How do Students' Attitudes Towards Programming and Self-Efficacy in Programming Change in the Robotic Programming Process?

Osman Erol [i]
Burdur Mehmet Akif Ersoy University

Abstract

The aim of this study is to examine the effect of robotic design with Arduino on students' attitudes towards programming and on their perceptions of self-efficacy in programming. The study group consisted of 25 sophomore students attending the Department of Computer Education and Instructional Technologies in a state university located in the south of Turkey. The study lasted 12 weeks and the participants performed robotic design activities with Arduino throughout the process. Firstly, participants prepared a prototype and then programmed it for 8 weeks, and they created their own designs in the remaining 4 weeks. The Computer Programming Attitude Scale and Computer Programming Self-Efficacy Scale were utilized as the data collection tools in this pretest-posttest experimental study. The findings revealed that robotic design activities with Arduino significantly improved the participants' attitudes towards programming and programming self-efficacy. In addition, according to the participants' views, the factors that cause this improvement can be listed as activities' being enjoyable, facilitating and concretizing the process, being interesting and practical. Moreover, these robotic design activities were found to contribute to students' understanding of finding bugs and the logic of programming.

Keywords: Robotic, Arduino, Attitude Towards Programming, Programming Self-Efficacy

DOI: 10.29329/ijpe.2020. 268.2

---

[i] **Osman Erol,** Assist. Prof. Dr., Computer Education and Instructional Technology Department, Burdur Mehmet Akif Ersoy University, ORCID: 0000-0002-9920-5211

Correspondence: oerol@mehmetakif.edu.tr

INTRODUCTION

Programming is the most challenging course in computer science due to its complexity. In the programming language learning process, most processes and concepts remain abstract to students, and they have difficulty in concretizing the information they have learned (Bennedsen & Caspersen, 2008), which causes beginner students to perceive programming as a difficult course, and to fail (Başer, 2013). The factors that affect the programming performance are motivation, attitude and self-efficacy (Jenkins, 2002). Attitude is a psychological variable that drives human behavior (Anderson, 1988), and it is one of the key affective factors in learning (Maio & Haddock, 2009). Therefore, the attitude towards programming affects success at programming (Aşkar & Davenport, 2009; Tai, Yu, Lai & Lin, 2003). In the related literature, many studies show that negative perception, low motivation, and attitude negatively affect the success at programming (Anastasiadou & Karakos, 2011; Erol & Kurt, 2017; Korkmaz & Altun, 2013). In addition, another factor that influences the success of programming besides attitude is the self-efficacy belief. According to Bandura (1995), self-efficacy perception is defined as the personal judgment of the individual about the capacity to perform an activity necessary to show a certain performance. Individuals with high self-efficacy beliefs have higher expectations from their work and are more successful in dealing with any difficulties they may encounter (Akkoyunlu & Kurbanoglu, 2004; Compeau & Higgins, 1995; Karsten & Roth, 1998). Thus, self-efficacy perception of programming can be defined as the individual's judgment on the capacity to solve a problem by using a programming language. During programming learning, it is possible that students may fail in the programming course because of their low self-efficacy perception, that is, accepting programming as a difficult course from the beginning (Askar & Davenport, 2009). In addition, there is a positive correlation between attitude and self-efficacy in the literature (Demirtaş, Cömert & Özer, 2011; Kutluca & Ekici, 2010).

