iejee

International Electronic Journal of Elementary Education
December 2020, Volume 13, Issue 2, 183-198

Developing Children's Computational Thinking through Physical Computing Lessons

Sun Hee Min[a], Min Kyeong Kim[*,b]

Received : 25 June 2020
Revised : 5 October 2020
Accepted : 22 December 2020
DOI : 10.26822/iejee.2021.183

[a] Sun Hee Min. Department of Elementary Education, Ewha Womans University, Seoul, Korea.
E-mail: sunnym73@naver.com
ORCID: http://orcid.org/0000-0002-4715-8978

[*,b]**Corresponding Author:** Min Kyeong Kim. Department of Elementary Education, Ewha Womans University, Seoul, Korea.
E-mail: mkkim@ewha.ac.kr
ORCID: http://orcid.org/0000-0002-6788-9890

Abstract

In this study, we designed and applied physical computing lessons for elementary 6th-grade students based on the software education guidelines in the Korean 2015 Revised National Curriculum (Ministry of Education, 2015a). The participants of this study were ten 6th-grade students of an elementary school in Gyeonggi-do province in Korea. The physical computing lessons used in this study supported the active interaction of the digital world and the physical world by constructing a physical model using specific media, and controlling it with a program. In order to understand the changes in the students' computational thinking after the class, we analyzed these changes in terms of computational concept, computational practice, and computational perception. Research has shown that physical computing lessons materialize students' computational concepts through computational practices, and improve their computational perspectives through the use of authentic contexts. We expect that the physical computing lessons and analysis tools developed through this study will provide educational implications for future software education.

Keywords:
Computational Thinking, Computational Concepts, Computational Practices, Computational Perspectives, Physical Computing, Elementary Education

Introduction

Computer science plays a vital role in today's technologically and globally connected world. Thus, it is essential to introduce computing ideas to students early in their schooling (Yadav, Hong, & Stephenson, 2016). To prepare for this social change, countries such as the United States, the UK, Australia, India, and Israel view computational thinking as a key competency that future generations should possess and be able to apply to various subjects, such as mathematics and science (Ryu & Han, 2015). The school environment has a uniquely large impact on future generations, as educators continuously prepare their students for technology-driven futures (Griffths, Nash, Maupin, & Mathur, 2020).

Papert (1996) first introduced the concept of computational thinking, while Wing (2006) later defined it as a fundamental ability that allows people to design and think using the language of computation. In other words, computational thinking involves numerous skills, such as logical thinking, algorithm selection, and systematic thinking, which can be used to solve problems in a variety of learning contexts and in daily life, not only in professional computer science fields (Tsai, Wang, & Hsu, 2019). Denning (2005) suggested that computational thinking today includes the use of abstraction, mathematics for algorithmic development, and efficient problem solving.

In this regard, the Korean Ministry of Education (MOE, 2015b) emphasizes the importance of software education in the formal curriculum with the goal of strengthening competence, including information ethics and attitudes, while presenting computational thinking as a key competency of software education. For elementary school, the Ministry proposed expanding and reorganizing the existing contents of the Information and Communication Technologies (ICT) application into basic software literacy education (MOE, 2015a). Educational programming languages have been applied to this end (Song & Gil, 2017; Shin & Bae, 2015). Other tools include robots (Marion et al., 2017) and various types of educational media as physical computing tools (Alimisi, 2013; Bakke, 2013; Chandra, 2010; Felica & Sharif, 2014; Kim & Kim, 2016; Resnick, 2006).

Since 2019, software education has primarily been taught in grades 5 to 6, and efforts are under way to address the lack of learning time to cultivate the core competencies of software education (Han, Cheong, & Lee, 2017). To address the limited amount of available class time, which is a significant problem facing those who design computing lessons, some authors have studied computing not solely as a practical subject (Kim, 2015; Ryu & Han, 2015), but in relation to other subjects, such as mathematics and social studies (Shin, Cho, & Kim, 2013), as well as methods of applying it to different creative activities (Kim, 2015; Kim, Kim, & Ryu, 2013; Song, 2013). In addition, researchers have designed educational programming languages, such as Scratch and Entry, and various physical computing tools and software to enable students to learn through experience, considering the developmental level of elementary school students (Anglei et al., 2016; Kim & Lee, 2014)

Therefore, in order to implement software education, it is necessary to study educational media and various evaluation methods and to introduce different types of content. In particular, because computational thinking is emphasized as a core competency of software education, research is needed on how students express computational thinking, and how they can be evaluated. However, considering the limitations of approaching only the cognitive aspects of computational thinking (Kim, 2009), or of evaluating it as a learning output (Seiter & Foreman, 2013), it is necessary to study various aspects of how students understand computational concepts. For example, students must not only know the concepts, but also find the changes of computational concepts in practice. This means approaching computational thinking as a process of problem solving (Wiggns & McTighe, 2005; Bers, 2010; CSTA, 2012; Denning, 2017; Wing, 2006), in the sense that a concept is only meaningfully learned when a student can use it to solve a unique problem.

To consider computational thinking in terms of the harmony of thinking and computing technology, it is necessary to examine students' computational concepts in actual computational practice. In physical computing lessons, activities that explore changes in behavior using programming and robots are expected to help students shape computational thinking through the harmonization of concepts and practice by implementing students' ideas through computing technology. In addition, in order to continuously demonstrate computational thinking, it is necessary that students' active attitudes change through recognition conversion. Therefore, by assessing changes in computational perspectives in the classroom, we expect that this work will have implications for strengthening the attitude and capacity emphasized in software education. In addition, the appropriate result can be benchmarked against the relevance of ICT use within the wider personal, cultural, social and psychological context of a person's daily life (Talaee & Noroozl, 2019).

Therefore, in this study, we aimed to analyze the characteristics of computational thinking by designing physical computing lessons and applying them to elementary school students. In addition, we looked at the changes and features in computational concepts, computational practices, and computational perspectives in order to examine various aspects of computational thinking through physical computing lessons. In order to support the development of computational thinking, we provide concrete instructional design and application for physical computing lessons, implications for evaluation, and ideas for follow-up research.

The following research questions guided this study:

1. How was the physical computational class designed for elementary school students?

2. How did computational thinking appear to elementary school students who experienced the physical computing classes?

Research Background

Physical Computing

For more than 30 years, constructionist tool kits, robotics, and physical computing kits have been present in educational contexts (Przybylla & Romeike, 2014). As Resnick (2007) observed, "In today's rapidly changing world, people must continually come up with creative solutions to unexpected problems. Success is based not only on what you know or how much you know, but on your ability to think and act creatively." In this respect, the physical computing tools that connect the digital world with the real world are expected to provide students with creative experiences for problem solving.

Physical computing covers the design and realization of interactive objects and installations, and allows students to develop concrete, tangible products that arise from the learner's imagination. This can be used in computer science education to provide students with interesting and motivating access to the different topic areas of a subject in constructionist and creative learning environments (Przybylla & Romeike, 2014).

