Universal Journal of Educational Research 7(5A): 129-137, 2019 http://www.hrpub.org
DOI: 10.13189/ujer.2019.071514

The Effectiveness of an Unplugged Coding Education System that Enables Coding Education without Computers

Jeong Beom Song

Chungcheongnamdo Office of Education Research and Information Institute, Republic of Korea

Abstract An unplugged block coding system that can be used to code without a computer was developed in 2014 with the support of the Korea Research Foundation. This unplugged block coding system is a type of tangible programming that can be used to code by combining unplugged blocks that contain a small board or sensor mounted in the block of a flowchart's command symbol. In this study, we set the dependent variables as computational thinking skill and interest in software education, and we attempted to verify the effectiveness of the unplugged block coding system. The study method was a nonequivalent pre-test-post-test experimental design. During the course of the study, the experimental group and the comparison group performed programming activities to control the movement of a car the groups had in common. In the experimental group, we conducted the tangible programming activity, which controls the movement of the car by using the unplugged block coding system. The comparison group used Educational Programming Language programming activities to control the motion of the car using Scratch 2.0. As a result, it was confirmed that the experimental group showed a significant improvement in computational thinking skill and interest in Software education compared to the comparison group.

Keywords Unplugged Block Coding System, Computational Thinking Skill, Tangible Programming Tool, Coding Education Tool Capable of Computerless Coding Education, Software Education

1. Introduction

Computational thinking is recognized as a necessary skill for everyone [1], and in Korea it refers to coding education as an effective method for developing computational thinking skills [2]. This is because current coding education consists of analyzing problems, organizing them logically, and solving problems in cooperation with peers through algorithmic thinking beyond the existing programming. It will be called 'software education' in Korea, and it will be applied to the middle school level in 2018 and as a regular education course in elementary school in 2019[3]. Therefore, elementary school students will learn how to apply software in real life and how to think and apply problem-solving orders through procedural thinking [16]. The process of solving real-life problems is expressed in natural language, and they are solved logically via the process of expressing ideas through flowchart symbols. After that, students learn basic programming methods and tools (educational programming languages, such as Scratch) and simple program design to input data and output results. In addition, they must understand the structure of sequence, selection, and repetition in this class. All of the above content is expected to be taught over the course of 17 classes (40 minutes each in the first grade). Therefore, the burden of learning a large amount of content in a limited time may be significant. In order to reduce the learning burden for students, it is necessary to increase the transfer of learning across content. In this study, we attempted to verify the educational effectiveness of a system that can facilitate coding education without a computer by combining the block coding system with symbol-shaped instructions in order to efficiently support the transition from flowchart learning to programming education.

2. Related Research

2.1. Software Education Curriculum in Korean Elementary Schools

With the advent of the fourth industrial revolution era, advanced countries are very interested in software

competitiveness [4] [5]. As a result, Korea has applied software education to the curriculum since 2017 to enhance students' interest in software and computational thinking. The software education curriculum for grades 5 to 6 of Korea's elementary schools comprises one complete chapter covered over 17 class sessions. The contents of the software-education curriculum consist of the understanding of software, procedural problem solving and programming elements and structure.

Table 1. The Software Education Curriculum of Korean Elementary Schools

Domain	**Core Concepts**	**Content Element**
		5-6th grade in elementary school
Technical systems	Communication	• Understanding Software
		• Procedural Problem-solving
		• Elements and Structure of Programming

There are three elements of software education, summarized in Table 2 below, which are organized into five achievement standards so as to aid the concrete understanding of the content elements [6][17]. These achievement standards are also summarized in Table 2. To meet standard #2 for example, students are instructed in expressing in words the steps of procedural problem solving, after which they express the steps again using flowcharts. For standard #3, students learn about educational programming languages. As a vast amount of content must be covered within 17 course hours, the transition from standard #2 to standard #3 can be expected to present substantial challenges in implementation due to the changes in the learning tools and the content elements. Therefore, we believe that methodological considerations will be necessary in order to improve the transfer of learning during the transition between achievement standards.