In order to solve all these problems in programming teaching, especially in teaching it to beginners, many visualization tools such as Code.org, Scratch, Small Basic, Alice, and Lego Mindstorm are used. The general purpose of these tools is to visualize the programming process, and make programming more understandable. In addition, these tools allow designing games and stories, which are also helpful in teaching programming (Schwartz, Stagner & Morrison, 2006; Lamb & Johnson, 2011; Lin & Liu, 2012). Recently, robotic kits and robotic design activities have been extensively used in teaching programming. Since students can actively create meaningful and original products in the robotic design studies, their motivation to learn increases and the learning process becomes more effective (Lin, Liu & Huang, 2012; Liu, Lin & Chang, 2010; Liu, Lin, Feng & Hou, 2013). Therefore, robotic design is an enjoyable, educational, and creativity-enhancing activity that may be used to develop students' programming and design skills (Gerecke &Wagner, 2007). Additionally, students are able to program the robots they develop themselves, and thus they can see the outcomes of the program they developed in a more concrete way. Educational robot design activities are based on Papert's constructionist approach. According to Papert (1980), students actively learn the best way when they design and construct meaningful products instead of being imparted the knowledge directly. In this way, learning occurs experientially while constructing products (Harel & Papert, 1991; Kafai, 2006; Mishra & Girod, 2006). Design activities offer different perspectives to the learners in the learning process, as it gives them the control over their own knowledge, instead of being passive recipients. According to Bustillo and Garaizar (2016), creating something may transform abstract concepts into concrete and well-understood concepts. Therefore, it can reduce the sense of uncertainty and complexity about the abstract programming concepts. Robot design is also related to design-based learning. In design-based learning, students take part in feasible problem scenarios, which usually involve a design process. In order to make these designs, the students do research and discover their skills during the process (Fortus, Dershimet, Krajcik, Marx, & Mamlok-Naaman, 2004; Ke, 2014). The designs required by the scenarios should be compatible with the content and be interesting for students. In addition, students develop high-level thinking skills like problem solving in the process of designing (de Vries, 2006; Ke, 2014). Fortus et al. (2004) propose a 5-step cyclic structure for the design-based learning process:

1. Identify and Define Context: Preparing projects according to course context and interests of students, and motivating them to act.

2. Background Research: Students conduct research and obtain information about the design, and the instructor provides the relevant concepts and skills.

3. Develop Personal and Group Ideas: Students propose solutions and ideas about their designs and share it within their group or with the entire class. Their peers and the instructor can also give advice.

4. Construct Artifacts: Students create new designs (artifacts) using the emerging ideas and combining their knowledge and skills.

5. Feedback: The students present their designs (artifacts). In this process, they receive feedback from their peers and the instructor.

In teaching programming by using robots, ready-to-use kits such as Lego Mindstorms and Mbot equipped with sensors, motors and programmable microcontrollers are used, which can be easily mounted on each other. In addition, these kits develop students' (K12 and younger) skills in the fields of Science, Technology, Mathematics and Engineering (STEM), and help them learn subjects in these areas (Benitti, 2012; Eguchi, 2010). Recently, instead of off-the-shelf kits, programmable microcontrollers such as Arduino have been more widely used in teaching programming. Since these electronic cards are open source, they are easy to program and popular in robotic applications. Arduino can be used to control devices like LEDs, buzzers, and motors by receiving data from input devices like sensors, and processing it through its micro controller. It can be used to create various types of computing products that interact with the surrounding environment (Jang, Lee & Kim, 2015). Unlike ready-made kits, the prototyping process (cable, motor, sensor connections etc.) takes longer and requires some technical knowledge. Still, a multitude of options are available for unlimited design. In the programming process, both syntax-based (Arduino IDE) and block-based compilers (Mblock, Scratch) can be used. In addition, robotics may give students the opportunity to learn about engineering and technology (Grubbs, 2013).

A review of the related literature reveals some studies conducted with engineering students by using Lego Mindstorms in teaching programming. Research shows that robotics instruction improves student motivation and attitudes towards programming, increases their success at programming, and also reduces the dropout rate in programming courses (Álvarez & Larrañaga, 2016; Korkmaz, 2016; Kurebayashi, Kamada & Kanamune, 2006; Liu, Newsom, Schunn & Shoop, 2013; Major, Kyriacou & Brereton, 2012). Davidson, Larzon and Ljunggren (2010) found that although not significantly improving self-efficacy, robotics instruction improves some sub-skills related to programming. In a study conducted with teacher candidates, the participants expressed positive opinions about the use of robotics in programming instruction, and their anxiety about learning programming in the design process was low (Şişman & Küçük, 2018). In another study, it was found that pre-service teachers' self-efficacy beliefs improved, and their level of content knowledge increased, and computational thinking was enhanced thanks to the use of robotics (Jaipal-Jamani & Angeli, 2017). In the studies conducted with Arduino, it was found that Arduino was useful in programming education and increased motivation and success; but the prototype (design) process was perceived as negative by students (Beug, 2012; Rubio, Hierro & Pablo, 2013). In general, both the use of ready-made kits and the use of Arduino have been found to be more effective and fun methods than the existing programming curriculum. In addition, robotics instruction is engaging and motivating in the programming process but it can be frightening because robots require mechanical installation.