The physical computing environment uses sensors and actuators that can replace the human senses. A microcontroller can be used as a learning medium that is capable of robot programming (Kim & Kim, 2016; Seo & Kim, 2016).

Physical computing tools can be divided into robot, board, or modular types. In the case of robot type tools, a physical output device, such as a motor, is reinforced. Programming allows us to move robots and control output devices, such as sounds or lights, and if sensors are used, we can interact with the real world, such as by following lines or avoiding obstacles. Board type refers to electronic boards including microcontrollers. Because it is necessary to understand electric circuits and apply electronic knowledge, it is not easy to apply this type of robotic learning in elementary school classes.

Finally, modular type means that various input and output devices are assembled, connected to a microcontroller, and controlled using an educational programming language. Ultimately, students will be able to experience the process of designing, building, and programming their own robots. The modular type has the advantages of both robot type and board type, and it can help give learners practical experience, which can aid them in finding ideas and solutions for real life problem solving.

The use-modify-create model (Lee et al., 2011), a learning model for software education, emphasizes learning by making through hands-on experience. In particular, the authors of the model pointed out that environments should encourage active learning through play. The model also emphasizes that learners should experience inventions, rather than imitations or implementations of algorithms (Futscheck & Moschiz, 2010).

Physical computing takes computational concepts into the real world, so students can use those concepts in authentic environments. Physical computing activities are strongly connected to the dimensions of computational thinking, namely, abstraction, algorithmic thinking, automation, decomposition, debugging, and generalization (Psycharis et al., 2017). Sometimes, digital making is also referred to as tangible programming (or physical computing, digital fabrication, or creation of graspable user interfaces). Digital making is simultaneously a tangible representation of digital CT that moves beyond text-based computer programming and coding (Kotopoulos et al., 2017).

Physical computing is a form of computing science that is connected to the arts, which leaves a great deal of room for creative work in the classroom. Additionally, physical computing allows for various connections to other STEM subjects; for example, simulation of behavior relates to biology, collection and analysis of measurements relates to physics, and logical operations relate to mathematics (Schulz & Pinkwart, 2015). Ongoing research aims to determine the effects of physical computing on students in computer science classes by investigating its impacts on students' motivation, creativity, constructionist learning, learning success, growth in competences, and understanding of computer science and computing systems (Przybylla & Romeike. 2014).

Kabátová and Peárová (2010) suggested certain points to consider when designing a class. For example, activities with robotic models, programmable kits, and toys are good opportunities to organize the lessons in a constructionist way. The constructionist ideas and principles (Papert, 1999; Rusk et al., 2008) we promote in our lessons are:

- learning by doing, genuine achievement, hard fun and playful learning, learning through designing,

- technology as building material combined with artistic materials, and

- taking time, freedom to make mistakes, teamwork.

Taken together, physical computing that can take advantage of tools, including board, modular, and robotic type tools, can help students learn by building physical systems that connect the physical and computing worlds. Robotics exemplifies an appropriate use of technology to create meaningful, open-ended, problem-solving activities (Felica & Sharif, 2014). In addition, lessons that use physical computing tools provide opportunities for students to understand abstract concepts through realistic experiences, support students in shaping their ideas, and facilitate communication and the fostering of social skills.

Computational Thinking

Regarding computational thinking, Wing (2006) outlined the basic skills necessary for all people living in the 21st century, such as reading, writing, computation, and using computing technology to solve problems. She also emphasized that abstraction and automation are key elements of computational thinking. Computational thinking uses abstraction and decomposition to attack a large, complex task or design a large, complex system. Computational thinking is a fundamental skill for everyone, not just for computer scientists; and to reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability (Wing, 2008).

Bers (2010) defined computational thinking as a type of analytical thinking that has many similarities to mathematical thinking (e.g., problem solving), engineering thinking (design and evaluation processes), and scientific thinking (systematic analysis). The term grew out of the pioneering work of Papert and colleagues on design-based constructionist programming environments; it refers to ways of algorithmically solving problems, and to the acquisition of technological fluency (Papert, 1980).

Yadav, Hong, and Stephenson (2016) emphasized the importance of thinking to all students by suggesting algorithms, abstraction, and automation as key elements of computational thinking. The authors also emphasized that teachers' understanding of computational thinking is essential for incorporating it into the classroom environment.

The essence of computational thinking involves breaking down complex problems into more familiar/manageable sub-problems (problem decomposition), using a sequence of steps (algorithms) to solve problems, reviewing how the solution transfers to similar problems (abstraction), and finally determining if a computer can help more efficiently solve those problems (automation). These computational thinking steps are foundational to computer science, but their power and utility extend far beyond any single discipline. We believe that the computational thinking ideas outlined in this paper are key to moving students from merely being technology literate, to using computational tools to solve problems and represent knowledge. Developing teachers' understanding of computational thinking and highlighting connections to their curricular context is key to successfully embedding computational thinking in K-12 classrooms. Tedre and Denning (2016) pointed out that CT as a concept has been studied for a longer time than suggested by Wing (2006), and it is necessary to know about problems, ideas, and risks that have already been solved during this history of CT. Also they examined a number of threats to CT initiatives: lack of ambition, dogmatism, knowing versus doing, exaggerated claims, narrow views of computing, overemphasis on formulation, and losing sight of computational models.

Brennan and Resnick (2012) suggested a way of approaching the three aspects of computational thinking in a study using Scratch. Having articulated the framework for computational thinking (concepts, practices, and perspectives), they described three approaches to assessing the development of computational thinking in young people who are engaging in design activities with Scratch. Computational practices focus on the process of thinking and learning, moving beyond what the students are learning to how they are learning

Table 1

Strength and Limitations of Assessment Approaches

Approach	Concepts	Practices	Perspectives
Approach #1: Project Analysis	Presence of blocks indicates conceptual encounters	N/A	N/A (possibly by extending analysis to include other website data, like comments)
Approach #2: Artifact-Based Interviews	Nuances of conceptual understanding, but with limited set of projects	Yes, based on own authentic design experiences, but subject to limitations of memory	Maybe, but hard to ask directly
Approach #3: Design Scenarios	Nuances and range of conceptual understanding, but externally selected projects	Yes, in real time and in a novel situation, but externally selected projects	Maybe, but hard to ask directly

(Brennan & Resnick, 2012, p. 22)

(Brennan & Resnick, 2012). Table 1 shows the limitations of each method of analyzing computational thinking.

Alternatively, researchers have also conducted studies on computational thinking in terms of subject and problem solving. Weintrop, Beheshti, Horn, Orton, Jona, Trouille and Wilensky (2016) discussed the definition and characterization of computational thinking in secondary mathematics in conjunction with STEM education.

Methodology

Lesson Design

In this study, to assess students' practical understanding, we analyzed computational thinking centering on the computational practices that appeared in class. To this end, a learning environment in which the students' own ideas could be manifested was provided by utilizing physical computing tools that support active interaction between the physical and computing environments.

In these physical computing lessons, the subject, content, and evaluation method were designed to teach the algorithms and programming areas emphasized in the guidelines for software education (Ministry of Education, 2015b) based on the use-modify-create model of robots (Lee, Martin, Denner, Coulter, Allen, Erickson et al., 2011).