Table 2. The Software Education Curriculum of Korean Elementary Schools

No.	**Content Element**	**Achievement Standard**
1	Understanding Software	Recognizing instances of the application of Software and understanding how these affect our lives
2	Procedural Problem-solving	Creating and applying the steps for problem solving according to procedural thinking
3	Elements and Structure of Programming	Experiencing the programming process by use of programming tools
4		Designing a simple program where data is inputted, the necessary operations are applied, and the results are printed
5		Understanding the structure of concatenation, selection, and repetition during the process of creating the problem-solving program

2.2. The Learning Tools Used for Software Education in Korea

Learning tools currently being used in Korea to teach elementary school children include unplugged coding education systems, educational programming languages, educational robots, and so on. Among those, the Scratch and Entry programming tools are used in educational programming languages [18]. These have the merit of being able to immediately carry out and verify commands, which aids in achieving competency in procedural problem solving regarded as important in software education in Korea. However, it has been brought up that such merit can be a hindrance to the consideration of procedural problem solving in students. In actual practice, according to studies, it was reported that beginners were seen to simply connect blocks of commands in sequence rather than undergo a consideration of the algorithms at play[7]. For example, in the case of Figure 1—changing the shape of the cat which is a "sprite", or character, in Scratch—the beginners would code as shown in Figure 2, whereas advanced students would code as shown in Figure 3. In addition, in Korea, the learning hierarchy for procedural problem solving of software education comprises, in descending order of merit: learning flowcharts, and using educational programming languages such as Scratch. In this case, there would be some pressure of additional learning needed for the transition from flowcharts to programming using Scratch. Furthermore, there are limits on software education curriculum in Korea, which is set at 17 course hours.

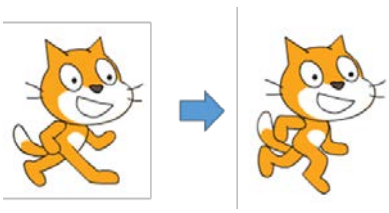


Figure 1. Animation of Scratch

Figure 2. Beginner's code type

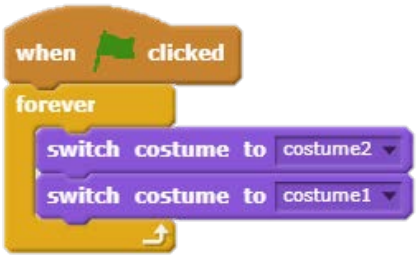


Figure 3. Experienced learners' code type

2.3. Tangible Programming Tool

These days, in Korea, Arduino boards like the one in Figure 4 are used pervasively as learning tools in coding education. Arduino has characteristics that can work in various Operating System environments and alongside a variety of software. Also, they have the merit of being relatively low-cost in comparison with other learning tools and have various shared libraries, to which many developers have contributed. However, they are difficult to manipulate for elementary school students and learning how to handle physical computing tools is time-consuming in comparison with programming processes. In this regard, they are difficult to apply to elementary school classes. The majority of the studies in Korea have been done on middle school students and older, in practice [8][9][10]. So the software education curriculum in Korean elementary schools has placed more stress on the implementation and verification of algorithms than on designing and assembling robots. The Tangible Programming Tool is the most typical example that contains those characteristics. It has features that can implement algorithms without using a computer. Little Bits from The Electronics Company is a representative example based on the Arduino such as shown in Figure 3. It can organize circuits easily by connecting blocks on the monitor screen. The following pictures show the composition of the components. Unlike the Arduino, it has merit in that it can be used readily even if users don't know circuitry or programming methods well. Also, according to recent studies, it is reported that Little Bits is of great use as a technologic software problem solving method in Maker education and STEAM programs [11]. However, the overall curriculum in Korea sets limits on software classes to 17 course hours. Besides, the learning hierarchy comprises clearly-defined standards, in descending order of merit: procedural problem solving in real-life examples, learning flowcharts, and how to code. Therefore, although Little Bits is an efficient learning tool in terms of enhancing creativity and problem-solving capability, it would not be suitable in that it can increase learning pressure for learners [20].