The aim of the present study is to examine the effect of robotic design with Arduino on students' attitudes towards programming and their beliefs about self-efficacy in programming. Based on this objective, the study sought answers to the following research questions:

- Is there a significant difference between the pre-test and post-test scores reflecting the participants' attitudes towards programming and their beliefs in programming self-efficacy?

- What are the factors influencing the participants' attitudes towards programming and their beliefs in programming self-efficacy during robotic activities, as reported by the participants?

METHODOLOGY

The current study utilized the pretest-posttest experimental design without a control group. Using this design, the changes in the attitude and self-efficacy of the participants between the two measurement times were examined. The lack of a control group can be considered as a limitation of the study. In addition, the participants' general views on activities, the motivating and challenging factors (attitude & self-efficacy) were obtained by interview. The research design is presented in Table 1 below.

Table 1. Research design

Group	Pretest	Process	Posttest
Test	PA_{Pre} PSE_{Pre}	Robotic Activities with Arduino	PA_{Post} PSE_{Post} Interview

PA_{Pre} : Attitudes Towards Programming Pretest

PSE_{Pre} : Programming Self Efficacy Pretest

PA_{Post} : Attitudes Towards Programming Posttest

PSE_{Post} : Programming Self Efficacy Posttest

Participants

Participants of the research consisted of 25 sophomore students taking a Visual Programming course at the Department of Computer Education and Instructional Technologies in a state university located in the south of Turkey during the spring semester of the 2017-2018 academic year. Participant demographic characteristics (gender, type of high school graduated, and programming level) are presented in Table 2 below.

Table 2. Participant demographic characteristics

Demographic		F	%
Gender	Female	10	40
	Male	15	60
High School of Graduation	General High School	9	36
	Vocational/Technical High School	16	64
Programming Level	Low	10	40
	Medium	11	44
	High	4	16

Overall, 40% of the participants were female and 60% were male. A higher percentage of the students were graduates of vocational or technical high schools (64%) than general high schools (36%). Most of the participants had prior experience in programming. Furthermore, the participants had taken a Programming Course in the previous term. However, the majority of the participants had low and medium level of proficiency in programming, and only a few had a high level of proficiency.

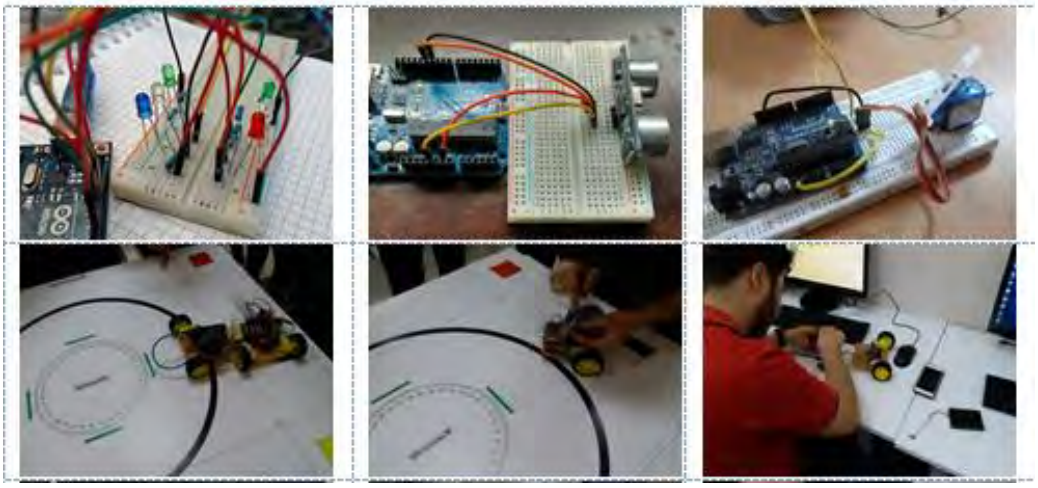
Data Collection Tools

The first data collection tool the study utilized was the *Computer Programming Attitude Scale*, developed by Başer (2013). The scale includes 38 items and four sub-dimensions as "self-confidence and motivation in programming," "the benefit of programming," "attitude towards success in programming," and "social perception of success in programming." In the 5-point Likert-type scale, each item is scored using values ranging between "1 - I Absolutely disagree" to "5- I Absolutely agree". Cronbach-α reliability coefficient of the scale was calculated as 0.88. The other data collection tool was the *Computer Programming Self-Efficacy Scale* developed by Ramalingam Wiedenbeck (1998) and adapted to Turkish by Altun and Nazman (2012). It is a 7-point Likert-type scale with nine items and two sub-dimensions named the "ability to perform simple programming tasks" and "ability to perform complex programming tasks." The Cronbach-α reliability coefficient of the scale was calculated as 0.85.

