questionnaire, potentially biasing their answers. One strategy to mitigate this limitation is to reorder, reverse, or reformulate the same questions in these questionnaires. Additionally, the mid- and long-term effects of using BBPE to program smart objects on young learners' attitudes and perceptions will not be noticeable, resulting in a research gap to be investigated in the future.

6 Conclusions

Acquiring programming skills is becoming vital for young learners, emphasizing real-life programming applications. In this paper, we argue that this can be achieved by using block-based programming to implement smart objects in smart home environments, advocating a constructionist approach. We assess our hypothesis on a 2-day workshop involving 28 8^{th}-grade students, where we assess learners' performance, attitudes, and perception, pre- and post-integration of block-based programming in smart home contexts. Our results unveil insights into their grasp of computational concepts and attitudes toward programming. Particularly, these findings

suggest that smart homes enrich introductory programming, providing a meaningful application domain that positively impacts young learners' programming skills. The next step involves designing and conducting more research to clearly identify the influence of smart object construction on the programming learning trajectory of young learners. After that, researchers and educators can assess how these avenues can be applied in introductory programming courses.

Acknowledgments

This work was partially funded by the German Federal Ministry for Education and Research (BMBF) within the project SMILE under grant number 01FP1613. The authors would like to thank for this support.

References

[1] David Bau. 2015. Droplet, a blocks-based editor for text code. *Journal of Computing Sciences in Colleges* 30, 6 (2015), 138–144.

[2] David Bau, Jeff Gray, Caitlin Kelleher, Josh Sheldon, and Franklyn Turbak. 2017. Learnable programming: blocks and beyond. *Commun. ACM* 60, 6 (2017), 72–80.

[3] Fabiane Barreto Vavassori Benitti. 2012. Exploring the educational potential of robotics in schools: A systematic review. *Computers & Education* 58, 3 (2012), 978–988.

[4] Paulo Blikstein et al. 2015. Computationally enhanced toolkits for children: historical review and a framework for future design. *Foundations and Trends® in Human–Computer Interaction* 9, 1 (2015), 1–68.

[5] Brian Broll, Akos Lédeczi, Peter Volgyesi, Janos Sallai, Miklos Maroti, Alexia Carrillo, Stephanie L Weeden-Wright, Chris Vanags, Joshua D Swartz, and Melvin Lu. 2017. A visual programming environment for learning distributed programming. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. 81–86.

[6] Leah Buechley, Mike Eisenberg, Jaime Catchen, and Ali Crockett. 2008. The LilyPad Arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 423–432.

[7] Leah Buechley, Nwanua Elumeze, and Michael Eisenberg. 2006. Electronic/computational textiles and children's crafts. In *Proceedings of the 2006 conference on Interaction design and children*. 49–56.

[8] Neil Fraser. 2014. Google blockly-a visual programming editor. Retrieved January 10, 2023 from https://developers.google.com/blockly/.

[9] Brittany Garcia, Sharon Lynn Chu, Beth Nam, and Colin Banigan. 2018. Wearables for learning: examining the smartwatch as a tool for situated science reflection. In *Proceedings of the 2018 CHI conference on human factors in computing systems*. 1–13.

[10] Mateus Carvalho Gonçalves, Otávio Neves Lara, Raphael Winckler de Bettio, and André Pimenta Freire. 2021. End-user development of smart home rules using block-based programming: a comparative usability evaluation with programmers and non-programmers. *Behaviour & Information Technology* 40, 10 (2021), 974–996.

[11] Justin Huang, Tessa Lau, and Maya Cakmak. 2016. Design and evaluation of a rapid programming system for service robots. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 295–302.

[12] Bas Jansen and Felienne Hermans. 2019. Xlblocks: a block-based formula editor for spreadsheet formulas. In *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 55–63.

[13] Yasmin B Kafai, Eunkyoung Lee, Kristin Searle, Deborah Fields, Eliot Kaplan, and Debora Lui. 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 (2014), 1–20.