Furthermore, the "Maze Escape" and "School Bus" lessons were developed and applied in conjunction with mathematics and social studies concepts. The classes explored the core concepts, factual content, and achievement skills of elementary school practical art subjects. In order to develop the subjects of the lessons, a preliminary study was conducted on four 5th graders and twenty 5th – 6th graders over the course of one year. In addition, whether real life application, inquiry, enjoyment, and cooperation were possible (Shin & Bae, 2014), and whether it was possible to connect with different regions (Choi, Choi, Ahn, Hong, & Jung, 2015) were all considered.

Participants

The subjects of this study were ten male 6th grade elementary school students in Gyeonggi-do province, Korea, who participated in the class voluntarily after being informed of the purpose of this study in advance. Before the students participated in the class, a separate introductory session was used to explain the purpose and contents of the study, and consent forms were used to obtain the student's and guardian's signatures. Physical computing lessons were held every Friday for a total of six sessions, 80 minutes per session; before each class started, 80 minutes of extra time was provided to help students understand the medium. The physical computing tool used LEGO bricks to assemble the body of a robot.

Data Collection & Analysis

In this study, quantitative and qualitative data were collected in order to analyze changes in students' computational practices following physical computing lessons. The main researcher participated in the research, introduced class topics, and observed the students' problem solving process; she also played the role of a teacher in assisting the study participants with overcoming any difficulties experienced during the physical computing lessons. In addition, the students' normal teachers were encouraged to help with class recording and collection of various materials. The rubrics and test tools used in the research process were revised and supplemented based on the results of the preliminary study through consultations with experts in elementary education and robotics education, and elementary school teachers.

Data collection was carried out through a computational concepts test conducted before and after class, observation of class participation, interviews, activity sheets, and anecdotal records. For data analysis, quantitative and qualitative analyses were performed using a hybrid research method to grasp the computational thinking of students who applied the physical computing lessons.

For the quantitative analysis, computational concepts test scores, worksheets, and interviews conducted using a computational practices rubric, as well as data collected through anecdotal records, were statistically analyzed. For the qualitative analysis, we attempted to understand students' computational thinking processes by observing students' participation in class, interviewing the students, and analyzing outputs and activities. In addition, a single case study was conducted to assess changes in individual computational thinking, and we attempted to explore certain aspects of computational thinking in detail through individual examples of how computational concepts, computational practices, and computational perceptions appear in physical computing lessons.

Assessment 1: Computational Concepts

The UK Bebras Computational Challenges (2015) is an online competition open to students in the UK

and English-speaking international schools around the world; it requires intelligence, but no previous knowledge. The hope is that the competition will increase youngsters' general interest in computer science and help them to understand that computational thinking has far-reaching applications in solving all sorts of life's problems.

In the computational concepts test, a pre-test and post-test were conducted using the same question; no answer to the test question was provided. It was conducted at 2-month intervals, including the class time between the pre-test and post-test. To measure computational concepts, the participating students had 40 minutes to solve 15 questions. However, when we scored their work, we did not use the additional scoring rubric as required by the UK Bebras Computational Challenges. Instead, correct answers received 1 point, and incorrect answers received no points (i.e., points were not deducted for incorrect answers). The post-test Cronbach's alpha for computational concepts excluding items 3 and 9 was .764.

Assessment 2: Computational Practices

To assess computational practices, we reconstructed the relevant rubric based on the three areas of experience—problem-solving, algorithm, and programming—presented in the MOE's Software Education Guidelines (MOE, 2015b). We evaluated the experiential domain of the problem-solving and algorithmic processes using revised rubrics based on Choi (2014) and Brennan, Balchm, and Chung (2015), respectively. We reconstructed the programming experience area by referring to the robot design rubric of the For Inspiration & Recognition of Science & Technology LEGO League (2015).

We also conducted interviews with the students based on Brennan and Resnick's (2012) observation that it is difficult to evaluate computing practices solely by analyzing output. The interview questions gauged how students understood and structured the problems, searched for solutions, and understood the role of programming in class activities. Students introduced their projects and explained their problem solving methods and the ideas or people who helped them. In addition, they explained the differences between how to program, and how to apply and explain the commands and solutions. To help students remember, the interviewer used their own output as an example. Table 2 shows some interview items that were used to evaluate computational practices.

Based on the actual experiences of the students themselves, we conducted two interviews using the students' outputs immediately after the end of each activity topic, considering the limitation of memory. We analyzed the reliability of the three graders in the scoring of the students' computational practices. Table 3 presents the inter-scorer reliability, assessed using Pearson's correlation coefficient.

Assessment 3: Computational Perspectives

Brennan, Balchm, and Chung (2015) divided computational perspectives into three areas: expressing, connecting, and questioning. They analyzed computational perspectives by looking at students' perspectives of physical computing lessons (perspectives of expression, cooperation, and use). For this study, an anecdotal record consisted of two narrative questions and five multiple choice questions, and students wrote anecdotes for each class.

Table 2

Interview Items for Evaluating Computational Practices

Strands	Interview details
Understanding and structuring the problem	Introduce your project.
Searching for problem solutions	How did you solve the problem? Did you have any ideas or people who helped you solve the problem? Have you made any changes in today's activity? Why did you fix it like that? Describe how you tried to solve the problem.
Understanding programming	How did you program it to solve the problem? What commands did you use? Please explain the commands used. Was there a difference between what you expected, and what was real?

Table 3

Inter-Scorer Reliability: Computational Practices

Items	Activity themes	A–B	A–C	B–C
Worksheets	#1 Maze Escape	.994**	.797**	.833**
	#2 School Bus	.748*	.795**	.808**
Interviews using outcomes	Total	.934**	.911**	.875**

** $p < .01$

Results and Discussion

Physical Computing Lessons

We conducted each physical computing lesson three times, centered on two themes, maze escape and school bus. Based on the practical subjects of the curriculum, which was last revised in 2015, the subjects of mathematics and social studies were linked to each other. We considered whether a topic was explored, used in daily life, facilitated cooperation, or caused pleasure (Shin & Bae, 2014), whether the use–modify–create model (Lee, Martin, Denner, Coulter, Allen, & Erickson, 2011) or the algorithmic invention model (Futscheck & Moschiz, 2010) could be applied, and whether the lesson could be connected with the community (Choi et al., 2015). Table 4 shows the contents of the lessons and activities conducted in this study.

The main activity of the physical computing lessons was to create and program robots; this allowed students to invent algorithms through experience. First, in the "Analyze problem" stage, the students identified a problem and clarified that problem by analyzing the given situation and conditions; at this point, the students removed any unnecessary information from consideration. Second, in the "Find ideas" stage, the students collected data and generated various ideas to identify the best ones; in addition, we constructed robots, identified the properties of the media, and collected ideas for problem solving. Third, in the "Formulate algorithms" stage, we designed a method of realizing an idea and designed a concrete process that included a program to specify the necessary algorithm. Fourth, in the "Play algorithms" stage, the program was tested based on each algorithm. Fifth, in the "Reflect algorithm" stage, problems were identified and corrected while the results were evaluated. Students checked their problem-solving processes and algorithms to find and fix errors. They also shared their results and conducted self- and peer evaluations in order to objectively view their own output. The students produced robots as a means of solving problems by programming and experimenting in teams of two. The students built and moved the robot themselves, embodied their ideas with algorithms, and debugged their programs through execution. Real life-based problematic situations helped students immerse themselves in the learning process, and easy-to-edit robots and programming tools helped students to check their ideas. Figure 1 shows the robot and maze used in class.