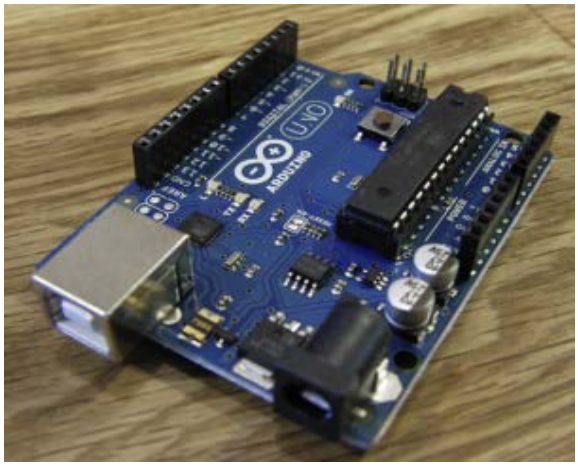


Figure 4. Arduino Uno

Figure 5. Composition of Little Bits

2.4. The Unplugged Coding Education System

The unplugged coding education system employed in this study is a tangible programming tool that can be used by beginners who have no prior experience with circuitry or programming methods. It was developed in 2014 with the support of the Ministry of Education and the National Research Foundation of Korea [19][12]. Designed to aid the transition from the learning of flowcharts to that of programming, this system comprises command blocks in the form of flowchart symbols, thus enabling anyone with experience in problem-solving through flowcharts to learn how to code. Photographs of the system's start / stop blocks, sensor inputs, control blocks for the LED, direction, speed, and angle units are depicted in Figure 6. Figure 7 depicts an example of how the blocks can be combined together. Figure 8 depicts sending a compiled program to an educational robot.

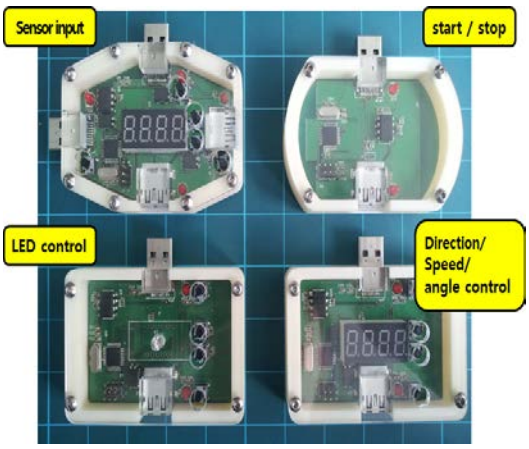


Figure 6. Actual View of Command Blocks

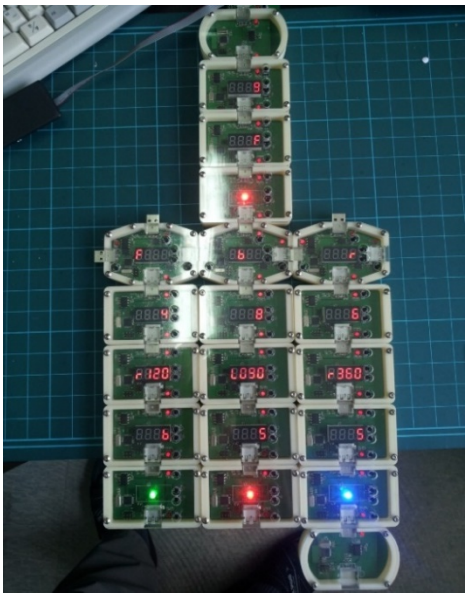


Figure 7. An Example of Combined Blocks

Figure 8. An Example of Combined Blocks

A detailed description of each command block shown in Figure 4 is as follows.

1) Start and Stop Programming Block

A USB 3.0 cable for communication with the robot is connected to the M-type connection terminal of the start block. The connection LED of the start / stop block always remains ON even if an operation is not performed. The stop block does not have a USB connection terminal and a connection LED on the bottom. Figure 9 shows the start and stop programming blocks.

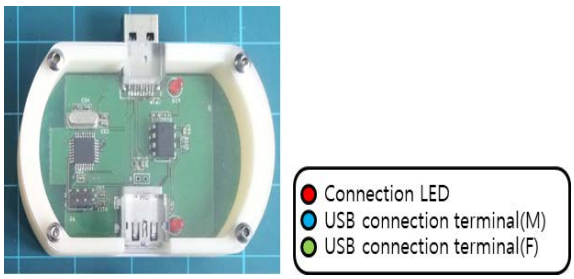


Figure 9. Beginning and Stop Programming Block

2) Sensor Input Programming Block

The USB connector on the side allows only the sensor input block of the same type to be connected. After the block is connected and power is supplied, the connection switch must be pressed to enable normal 485 communication between blocks. Sensor input setting is possible through the command switch button. It can set a control of front (F), back (B), left (L) and right (R). Two or more sensors can be set at the same time. Figure 10 shows the sensor input programming blocks.