In addition, a semi-structured interview form was prepared by the researcher to obtain the participants' views on the activities. Using this form, the participants' general views on activities, the motivating and challenging factors, and the factors affecting their thinking about programming (attitude & self-efficacy) were examined. For these purposes, interview questions and probe questions were prepared and expert opinions were obtained from some experts in the field of programming. Interviews were conducted with all the participants at the end of the post-tests in the form of focus group interviews in groups of six people.

Procedure

Conducted as part of a Visual Programming course, the study took 12 weeks, including the testing process. The objective was to improve the participants' attitudes towards programming and their programming self-efficacy beliefs by using Arduino designing activities. In the first week, the participants were informed about Arduino and its basic components. In the following weeks, the participants performed the Arduino activities with the help of the instructor. During the process, the participants first created a prototype and then programmed it. During the last four weeks, the participants created and programmed their own design. They identified the project topics and conducted research before starting the design. In this process, the students were given continuous feedback by their instructor and classmates. At the end of the process, they transformed their ideas into design and created the product. The content and activities related to the course are presented in Table 3 and Figure 1.



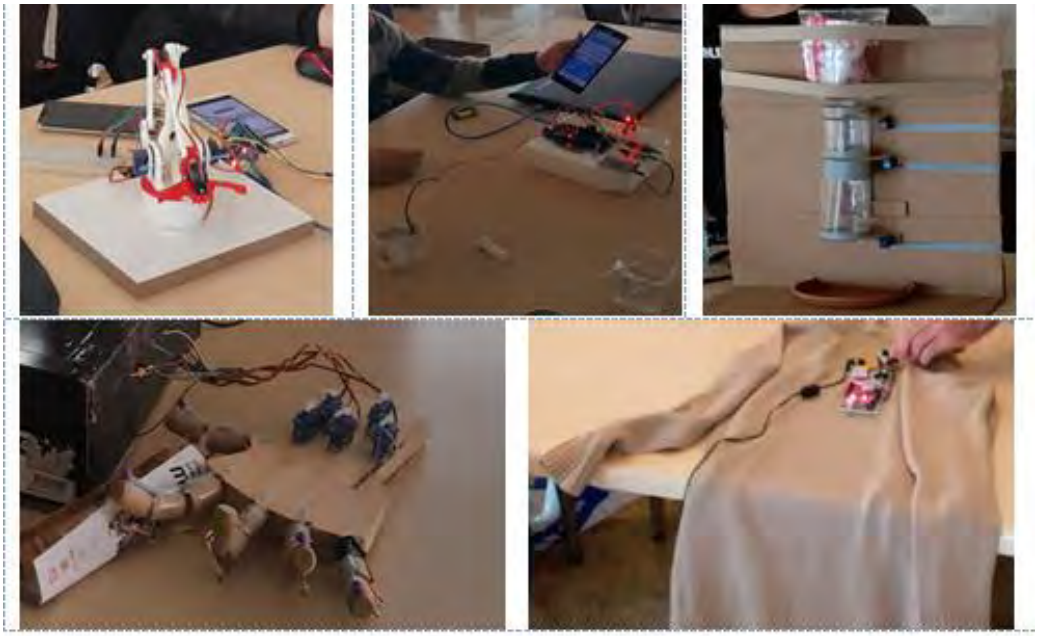


Fig. 1. Arduino activities and designs

Table 3. Course content

Week	Activities
Weeks 1&2	Introducing the Arduino and Presenting Information about the Basic Components
Weeks 3-6	Arduino Design Activities with Components (led, button, buzzer, distance sensor, heat sensor, light sensor, display, motors etc.)
Weeks 7-8	Robotic Design (obstacle-avoiding robot, line-following robot)
Weeks 9-12	Creating individual designs