[14] Eva-Sophie Katterfeldt and Nadine Dittert. 2018. Co-designing Smart Home Maker Workshops with Girls. In *Proceedings of the Conference on Creativity and Making in Education*. 100–101.

[15] Eva-Sophie Katterfeldt, Nadine Dittert, and Heidi Schelhowe. 2009. EduWear: smart textiles as ways of relating computing technology to everyday life. In *Proceedings of the 8th International Conference on Interaction Design and Children*. 9–17.

[16] Caitlin Kelleher and Randy Pausch. 2005. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)* 37, 2 (2005), 83–137.

[17] Andrew J Ko, Brad A Myers, and Htet Htet Aung. 2004. Six learning barriers in end-user programming systems. In *2004 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 199–206.

[18] Janet L Kolodner, David Crismond, Jackie Gray, Jennifer Holbrook, and Sadhana Puntambekar. 1998. Learning by design from theory to practice. In *Proceedings of the international conference of the learning sciences*, Vol. 98. Atlanta, GA, 16–22.

[19] Colleen M Lewis. 2010. How programming environment shapes perception, learning and goals: logo vs. scratch. In *Proceedings of the 41st ACM technical symposium on Computer science education*. 346–350.

[20] Makeblock. 2019. mBlock - The educational programming software. Retrieved January 17, 2023 from http://www.mblock.cc.

[21] Cecilia Martinez, Marcos J Gomez, and Luciana Benotti. 2015. A comparison of preschool and elementary school children learning computer science concepts through a multilanguage robot programming platform. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*. 159–164.

[22] Matthew M Mehalik, Yaron Doppelt, and Christian D Schuun. 2008. Middle-school science through design-based learning versus scripted inquiry: Better overall science concept learning and equity gap reduction. *Journal of engineering education* 97, 1 (2008), 71–85.

[23] Alexandros Merkouris, Konstantinos Chorianopoulos, and Achilles Kameas. 2017. Teaching programming in secondary education through embodied computing platforms: Robotics and wearables. *ACM Transactions on Computing Education (TOCE)* 17, 2 (2017), 1–22.

[24] Joseph E Michaelis and Bilge Mutlu. 2019. Supporting Interest in Science Learning with a Social Robot. In *Proceedings of the 18th ACM International Conference on Interaction Design and Children*. 71–82.

[25] Amon Millner and Edward Baafi. 2011. Modkit: blending and extending approachable platforms for creating computer programs and interactive objects. In *Proceedings of the 10th International Conference on Interaction Design and Children*. 250–253.

[26] Omar Mubin, Catherine J Stevens, Suleman Shahid, Abdullah Al Mahmud, and Jian-Jie Dong. 2013. A review of the applicability of robots in education. *Journal of Technology in Education and Learning* 1, 209-0015 (2013), 13.

[27] Brad A. Myers, Andrew J. Ko, and Margaret M. Burnett. 2006. Invited Research Overview: End-User Programming. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems* (Montréal, Québec, Canada) *(CHI EA '06)*. ACM, 75–80. https://doi.org/10.1145/1125451.1125472

[28] Seymour Papert. 1980. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, New York, NY.

[29] Seymour Papert and Idit Harel. 1991. Situating constructionism. *constructionism* 36, 2 (1991), 1–11.

[30] Vivek Paramasivam, Justin Huang, Sarah Elliott, and Maya Cakmak. 2017. Computer science outreach with end-user robot-programming tools. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. 447–452.

[31] Carlos Pereira Atencio. 2019. Ardublockly. Retrieved January 15, 2023 from https://ardublockly.embeddedlog.com.

[32] Mareen Przybylla and Ralf Romeike. 2014. Physical Computing and Its Scope–Towards a Constructionist Computer Science Curriculum with Physical Computing. *Informatics in Education* 13, 2 (2014), 241–254.

[33] Yizhou Qian and James Lehman. 2017. Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education (TOCE)* 18, 1 (2017), 1–24.