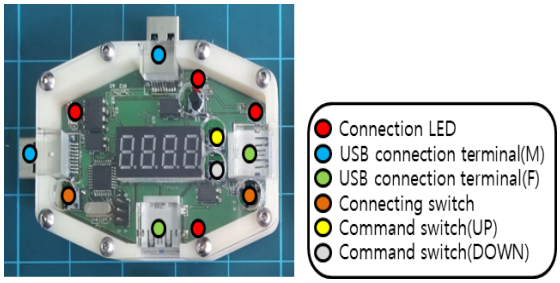


Figure 10. Sensor Input Programming Block

3) LED Control Programming Block

After the block is connected and power is supplied, the connection switch must be pressed to enable normal 485 communication between blocks. LED mode setting is possible through the command switch button. It can set up to six modes: red, red (blinking), green, green (blinking), blue, and blue (blinking). Figure 11 shows the LED control programming blocks.

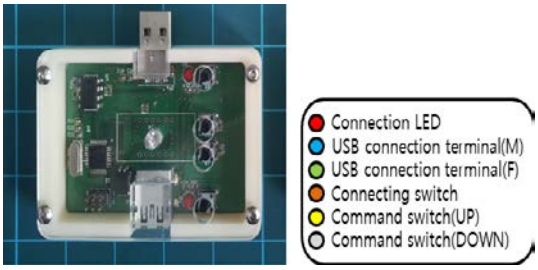


Figure 11. LED Control Programming Block

4) Direction, Speed, and Angle Control Programming Block

After the block is connected and power is supplied, the connection switch must be pressed to enable normal 485 communication between blocks. The shape of the block is the same, but it has three kinds of functions. The control of direction, speed, and angle are available through command switch buttons. Figure 12 shows the direction, speed, and angle control programming blocks [21],[22].

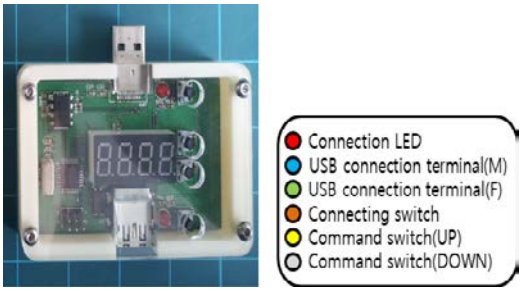


Figure 12. Direction, Speed, and Angle Control Programming Block

3. Methods

3.1. Hypothesis

For the purpose of ascertaining which of the two learning tools – programming activities using Scratch or tangible programming activities using the unplugged coding education system – were more effective in improving the learners' computational thinking and interest in software education, we set the hypotheses for this study as follows:

H1: Tangible programming activities using the unplugged coding education system will be more effective in improving learners' computational thinking compared to the use of the Scratch 2.0 tool.

H2: Tangible programming activities using the unplugged coding education system will lead to a higher degree of learner interest in Software education compared to the use of the Scratch 2.0 tool.

3.2. Samples

The population of this study's experiment was 215 students from grades 3 to 6 enrolled in five elementary schools in Chungcheongnam-do. Students were randomly selected from this population, with 30 students each assigned to the experimental and comparison groups. All 60 of the subjects of this study had no prior experience with tools such as Scratch or the unplugged coding education system. The experimental group comprised 17 male and 13 female students, while the comparison group comprised 18 male and 12 female students.

3.3. Process and Design of the Study

The two groups of subjects in this study received the same software education. The difference between them was the learning tool that was employed – the experimental group used the unplugged coding education system while the comparison group used the Scratch learning tool. Experimental treatments were applied at the same time over the same time period, and classes were led by separate teachers who were adept at the use of each learning tool. First, as none of the students had prior experience with Software education, they were subjected to two hours of prior education on procedural problem solving based on the Software education curriculum. By comparing the levels of computational thinking and interest in Software education between the groups, we confirmed that the two groups were homogeneous. Next, the experimental group was subjected to tangible programming education using the unplugged coding education system and the comparison group was subjected to six sessions of programming education using the Scratch tool. Subsequently, ex-post-tests were conducted in order to verify the between-group differences in computational thinking and interest in software education.