Data analysis

In the study, MANOVA was conducted to measure whether the change in the participants' attitudes towards programming and self-efficacy in programming scores by time (pretest-posttest) was significant. The assumptions such as normality, multivariate normality, linearity, homogeneity, and multiple linear correlation were also tested before MANOVA was performed. Then, posthoc tests were conducted to examine the change in each variable (attitude, self-efficacy) and sub-factors of variables from pre-test to post-test. In addition, where more than one comparison is needed to minimize the Type-I error probability, the Bonferroni correction was applied, and the significance level was divided by the number of tests. Furthermore, to determine the magnitude of the difference, η^2 effect size value was determined. The Eta square correlation coefficient was interpreted as 0.01, 0.06, and 0.14, as small, medium, and large effect sizes, respectively (Cohen, 1988). In cases where there was no difference, statistical power was reported. In addition, content analysis was performed to analyze the qualitative data collected by the focus group interview. This process included the coding of the data, finding the themes, and organizing the codes and themes.

RESULTS

Table 4. MANOVA results for participants' attitudes towards programming & self-efficacy mean scores based on the time of measurement

Source of the variance	Variable	Type III Sum of Squares	Df	Mean Square	F	P	η2
Time	Attitude towards Programming	,245	1,000	,245	7,529	,011	,239
	Self Efficacy in Programming	2,090	1,000	2,090	5,583	,027	,189

*Wilks' $\Lambda = .638$, $F[2\text{-}23] = 6.523$, $\eta^2 = .362$, $p < .05$

The test of MANOVA was conducted in order to examine the effect of time on participants' attitudes towards programming and self-efficacy in programming. The Wilks' Λ result showed a significant effect of time on programming attitude and self-efficacy F(2,23) = 6.523, p < .05; Wilks'Λ = .638, partial η2 = .362. According to the Table, the participants' attitudes towards programming and self-efficacy in programming changed significantly. Also, the posthoc tests showed that both programming attitude and self-efficacy improved significantly in between the administration of the pre-test and the post-test (Fig 2).

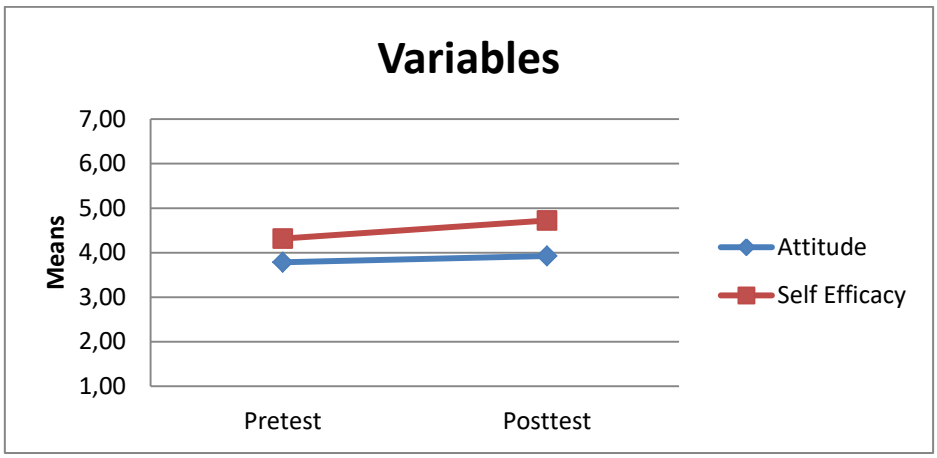


Fig. 2. The Means of Participants' Attitude towards Programming and Self Efficacy in Programming According to the Time of Measurement