[34] Kanjun Qiu, Leah Buechley, Edward Baafi, and Wendy Dubow. 2013. A curriculum for teaching computer science through computational textiles. In *Proceedings of the 12th international conference on interaction design and children*. 20–27.

[35] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. 2009. Scratch: programming for all. *Commun. ACM* 52, 11 (2009), 60–67.

[36] Mitchel Resnick, Fred Martin, Robert Berg, Rick Borovoy, Vanessa Colella, Kwin Kramer, and Brian Silverman. 1998. Digital manipulatives: new toys to think with. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 281–287.

[37] Mitchel Resnick, Fred Martin, Randy Sargent, and Brian Silverman. 1996. Programmable bricks: Toys to think with. *IBM Systems journal* 35, 3.4 (1996), 443–452.

[38] Iván Ruiz-Rube, José Miguel Mota, Tatiana Person, José María Rodríguez Corral, and Juan Manuel Dodero. 2019. Block-based development of mobile learning experiences for the internet of things. *Sensors* 19, 24 (2019), 5467.

[39] Sue Sentance, Jane Waite, Lucy Yeomans, and Emily MacLeod. 2017. Teaching with physical computing devices: the BBC micro: bit initiative. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*. 87–96.

[40] Mazyar Seraj. 2020. Impacts of block-based programming on young learners' programming skills and attitudes in the context of smart environments. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. 569–570.

[41] Mazyar Seraj, Serge Autexier, and Jan Janssen. 2018. BEESM, a block-based educational programming tool for end users. In *Proceedings of the 10th Nordic Conference on Human-Computer Interaction*. 886–891.

[42] Mazyar Seraj, Cornelia S Große, Serge Autexier, and Rolf Drechsler. 2019. Look what I can do: acquisition of programming skills in the context of living labs. In *2019 IEEE/ACM 41st International Conference on SE: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 197–207.

[43] Mazyar Seraj, Cornelia S Große, Serge Autexier, and Rolf Drechsler. 2019. Smart Homes Programming: Development and Evaluation of an Educational Programming Application for Young Learners. In *Proceedings of the 18th ACM International Conference on Interaction Design and Children*. 146–152.

[44] Mazyar Seraj, Eva-Sophie Katterfeldt, Serge Autexier, and Rolf Drechsler. 2020. Impacts of Creating Smart Everyday Objects on Young Female Students' Programming Skills and Attitudes. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 1234–1240.

[45] Mazyar Seraj, Eva-Sophie Katterfeldt, Kerstin Bub, Serge Autexier, and Rolf Drechsler. 2019. Scratch and Google Blockly: How Girls' Programming Skills and Attitudes are Influenced. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*. 1–10.

[46] Snap4Arduino. 2019. Snap4Arduino Homepage. Retrieved January 6, 2023 from http://snap4arduino.rocks.

[47] Mauricio Verano Merino and Tijs Van Der Storm. 2020. Block-based syntax from context-free grammars. In *Proceedings of the 13th ACM SIGPLAN International Conference on Software Language Engineering*. 283–295.

[48] David Weintrop. 2019. Block-based programming in computer science education. *Commun. ACM* 62, 8 (2019), 22–25.

[49] David Weintrop, Alexandria K Hansen, Danielle B Harlow, and Diana Franklin. 2018. Starting from Scratch: Outcomes of early computer science learning experiences and implications for what comes next. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*. 142–150.

[50] David Weintrop and Uri Wilensky. 2017. Between a block and a typeface: Designing and evaluating hybrid programming environments. In *Proceedings of the 2017 conference on interaction design and children*. 183–192.

[51] David Weintrop and Uri Wilensky. 2017. Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education (TOCE)* 18, 1 (2017), 1–25.

[52] Bryant Zimmerman. 2017. Programming education for students of any age: LEGO® education-mindstorms® EV3 robotics. *Journal of Computing Sciences in Colleges* 33, 1 (2017), 42–43.

Received 2024-07-04; accepted 2024-08-08