G_1	X_1	O_1	X_3	O_3
G_2	X_2	O_2	X_4	O_4

G_1: Experimental group, G2: Comparison group
X_1, X_2: Basic education of algorithms and programming
X_3: Tangible programming education using the unplugged coding education system
X_4: Programming education using Scratch
O_1, O_3: Pre-test
(computational thinking, interest in software education)
O_2, O_4: Post-test
(computational thinking, interest in software education)

Figure 13. Comparison between the two groups

3.4. Educational Program

In this study, the education program was designed and generally applied to the experimental group and the comparison group based on the algorithm and programming area of the 2015 software education curriculum as shown in Table 3. However, if there was a difference between the experimental and the comparison group, the educational tool used for programming education was different. On the other hand, the lesson contents were presented in order to create and revise a program that would reach the standards for software-education contents set by the Ministry of Education.

Table 3. The content of this study

No.	Subject	Learning Contents	Software-Education Curriculum
1	Algorithm and Programming	• Introduction of troubleshooting strategies and procedure	• Practice of problem solving process
2		• Introduction of flowcharts and programming tools and guide for programming procedure	• Practice of Algorithm
3	Sequence	• Understanding an algorithm for turning on a lightbulb based on an example	• Practice of Algorithm • Practice of programming
4		• Creating a program for turning on a lightbulb	
5		• Moving a vehicle in a predetermined order	
6	Conditional	• Understanding a conditional algorithm using a collision avoidance example	• Practice of Algorithm • Practice of programming
7		• Designing and programming a collision avoidance algorithm	
8		• Generating movement of various vehicles using conditional sentences	
9	Loop	• Learning iterative algorithms using an example with light, sound, and touch sensors	• Practice of Algorithm • Practice of programming
10		• Generating movement of vehicles using iterative sentences	
11		• Modifying movement of vehicles using iterative sentences	
12	General activity	• Understanding algorithms by seeing a line tracer example	• Practice of problem solving process • Practice of Algorithm • Practice of programming
13		• Creating a line tracer using a light sensor and a motor	
14			

Table 4. The content of this study

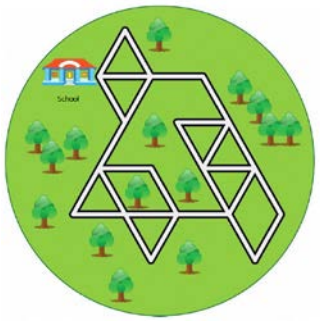
No.	Subject	Assessment Factors	Difficulty
1	Lighting Programming	Binary numbers	Easy
2	Beaver's dress code	Pattern recognition, decision trees	Easy
3	Water supply	Abstraction	Medium
4	Origami	Variable processing (functions)	Medium
5	Pirate games	Min Max algorithm	Hard
6	Park cleaning	Graphs	Hard

4. Measurement Instrument

4.1. Computational Thinking

We measured computational thinking using the Bebras Challenge Korea assessment sheet. This is a measurement tool that adapted the Bebras Challenge to suit the educational realities of Korea. The tool makes use of the concepts of discrete structure and algorithms, and it is widely used in diverse studies for the purpose of measuring computational thinking [13][14]. In this study, we employed two questions from each difficulty level (categories A, B, and C), which we then subjected to review by an expert in educational technology, an expert in primary education, and an expert in computer education. Table 4 shows the concrete contents of the Korean Bebras Challenge used in this study. Table 5 shows the concrete contents of the Korean Bebras Challenge used in this study.

Table 5. The content of this study

Task
Park Cleaning • Background Jihye is the school's park caretaker and is always trying to keep the promenade in the park clean. There are 27 walkways, each 10 meters long. A beaver living in the park plays with logs and leaves them on the road often. So Jihye departs from school every morning, checks all the walkways, and then returns to school. She wants to move the shortest distance to make this work easier and faster.  • Task At least how long should Jihye walk every morning? A) 270mB) 280m C) 300m D) 540m

4.2. Interest in Software Education

For the purpose of measuring interest in software education, in this study we used a tool developed by Shim et al. (2014) for measuring interest [15]. The instrument is coded in a 5-point Likert scale, comprising two questions that ask respondents about their degree of interest in computer-related curriculum and programming. Using Cronbach's α as a measure of internal consistency to establish overall reliability, the value of Cronbach's α was found to be 89.

5. Results

The homogeneity of the two subject groups was examined after they were administered classes on 'procedural problem-solving methods' through a pre-test. The results of the pre-test regarding computational thinking and interest in software education for both groups are summarized in Table 6.