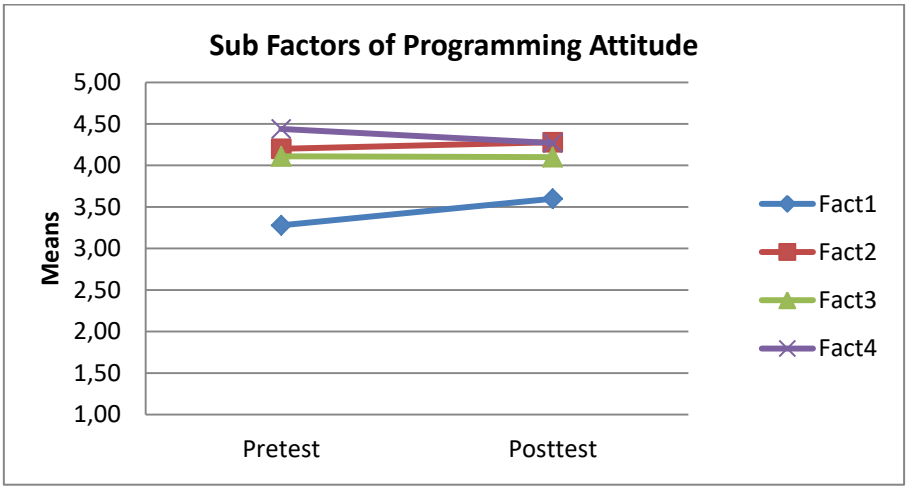
The change in each variable with its sub-factors according to time was also examined. The change of the participants' attitudes towards programming sub-factors according to time are shown in Table 5 below.

Table 5. MANOVA results for participants' attitudes towards programming sub factors mean scores based on the time of measurement

Source of the variance	Sub Factors	Type III Sum of Squares	Df	Mean Square	F	p	η2	Observed Power
Time	Self-confidence and motivation in programming	1,24	1,00	1,24	21,55	,00	,47	
	The benefit of programming	,08	1,00	,080	,73	,399	,030	,13
	Attitude towards success in programming	,02	1,00	,020	,11	,735	,005	,06
	Social perception of success in programming	,38	1,00	,37	1,68	,206	,066	,24

*Wilks' $\Lambda = .513$, $F[4\text{-}21] = 4.984$, $\eta^2 = .487$, $p < .05$

The test of MANOVA was conducted in order to examine the effect of time on participants' attitudes towards programming sub-factors. The Wilks' Λ result showed a significant effect of time on the sub-dimensions of programming attitude F(4,21)= 4.984, p < .05; Wilks'Λ=.513, partial η2=.487. According to the Table, the participants' self-confidence and motivation in programming changed significantly over time. Additionally, the post hoc tests showed that self-confidence and motivation in programming increased significantly from the pre-test to the post-test (Fig 3). Although not significant, the mean score for the "the benefit of programming" sub-factor increased, but the mean scores for the "attitude towards success in programming" sub-factor and the "social perception of success in programming" sub-factor slightly decreased. The change in the programming self-efficacy sub-factors according to time is shown in Table 6.



Fact 1 = Self-confidence and motivation in programming
Fact 2 = The benefit of programming
Fact 3 = Attitude towards success in programming
Fact 4 = Social perception of success in programming

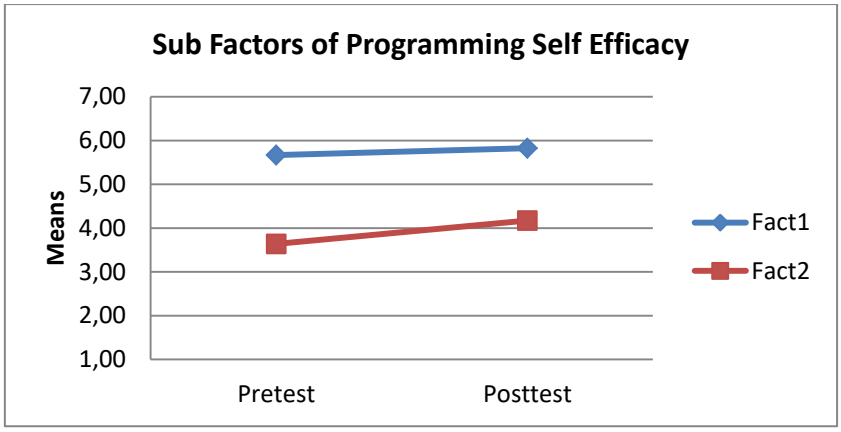
Fig. 3. The Means of Participants' Attitudes towards Programming Sub Factors According to the Time of Measurement

Table 6. MANOVA results for programming self-efficacy sub factors mean scores based on the Time of Measurement

Source of the variance	Sub Factors	Type III Sum of Squares	df	Mean Square	F	p	η2	Observed Power
Time	Ability to perform simple programming tasks	,320	1,00	,320	,548	,466	,022	,110
	Ability to perform complex programming tasks	3,556	1,00	3,556	5,818	,020	,195	,639