Students observed the maze, moved like a robot, and discussed their ideas with other students. In addition, the discussion was organized using pictures, texts, and symbols. Students acted like robots, extracting the elements necessary for movement, understanding problems, and finding ideas. Through this process, the robot's behavior was sequentially arranged, and each algorithm was created. The students' algorithms were embodied through programming, and modified and supplemented through practice. Figure 2 illustrates the students' ideas that allowed the robot to escape the maze.

Table 4

Physical Computing Lessons: Contents and Activities

Lesson steps	Activity themes	
	#1. Maze Escape	#2. School Bus
Analyze problems and find ideas	Make robots using basic building instructions Explore robot movement	Build a town map and school bus, share ideas, and make a plan with peers
Formulate and play algorithms	Modify robots, get directions using robots, programming, and testing	Recreate school bus, do programming and test
Play and reflect algorithm	Execute maze escape	Optimize school bus movements

Figure 1

Students' Robot and Maze Escape Activities

Figure 2
Students' Ideas for Escaping The Maze

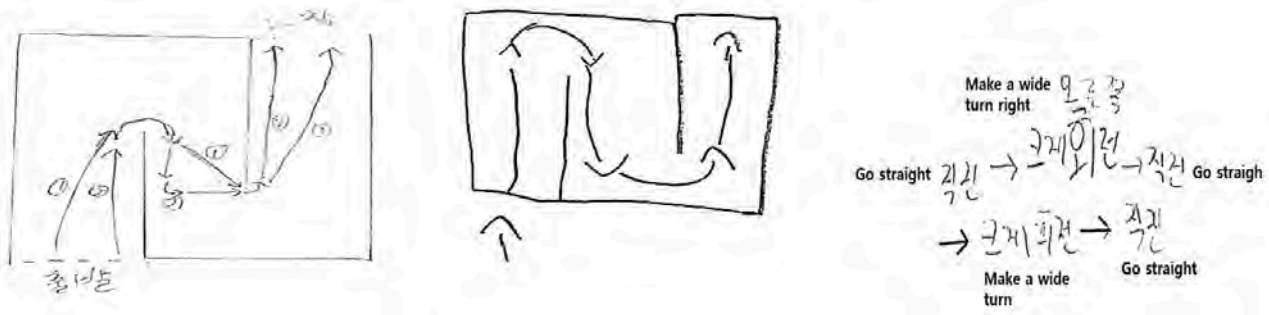
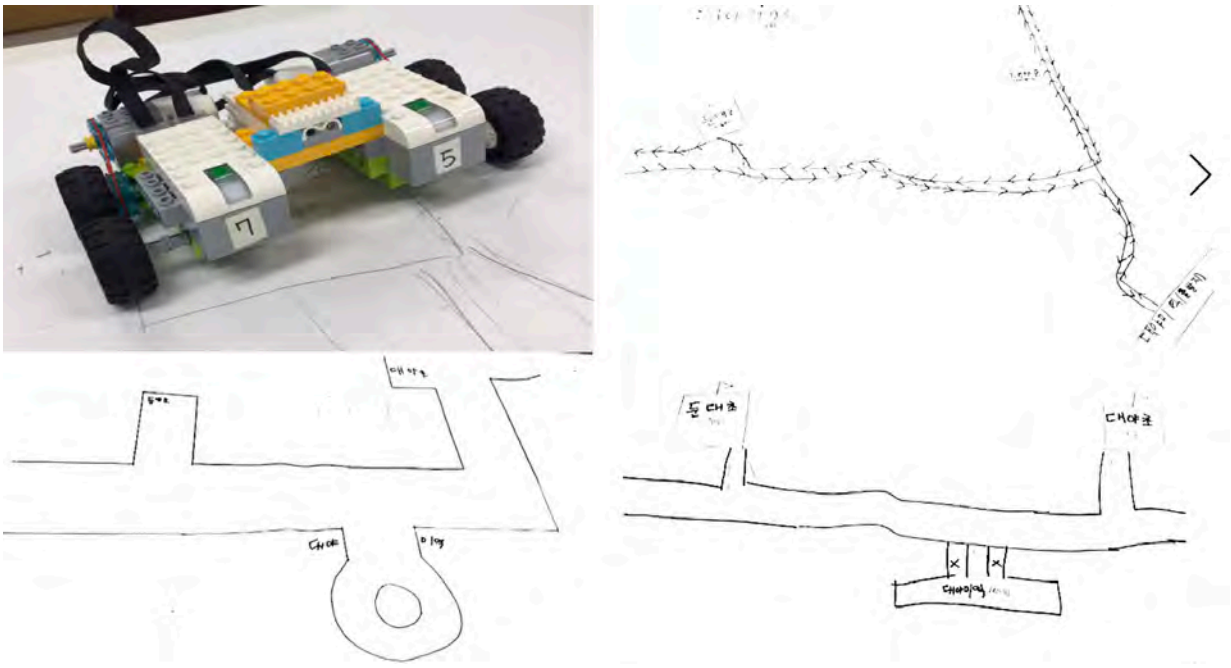


Figure 3
Students' Outcomes: School Bus Robots and Students' Maps



In the maze escape class, the students used a basic type of robot to allow them to focus on algorithms and programming strategies. In order to move through the maze, the students modified the robot in their own ways. The maze was divided into two stages, which were delineated by a middle point and a final goal point, so the students could solve the task step-by-step. This activity used assembly diagrams to create and test basic models, so the students were able to focus on programming. In addition, the students connected their two robots according to their own ideas. The students used the command function to control the robot by specifying each movement of the robot as an action. In particular, to move the robot, it was given a command to forward, reverse, or change direction. To this end, the students identified a method of controlling the motor connected to the wheel. The mission to go through the maze continuously challenged the students to solve problems and made them feel as if they were the real drivers of the robot.

In the school bus class, the students created a school bus (robot) that could solve a problem in their local neighborhood. Students made a school bus and created a map that connected two schools and subway stations in the areas where they lived. Students created robots by adding their own ideas on top of the basic robot model they had experienced in the maze activities. The robot was programmed to drive on the road that they mapped. Figure 3 shows the robots and maps that were used in class.

This class consisted of 3 lesson stages per activity, and lasted for 80 minutes per lesson stage. In the first stage, the students analyzed problems and came up with ideas to address those problems. In the second stage, the algorithm was formulated and executed, which was accomplished by converting each algorithm into a program. The 3rd stage consisted of performing and reflecting algorithms. At this time, self-evaluation and mutual evaluation were conducted while observing the movement of the robot. Figure 4 presents the problem-solving structures of the maze escape and school bus classes.