As a result of the pre-test for computational thinking, the average scores of the experimental and comparison groups were, respectively, 3.51 and 3.47, indicating no statistically significant difference between the groups ($p>.05$). Likewise, the scores for interest in software education were 3.77 and 3.62, respectively, again indicating no statistically significant difference between the groups ($p>.05$). Therefore, we may consider that, in terms of computational thinking and interest in software education, the experimental group and comparison group are homogeneous.

Following the pre-test, in accordance with the research procedure, the experimental group was subjected to six sessions of tangible programming activities using the unplugged coding education system while the comparison group was subjected to programming education using the Scratch tool. After the experimental treatment, a post-test was conducted on the computational thinking and interest in software education of each group. The results of the post-test, by which hypotheses 1 and 2 of this study were verified, are summarized in Table 7.

Table 6. Pre-Test

Feature	Group	N	Ave	S.D	t	p
Computational Thinking	G_1	30	3.51	0.485	0.396	0.69
	G_2	30	3.47	0.313		
Interest in Software Education	G_1	30	3.77	0.489	1.188	0.24
	G_2	30	3.62	0.453		

Table 7. Post-test

Feature	Group	N	Ave	S.D	t	p
Computational Thinking	G_1	30	4.24	0.58	3.821	0
	G_2	30	3.7	0.51		
Interest in Software Education	G_1	30	4.38	0.64	3.968	0
	G_2	30	3.79	0.5		

First, looking at the results of the post-test for computational thinking, the average score of the experimental group was 4.24, compared to an average of 3.7 in the comparison group. This difference was found to be statistically significant ($p<.01$). Looking at the scores for interest in software education, the average score of the experimental group was 4.38, which indicates a significant improvement compared to the comparison group's average score of 3.79 ($p<.01$).

Therefore, hypotheses 1 and 2 of this study—that 'the use of the unplugged coding education system will be more effective in terms of improving computational thinking and interest in software education compared to the use of the Scratch tool'—were supported.

6. Discussion and Conclusions

In this study, we have sought to consider alternatives for the current issue of software education in Korean elementary schools, where the content is covered over 17 class sessions and is in particular limited to 6th-grade students. As one alternative, here we considered enhancing learning transfer by replacing the current transition from flowcharts to educational programming languages with a transition from flowcharts to an unplugged coding education system comprising blocks in the form of flowchart symbols, in addition to exploring the possibilities of this method. For this purpose, we conducted a study where the experimental group was subjected to the unplugged coding education system after learning flowcharts, while the comparison group was subjected to classes using the Scratch tool after learning flowcharts. The measurement instruments employed in this study were computational thinking and interest in software education [20] [21] [22].

The results of the tests conducted in this study indicate that, compared to the group of students who received classes using the Scratch tool after learning flowcharts, the experimental group that used the unplugged coding education system in the form of flowcharts showed significant improvements in terms of both computational thinking and interest in software education.

This result may be attributable to the particularity of the Korean software education curriculum, where the number of class sessions are limited and students are taught flowcharts before progressing to the learning of programming languages. Therefore, in order to generalize the findings of this study, the experiment must be applied to a more diverse range of subjects and situations.

REFERENCES

[1] Wing, J, "Computational Thinking," *Communications of the ACM*, vol. 49, no. 3, (2006), pp. 33-35.

[2] J. M. Lee, H. K. Park, and H. S. Choi, "Effects of SW Education Using Robots on Computational Thinking, Creativity, Academic Interest and Collaborative Skill," *JOURNAL OF The Korean Association of information Education*, vol. 22, no. 1, (2018), pp. 9-21.

[3] M. H. Jo, "Analysis of Elementary Pre-service Teachers' Experiences and Understanding of Software Education," *JOURNAL OF The Korean Association of information Education*, vol. 22, no. 1, (2018), pp. 81-89.

[4] K.B. Kim, "Developing an Intelligent Health Pre-diagnosis System for Korean Traditional Medicine Public User", *Journal of Information and Communication Convergence Engineering*, vol. 15, no. 2, pp. 85-90, Jun 2017.

[5] H. Seo, "High Performance Implementation of SGCM on High-End IoT Devices", *Journal of Information and Communication Convergence Engineering*, vol. 15, no. 4, pp. 212-216, Dec 2017.

[6] Ministry of Education of Korea, Practical Arts / Information Science Curriculum, Ministry of Education in Korea, (2015).

[7] Hussein, L. M. The Effectiveness of Teaching Educational Research Course on the Development of Scientific Research Skills, Academic and Personal Integrity among Female Students of Al-Qassim University. International Journal of Asian Social Science, 7(5), 392-409. (2017).