*Wilks' Λ =.801, F[2-23] = 2.865 , η^2 = .199, p < .05

The test of MANOVA was conducted in order to examine the effect of time on programming self-efficacy sub-factors. The Wilks' Λ result showed a significant effect of time on sub-dimensions of programming self-efficacy F(2,23) = 2.865, p < .05; Wilks'Λ = .801, partial η2 = .199. According to the Table, participants' ability to perform complex programming tasks changed significantly over time. Additionally, the post hoc tests showed that their ability to perform complex programming tasks increased significantly from the pre-test to the post-test (Fig 4). Although not significant, the mean score for the "ability to perform complex programming tasks" sub-factor slightly increased.



Fact1: Ability to perform simple programming tasks
Fact2 : Ability to perform complex programming tasks

Fig. 4. The Means of Programming Self Efficacy Sub Factors According to the Time of Measurement

In addition, the participants' general views on activities, motivating and challenging factors and effects on their thinking about programming (attitude & self-efficacy) were examined. The data obtained are shown in the Tables as themes.

Table 7. Opinions about Arduino Robotic Activities

Positive	
	Enjoyable
	Efficient
	Facilitative
	Interesting
	Applied
Negative	
	Hard
	Insufficient
	Expensive
	Complicated

The general opinions of the participants about the activities were gathered under "positive" and "negative" themes. As can be seen in Table 7, on the positive side, the participants stated that the activities were "enjoyable","efficient","facilitator", "interesting" and "applied"; and on the negative side, they stated that the activities were "hard," insufficient", "expensive" and "complicated." Furthermore, the participants thought that "practicing", "being funny", "continuous testing", "creating a product","accomplishment", "wondering" and "concretization" were the motivating factors. On the contrary, they stated that "hardware malfunctioning problems", "cable connection problems", "lack of prior knowledge", "expensive equipment", "card connections problem" and "overcrowded working area" were the challenging factors (Table 8).

Table 8. Motivating and challenging factors

Motivating Factors	
	Practicing
	Being funny
	Continuous testing
	Creating a product
	Accomplishment
	Wondering
	Concretization

Challenging factors	
	Hardware malfunctioning problems
	Cable connection problems
	Lack of prior knowledge
	Expensive Equipment
	Card connections problem
	Overcrowded Working Area

When the participants' views about the effects of the activities on their attitudes towards programming were examined, "understanding the logic of programming ","making programming enjoyable", "simplifying programming","concretizing programming concepts", "increasing interest in programming", "improving analytical thinking" and "helping to find bugs easily" emerged as the positive factors, while "making programming difficult/complicated" and "having no effect on programming" were the negative factors reported (Table 9).

Table 9. Effects on attitudes towards programming & self-efficacy

Positive Effects	
	Understanding the logic of programming
	Making programming enjoyable
	Simplifying programming
	Concretizing programming concepts
	Increasing interest in programming
	Improving analytical thinking
	Helping to find bugs easily
Negative Effects	
	Making programming difficult/complicated
	Having no effect

DISCUSSION AND CONCLUSION

Attitude and perception of self-efficacy are two of the basic affective factors affecting the programming performance. Therefore, a negative attitude towards programming or perceived low efficacy in programming may cause failure (Bennedsen et al., 2008; Jenkins, 2002). Students' programming performance can be increased by using the methods and techniques to improve their attitudes towards programming and their perceived self-efficacy. Recently, robotic design activities have been increasingly included in teaching how to program. Thus, in this study, the effect of Arduino robotic activities on students' attitudes towards programming and their perceived self-efficacy were investigated. The findings reveal that Arduino robotic activities increase participants' attitudes towards programming and programming self-efficacy. In addition, in terms of sub-dimensions, the self-confidence and motivation in programming and the ability to perform complex programming were found to increase throughout the process. Supporting the literature, it was also found that robotic training improves the attitudes towards programming, increases motivation (Álvarez & Larrañaga, 2016; Korkmaz, 2016; Kurebayashi et al., 2006; Liu et al., 2013; Major et al., 2012), and improves complex programming competencies (Davidson et al., 2010). One of the most important reasons for this may be that the students are active throughout the learning process, and can make their own designs. According to Papert (1980), the best learning occurs when designing and constructing meaningful products because learning takes place through hands-on experience (Harel & Papert, 1991; Kafai, 2006; Mishra & Girod, 2006). In addition, according to the students participating in the current study, being active in the process and creating a product motivated them. Thus, creating a robot may have aroused their curiosity and made the complicated and boring programming process easier and more fun. All of these factors may have improved students' attitudes towards programming. The students think that the activities enable concretizing the concepts of programming, allow continuous testing and facilitate the process, which seems to have a positive effect on their perceptions of their programming self-efficacy. Further, related to design-based learning, such robotics activities may transform abstract concepts into tangible products, reducing the sense of uncertainty and complexity about the abstract programming concepts (Bustillo and Garaizar, 2016). In