Computational Concepts

To analyze the changes in computational concepts of the students who participated in the physical computing classes, we administered the UK Bebras Computational Challenge (2015) questionnaire before and after the class, and statistically analyzed the results. Each item on the questionnaire is based on five elements: abstraction, evaluation, generalization, decomposition, and algorithmic thinking of the computational concept. The computational concept test consists of a total of 15 questions, each of which are assigned one point; the maximum score is 15 points.

The computational concept test results revealed a difference between the pre- and post-intervention scores of -3.074 at p= .013, a statistically significant difference at p< .05 (see Table 5). In short, following the physical computing lesson, the students showed an improved understanding of computational concepts.

To fully understand the changes in the students' computational concepts, we extracted scores for algorithmic thinking, abstraction, and decomposition, which are the elements of automation that scholars commonly highlight as core elements of computational thinking. Table 6 shows the components and difficulty of each item.

Among the problems with high difficulty level, on the robot painting problem, the students showed nearly twice the number of correct answers on the post-test vs. the pre-test. For the monster problem, they had more than double the number of correct answers on the post-test. The results showed that the physical

Figure 4
Flow Charts: Maze Escape (Left) and School Bus (Right)

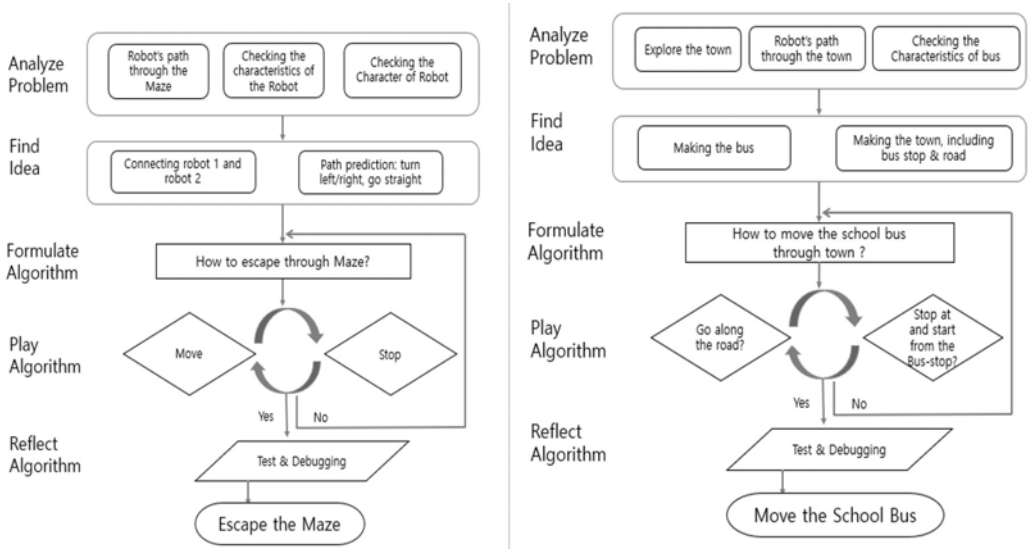


Table 5
Computational Concepts Test Results

	M	*SD*	cases	*t*	*p*
Pre-test	8.80	2.485	10	-3.074	.013*
Post-test	10.10	2.514	10		

* p< .05

Table 6
Computational Concepts: Algorithmic Thinking

Problem	level	Algorithmic thinking	Abstraction	Decomposition	Testing results		Variation
					Pre-test	Post-test	
Watering	low	•			10	10	0
Tic Tac Toe	middle	•	•		10	10	0
Abacus		•	•	•	8	9	+1
Village Network		•	•	•	8	7	-1
Drawbot	high	•	•	•	3	5	+2
Monster		•	•	•	3	7	+4

computing lesson helped the students solve complex problems and discover and apply rules. This supports previous research showing that computational thinking generates the strategic knowledge that is necessary for problem solving (Bers, 2010), and that cognitive tasks related to computational thinking can be addressed through programming (Grover & Pea, 2013). In particular, in the maze escape and school bus tasks in this study, the problem is to grasp the movement path of the robot; this task involves problem decomposition, algorithmic thinking, and abstraction.

Computational Practices

To identify the changes in the students' computational practices that took place during the physical computing classes, we assessed changes in four areas: understanding and structuring the problem, exploring the problem-solving method, experiencing the algorithm, and understanding programming. The maximum score for each area on the computational practice test was 5 points. Figure 5 shows the students' average computational practices scores.

In the area of comprehension of computational practices, the students showed higher average scores in understanding the problem, structuring the area, and programming in the school bus class than in the maze escape class. This is likely because the school bus class is based on material that directly relates to the students' daily lives, so their scores rose in the area of problem understanding. In the area of understanding programming, the students were familiar with programming using commands because they understood the functions of each instruction through practice.

In contrast to the other two computational practice areas, in the areas of problem solving and algorithm experience, the mean score was lower in the school bus class than in the maze escape class. We took this to mean that the students could relate better to the school bus scenario, so it interested them more, but that developing actual bus routes was complicated, and the complex considerations made it difficult for the students to develop algorithms. Regarding the overall computational practice scores by class subject, the students had a higher average score, 9.43, in the school bus class than they did in the maze escape class, 9.22. Table 7 shows the individual students' computational practice scores.

In the problem understanding and programming area, the students' average score was higher in the school bus activity than in the maze escape activity. This may be because the students fully understood a need for a school bus, and the task was related to their everyday lives, so their scores increased in the problem understanding area. In addition, it seems that the functions and programming methods of each area of instruction gradually became more familiar with iterative programming. On the other hand, in the problem solving and algorithm area, the average score of the students in the school bus classes decreased. This seems to be because the situation became more complicated, so it was difficult to address it with an algorithm, despite the fact that the school bus problem is related to the students' everyday lives and the students were interested in the problem.

We conducted correlation analysis to analyze the relationship between students' computational practice scores by subject and the computational practices they displayed in their interviews. The result was .863, which was statistically significant ($p < 0.01$). To analyze the relationship between computational practices and concepts, Pearson's correlation analysis was conducted, which showed correlation