[8] Ihtiyaroglu, N., & Ates, Ö. T. Analyzing the Relationship between the Students' Stress-Coping Styles and School Attachment. Asian Journal of Education and Training, 4(4), 371-379. (2018).

[9] Iwuchukwu, G. C., & Iwuchukwu, R. N. Sociolinguistics and Language Education in Nigeria. Global Journal of Social Sciences Studies, 4(1), 13-22. (2018).

[10] Iwuchukwu, G. C., Ineji, P. U., & Inyang, E. Language, Communication, Poverty Eradication and the Fadama Projects in Nigeria. Global Journal of Social Sciences Studies, 4(1), 1-12. (2018).

[11] Jafari, S., Jafari, S., & Kafipour, R. Iranian Housewives Motives for English Language Learning from a Discursive Psychology Perspective. International Journal of English

Language and Literature Studies, 7(4), 138-149. (2018).

[12] Karaçam, A. Zest for Work in Physical Education and Sport Teachers' Perceptions of Success. Asian Journal of Education and Training, 4(4), 391-395. (2018).

[13] Khan, A., & Mohammad, J. Current Practices in Higher Education Institutes Pakistan and Gap Reduction between Industry and Academia: A Systematic Literature Review Approach. Asian Journal of Contemporary Education, 2(2), 173-181. (2018).

[14] Ahmed, U., Zin, M. L. M., & Majid, A. H. A. (2016). Impact of Intention and Technology Awareness on Transport Industry's E-service: Evidence from an Emerging Economy. Journal of Industrial Distribution & Business, Vol. 7, no. 3. pp. 13-18. (2016).

[15] S. H. Kim, "Analysis of Scratch code for Student Assessment about Computational Thinking Capability", *The Journal of Korean Association of Computer Education*, vol. 18, no. 5, (2015), pp. 25-34.

[16] J. K. Yoon and Y. S. Kim, " Influence of Programming Education Utilizing Arduino on Creative Problem Solving Ability of High School Students", *The SNU Journal of Education Research*, vol. 27, no. 3, (2018), pp. 53-73.

[17] S. M. Shin and C. H. Lee, " The Effects of Technology Education Using Arduino on Attitudes toward Technology of Middle School Students", *Journal of Korean practical arts education*, vol. 30, no. 4, (2017), pp. 221-240.

[18] S. Y. Shim, J. O. Kim and J. S. Kim, "Development of STEAM Learning Program Using Arduino to Improve Technological Problem-Solving Ability for Middle School Students", *THE KOREAN JOURNAL OF TECHNOLOGY EDUCATION*, vol. 16, no. 1, (2016), pp. 77-100.

[19] J. M. Tae, "Development of STEAM Education Program Utilizing Software Teaching Tool for Elementary School Pupils", *The Journal of the Korean Society for Gifted and Talented*, vol. 15, no. 2, (2016), pp. 121-147.

[20] J. B. Song and T. W. Lee, "Validation of the Unplugged Robot Education System Capable of Computerless Coding Education," *Journal of the Korea society of computer and information*, vol. 20, no. 6, (2015), pp. 151-159.

[21] Ali A, Haseeb M. Radio frequency identification (RFID) technology as a strategic tool towards higher performance of supply chain operations in textile and apparel industry of Malaysia. Uncertain Supply Chain Management. 2019;7(2):215-26.

[22] Jermsittiparsert K, Sriyakul T, Rodboonsong S. Power (Lessness) of the State in Globalisation Era: Empirical Proposals on Determination of Domestic Paddy Price in Thailand. Asian Social Science. Vol. 9, no. 17. 209. (2013).

[23] J. M. Lee, Y. J. Jung and H. K. Park, "Gender differences in computational thinking, creativity, and academic interest on elementary SW education," *Journal of The Korean Association of Information Education*, vol. 21, no. 4, (2017), pp. 381-391.

[24] J. Y. Noh and J. M. Lee, "The effects of SW education using robot on computational thinking," *Journal of The Korean Association of Information Education*, vol. 21, no. 3, (2017), pp. 285-296.

[25] K. H. Shim, S. W. Lee and T. W. Suh, "Development and evaluation of a STEAM curriculum utilizing arduino," *The Journal of Korean Association of Computer Education*, vol. 17, no. 4, (2014), pp. 23-32.