Figure 5

Computational Practice Scores in Physical Computing Lessons

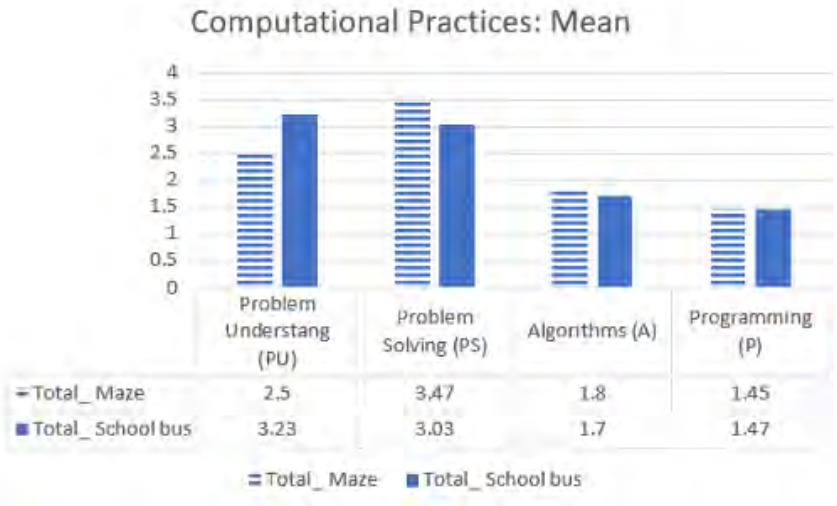


Table 7

Individual Students' Computational Practices Scores

No	Maze Escape					School Bus				
	PU*	PS*	A*	P*	Total	PU*	PS*	A*	P*	Total
1	3.67	4.83	2.33	2	11.83	3.33	2.83	1.67	1.67	9.5
2	2	4	1.67	1	8.67	2.67	3.5	1.67	1.5	9.34
3	2	3	1.67	1.67	8.34	3.67	2.33	1	1	8
4	2	3.67	2.33	2.33	10.33	4	2.5	1.67	1.67	9.84
5	2	4.5	1.67	1	9.17	3	3.67	2.67	1.5	10.84
6	2	4.17	3	2.5	11.17	4	2.5	1.33	1	8.83
7	2.67	2	1	1	6.67	3	2	1.67	1.5	7.5
8	2	2	1.67	1	6.67	2.33	3	1.33	1.5	9.16
9	3.67	3.33	1	1	9	3.33	4.17	1.67	1.67	11.51
10	3	3.17	1.67	1	8.84	4	3.83	2.33	1.67	9.83
Mean	2.50	3.47	1.80	1.45	9.22	3.23	3.03	1.70	1.47	9.43

**PU: Understanding the problem & finding ideas, PS: Exploring to solve the problem, A: Play algorithm, P: Understanding programming.*

coefficients of .758 (p< .05) for the pre-test and .877 (p< .01) for the post-test. The correlation coefficient for computational practices (r= .711, p< .05) was statistically significant, indicating that, when conducted after the classes, interpersonal interviewing was effective. The post-test scores for computational concepts and the activities (r= .877, p< .01) and interviews (r= .711, p< .05) were all highly correlated. We found that, following the physical computing lessons, the students were able to demonstrate their understanding of computational practices in the class activities and in interviews.

Computational Perspectives

We used anecdotal records to evaluate the student's perspectives on computing accidents (Brennan & Resnick, 2012; Brennan, Balchm & Chung, 2015). We asked students to rate their expression, collaboration, and use of robots and computing abilities with regard to robots and computing on Likert scales of 1 to 5 for each item. Table 8 shows the students' average scores for each item in each step, which were used to examine the changes in their computational perceptions.

Next, Figure 6 shows the students' average scores. In terms of computational perspective, students who attended the physical computing lessons were able to use robots and computing to create something, to be aware of expressions, to collaborate with peers, and to solve problems.

We found it noteworthy that students' expression increased in the actual experiential algorithm execution and reflection stages. In the problem analysis and "find ideas" stages, the expressing, collaborating, and use of robots and computing perspectives of the school bus class (#1) increase more than those of the maze escape class (#2). Our analysis shows that the process of finding problems and discussing ideas with friends leads to a change in computing perspective. In particular, presenting practical problems, such as those having to do with school buses, helps improve computational thinking, which is in line with previous work (Bers, 2010; CSTA, 2012; Wing, 2006).

Pearson's correlation analysis of the relationships between computational concepts and perspectives revealed no statistically significant results (p< .05). The correlation coefficient between computational perspectives and computational practice was .469, but this finding was not significant (p< .05). We found the relationships between the subdomains to be related to the students' recognition of the expression

Table 8

Average Computational Perspectives Scores

Activity \ Step	Analyze Problem & Find Ideas			Formulate Algorithms & Play Algorithms A			Play Algorithms & Reflect Algorithms		
	E-1	C-1	U-1	E-2	C-2	U-2	E-3	C-3	U-3
#1: Maze escape	3.4	3.4	3.3	4	3.8	3.8	4.1	4.1	3.8
#2: School bus	4	4	3.9	3.5	3.5	3.7	3.9	4.1	4
Average score	Maze: Expression (3.83), Collaboration (3.77), Use of robots and computing (3.3)								
	School bus: Expression (3.8), Collaboration (3.87), Use of robots and computing (3.87)								

**E: Expression, C:Collaboration, U: Use of robots and computing*

of school buses and computational perspectives (r= .655, p< .05), and their perspectives of connections (r= .735, p< .05). In the school bus class, there was a strong correlation between the expression of computational perspectives and the perspectives of cooperation, given that the correlation coefficient was between .60 and .80.

In summary, the results of this study show that students who participated in physical computing lessons showed an improved understanding of computational concepts. In particular, the number of correct answers increased for the elements of algorithmic thinking, decomposition, and abstraction, including on items with high complexity. The overall average score for computational practices was higher after the school bus lesson, 9.43 points, than after the maze escape lesson, 9.22 points, indicating that the students' overall computational practice comprehension improved. Students' computational practices improved in both classes in terms of their understanding of problems, structures, and programming. The mean scores for problem-solving search area and algorithm experience were lower after the school bus lesson than the maze escape lesson. We interpreted this difference to reflect the fact that the school bus lesson introduced a complex real-life problem that students found difficult to solve, and we concluded that difficulty solving the problem influenced the students' understanding of programming.

In terms of computational perspectives, in the maze escape lesson, as the lesson progressed, scores related to expression, collaboration, and utilization increased. In contrast, in the school bus lesson, we found that the students had difficulty developing algorithms for the complicated bus route problem, and that this difficulty was reflected in their lower algorithm formatting and performance scores.

Conclusion

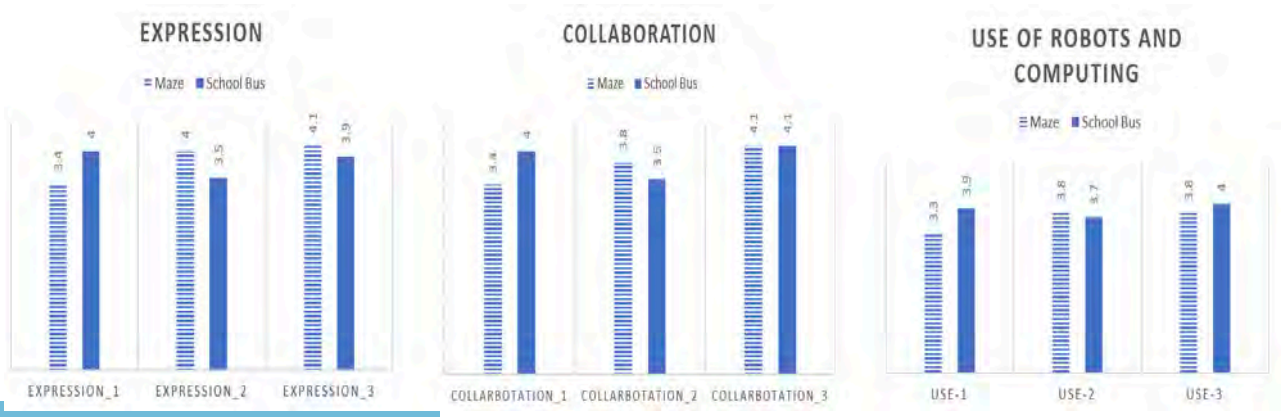
In this study, we developed and applied two sets of physical computing lessons for elementary school students in preparation for their imminent computing education. We measured the students' computational thinking in terms of concepts, practices, and perspectives, not just cognitive aspects, in an effort to overcome previous researchers' focus on only cognitive changes, such as changes in logical thinking (Lee, Cheon, & Kim, 2017), programming understanding (Resnick, 2006), and problem-solving ability (Kabátová & Pekárová, 2010; Son & Son, 2014).

This study has some limitations. For example, it targeted voluntary participants, only male students participated, and it was conducted with a small number of students (10). Thus, it is difficult to generalize the results of this study.

We found that the physical computing lessons supported problem decomposition, abstraction, and algorithmic representations that are covered in students' computational concepts. In particular, the lessons provided an opportunity for students to compare and modify their own mental models and the real models they created for the experiments. The robot programming activities in which students participated during class helped to shape their computational concepts through computational practices. In addition, instructional activities that described how school buses move around schools and neighborhoods helped shape a problem in the students' daily lives. This supports the abstraction, extraction, and expression of key information, thus helping students to distinguish important information from ancillary information, and thereby form computational concepts. This is also reflected in the current emphasis on providing opportunities to

Figure 6

Computational Perspectives: Expression, Collaboration, and Use of Robots and Computing



develop design in software education (Kim & Han, 2012; Jeon & Han, 2016; Brennan, Balchm, & Chung, 2015).

Second, the physical computing lesson facilitated the students' computational thinking by supporting them in rendering their ideas into computing technology, while solving problems with hardware (robots) and software (programming). In addition, the action of formalizing algorithms was educational in terms of activating students' mathematical expressions (symbols, texts, pictures, etc.). In particular, it was possible to utilize physical computing tools to support inquiry learning by enabling variable control and feedback (Reys, Lindquist, Lamdin & Smith, 2015).

Third, the problematic situations encountered by the students in these classes changed their computational perspectives and allowed the students' active interest in the subjects to manifest. Students who took part in the physical computing classes recognized the necessity of cooperation when using robots to solve problems. This resulted in the emergence of active attitudes, such as actively learning the programming language and communicating their learning to other students. In fact, after the study, the students formed clubs and volunteered to conduct research on robots and coding.

This study has many educational implications. The factors that affect computational thinking can be identified by analyzing the patterns of computational concepts and perspectives that are revealed through computational practice. In particular, this study provides concrete implications of the use of physical computing lessons for elementary students, and has educational implications for teachers, researchers, and parents who will be conducting software education in the future.

Based on the results of this study, we suggest the following. First, considering that physical modeling and programming support the formation of computational concepts in physical computing lessons, it is necessary to develop computational practice activities so that students construct knowledge while constructing actual models. Second, when conducting software training, it is important to encourage an understanding of computational concepts, including problem resolution, abstraction, and algorithmic thinking, rather than focusing on automation. Considering that the students' ideas and the ways in which they formed algorithms differed depending on how they perceived the problem, education on the abstraction phase that breaks down the problematic situation and emphasizes an understanding of the core concepts should be made a priority. Third, we conducted this study with only male students, and further study is needed to assess whether there might be gender differences in the results, particularly given that most of the male participants had related experience and a high degree of interest in the class subject.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgement

This study was based on parts of a dissertation title, 'A study on development and application of physical computing lessons to promote computational thinking in elementary school students' by the first author.

References

Alimisis, D. (2013). Educational robotics: Open questions and new challenge. *Themes in Science & Technology Education, 6*(1), 63-71.

Anglei, C., Voot, J., Fluck, A., Webb, M., Cox, M., Maiyn-Smith, J., & & Zagami, J. (2016). A K-6 computational thinking curriculum framework: Implications for teacher knowledge. *Educational Technology & Society, 19*(3), 47-57.

Bakke, C. K. (2013). *Perceptions of professional and educational skills learning opportunities made available through K-12 robotics programming* (Unpublished doctoral dissertation). University of Capella.

Bebras Computational Challenges Recommended by UK Bebras Challenges (2015). Retrieved from http://bebras.uk

Bers, M. U. (2010). The tangible robotics program: Applied computational thinking for young children. *Early Childhood Research & Practice, 12*(2), 1-19.

Brennan, K., & Resnick, M. (2012). *Using artifact-based interviews to study the development of computational thinking in interactive media design.* Paper presented at annual American Educational Research Association meeting, Vancouver, Canada.

Chandra, V. (2010). *Teaching and learning mathematics with robotics in middle-year of schooling.* Paper presented at the envision the future: The role of curriculum materials and learning environments in educational reform. Hualien, Taiwan.

Choi, D. H., Choi, S. H., Ahn, J. J., Hong, H. J., & Jung, G. C. (2015). *Education for Sustainable Development (ESD) by teachers' doing: Working with future generation.* Seoul: UNESCO.

Choi, H. S. (2014). Developing lessons and rubrics to promote computational thinking. *Journal of The Korean Association of Information Education, 18*(1), 57-64.

Computational Thinking recommended by Computer Science Teachers Association (2012). Retrieved from https://www.csteachers.org

Creative Computing recommended by Brennan, K., Balchm, C., & Chung, M (2015). Retrieved from http://scratched.gse.harvard.edu/guide/

Denning. P. (2009). The profession on IT: Beyond computational thinking. *Communications of the ACM, 52*(6), 28-30. Retrieved from http://doi.org/10.1145/1516046.1516054

Denning. P. (2017). Computational thinking in science. *American Scientist, 105*(1), 13-17. Retrieved from http://doi.org/10.1511/2017.124

Felica, A., & Sharif, S. (2014). A review on educational robotics as assistive tools for learning mathematics. *International Journal of Computer Science Trends and Technology, 2*(2), 62-84.

For Inspiration & Recognition of Science & Technology LEGO League (2015). Retrieved from https://www.firstlegoleague.org/

Futscheck, G., & Moschitz, J. (2010). Developing algorithmic thinking by inventing and playing Algorithms. *Constructionism*, 1-10.

Griffths, A. j., Nash, A.M., Maupin, A., & Mathur, S. K. (2020). Her voice: Engaging and preparing girls with disabilities for science, technology, engineering, and math careers. *International Electronic Journal of Elementary Education, 12*(3), 293-301.

Grover, S., & Pea, R. (2013). Computational thinking in K-12: Review of the state of the field. *Educational Researcher, 42*(1), 38-43.

Han, J. M., Jung, U. R., & Lee, Y. J. (2017). Analysis on research trends related computational thinking in Korea. *The Korean Association of Information Education: Summer Conference, 21*(2), 3-5.

Jeon, S. J., & Han, S. K. (2016). Development of UMC teaching and learning strategy for computational thinking. *Journal of The Korean Association of Information Education, 20*(2), 131-138.

Kabátová, M., & Peárová, J. (2010). Learning how to teach robotics. *Constructionism*, 1-8.

Kim, D. J., Kim, S. H., & Ryu, H. C. (2013). STEAM educational outreach program based on physical computing. *Journal of The Korean Association of Information Education, 17*(2). 279-283.

Kim, J. H. (2009). *Secondary education program for problem-solving ability based on computational thinking* (Unpublished doctoral dissertation). Korea University, Seoul.

Kim, J. H., & Kim, D. H. (2016). Development of physical computing curriculum in elementary school for computational thinking. *Journal of The Korean Association of Information Education, 20*(1), 69-82.

Kim, K. S. (2017). An analysis of software curriculum of Korean elementary teacher education school. *Journal of The Korean Association of Information Education, 21*(6), 723-732.

Kim, M. J., & Lee, T. W. (2014). Development of the software educational program using LEGO WEDO. *Journal of The Korean Association of Information Education, 18*(2), 37-40.

Kim, S. H., & Han, S. K. (2012). Design-based learning for computational thinking. *Journal of The Korean Association of Information Education, 16*(3), 319-326.

Kim, T. H. (2015). *STEAM education program based on programing to improve computational thinking ability* (Unpublished doctoral dissertation). Jeju University, Jeju.

Kotopoulos, D., Flyod, L., Khan, S., Namukase, I. K., Somanath, S., Weber, J., & Yiu, C. (2017). A pedagogical framework for computational thinking: Mathematics and programming. Retrieved from http://doi.org/10.100/s40751-017-0031-2

Lee, I., Martin, F., Denner, J., Coulter, B., Allen, W., Erickson, J., Malyn-Smith, J., & Wener, L. (2011). Computational thinking for youth in practice. *Association for Computing Machinery, 2*(1), 32-37.

Lee, Y. J., Jeon, H. G., & Kim Y. S. (2017). Development and applyment selection standards of physical computing teaching aids for elementary SW education according to the 2015 revised curriculum. *Journal of The Korean Association of Information Education, 21*(4), 437-450.

Marion, P., Deits, R., Valenzuela, A., d'Arpino, C. P., Izatt, G., Manuelli, L., Antone, M., Koolen, T., Carter, J., Fallon, M., Kuindersma, S., & Tedrake, R. (2017). Director: A user interface designed for robot operation with shared autonomy. *Journal of Field Robotics, 34*(2), 262-280.

Ministry of Education (2015a). *2015 Revision curriculum*. Korea.

Ministry of Education (2015b). *Activation plan for SW education in K-12*. Korea.

Papert, S. (1980). *Mindstorms: children, computer and powerful ideas.* Basic Books.

Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning, 1*(1), 95-123.

Papert, S. (1999). *Ghost in the Machine. ZineZone.com interview on how computers fundamentally change the way kids learn.* Retrieved from http://www.papert.org/articles/GhostInTheMachine.html\

Resnick, M. (2006). Computer as pain brush: Technology, play, and the creative society. Play=learning: How play motivates and enhances children's cognitive and social-emotional growth, 192-208.

Resenick, M. (2007). Sowing the seeds for a more creative society. *Learning & Leading with Technology*, 18–22.

Przybylla, M., & Romeike, R. (2014). Physical computing and its scope – Towards a constructionist computer science curriculum with physical computing. *Informatics in Education, 13*(2), 241-254.

Psycharis, S., Kalovrektic, K., Sakellaridi, E., Korres, K., & Mastorodimos, D. (2017). Unfolding the curriculum: Physical computing, computational thinking and computational experiment in STEM's transpupillary approach. *European Journal of Engineering Research and Science, Special Issue: CIE 2017*, 19-24.

Reys, R. E., Lindquist, M. M., Lamdin, D. V., & Smith, N. L. (2015). *Helping children learn mathematics* (11th edition). John Wiley & Sons.

Rusk, N., Resnick, M., Berg, R., & Pezalla-Granlund, M. (2008). New pathways into robotics: Strategies for broadening participation. *Journal of Science Education and Technology, 17*(1), 59-69.

Ryu, M. Y., & Han, S. K. (2015). Development of computational thinking-based educational program for SW education. *Journal of Korean Association of Information Education, 19*(1), 11-20.

Schulz, S., & Pinkwart, N. (2015). Physical computing in stem education. In *Proceedings of the Workshop in Primary and Secondary Computing Education* (pp. 134-135). ACM.

Seiter, L., & Foreman, B. (2013). Modeling the learning progressions of computational thinking of primary grade students. In *Proceeding of the Ninth Annual International Association for Computing Machinery Conference* (pp. 59-66). ACM.

Seo, J. H., & Kim, Y. S. (2016). Development and application of educational contents for software education based on the integrative production for increasing the IT competence of elementary students. *Journal of Korean Association of Information Education, 20*(4), 357-366.

Shin, S. K., & Bae, Y. K. (2014). Analysis and implication about elementary computer education in India. *Journal of the Korea Association Education, 18*(4), 585-594.

Shin, S. K., & Bae, Y. K. (2015). A study on the hierarchical instructional system design of software education by school system. *Journal of the Korea Association Education, 19*(4), 533-544.

Shin, S. Y., Cho, H. K., & Kim, M. R. (2013). A curriculum development on the robot literacy related with a mathematics and science curriculum for elementary and secondary school students. *Journal of The Korean Association of Information Education, 16*(6), 55-70.

Son, K. H., & Son, W. S. (2014). The development and application to computer programming education using Arduino. *The Journal of Education, 34*(3), 169-179.

Song, U. S. (2013). Development of robot education program for pre-service elementary teachers using educational robot and its application. *Journal of Digital Contents Society, 14*(3), 333-341.

Song, U. S., & Gil, J. M. (2017). Development and application of software education program based on blended learning for improving computational thinking of pre-service elementary teachers. KIPS Tr. *Software and Data Eng, 6*(7), 353-360.

Talaee, E., & Noroozl, O. (2019). Re-conceptualization of "Digital Divide" among primary school; children in an Era of saturated access to technology. *International Electronic Journal of Elementary Education, 12*(1), 27-35.

Tedre, M., & Denning, P. J. (2016). The long quest for computational thinking. In Proceedings of the 16th Koli Calling Conference on Computing Education Research (Koli, Finland, Nov. 24–27, 2016), 120–129.

Tsai, M. J., Wang, C. Y., & Hsu. P. F. (2019). Developing the computer programming self-efficacy scale for computer literacy education. *Journal of Education Computing Research, 56*(8), 1345-1360.

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends, 60*(6), 565-568.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127-147.

Wiggns, G., & McTighe, J. (2005). *Understanding by design.* ASCD.

Wing, J. M. (2006). Computational thinking. *Communications of the Association for Computing Machinery, 19*(3), 33-35.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society, 366*, 317-3725.