addition, students can see the outcomes of the program in a more concrete way by developing and programming robots by themselves. All these may have enhanced the students' efficacy and improved their ability to perform complex programming tasks. Furthermore, the qualitative findings indicate that the activities in this study helped the students to find bugs easily, helped them understand the logic of programming, and improved their analytical thinking. However, cable and card connection problems, hardware problems, lack of prior knowledge, expensive equipment, and other challenges negatively affected their attitudes towards programming and their perceptions of programming self-efficacy. According to literature, prototype (designing) process is perceived negatively by students due to mechanical installation problems (Beug, 2012; Rubio et al., 2013).

All in all, robotics activities implemented with Arduino improve student attitudes towards programming and their perceived self-efficacy in programming. The positive factors that cause this improvement can be listed as activities' being enjoyable, facilitative, interesting, practical, and helpful to concretize the process. In addition, these activities contribute to students' improved comprehension of the programming logic and learning how to find bugs. However, students may feel challenged by some Arduino connection and cable connection problems. Based on the findings, the following suggestions could be made:

- Robotic design activities can be used to popularize programming,
- Simple robot kits without card and connection problems should be used to reduce the negative impact, and
- In teaching programming languages, learner-oriented learning environments could be designed to increase learner motivation.

REFERENCES

Akkoyunlu, B., & Kurbanoglu, S. (2004). A study on teachers' information literacy self-efficacy beliefs. *Hacettepe Üniversitesi Eğitim Fakültesi Dergisi*, 27, 11-20.

Altun, A., & Mazman, S. G. (2012). Developing computer programming self-efficacy scale. *Journal of Measurement and Evaluation in Education and Psychology*, 3, 2, 297–308.

Álvarez, A., & Larrañaga, M. (2016). Experiences incorporating Lego Mindstorms Robots in the basic programming syllabus: Lessons learned. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 117–129. DOI:10.1007/s10846-015-0202-6

Anderson, L. W. (1988). Attitudes and their measurement. Keeves, J. P. (Ed.). In *Educational research, methodology and measurement: An international handbook* (s.421-426). New York: Pergamon Press.

Anastasiadou, S.D., & Karakos, A.S. (2011). The beliefs of electrical and computer engineering students' regarding computer programming. *The International Journal of Technology, Knowledge and Society,* 7(1*),* 37-51.

Aşkar, P., & Davenport, D. (2009). An Investigation of Factors Related to Self-Efficacy for Java Programming Among Engineering Students, *The Turkish Online Journal of Educational Technology – TOJET January. 8*(1).

Bandura, A. (1995). *Self-efficacy in changing societies*. Cambridge university press.

Başer, M. (2013). Developing attitude scale toward computer programming. *The Journal of Academic Social Science Studies,* 6 (6), *199 – 215.* http://dx.doi.org/10.9761/JASSS1702

Benitti, F.B.V. (2012). Exploring the educational potential of robotics in schools: A systematic review. *Computers & Education*, *58*(3), 978-988. https://doi.org/10.1016/j.compedu.2011.10.006

Bennedsen, J., & Carpersen, M. E. (2008). Exposing the programming process. Bennedsen, J.,Carpersen, M. E., & Kolling, M. (Eds.). In *Reflection on the theory of programming: Methods and implementation* (pp.6-16). Springer Berlin Heidelberg New York

Beug, A. (2012). *Teaching introductory programming concepts: A comparison of Scratch and Arduino*. Unpublished Mastes Thesis, The Faculty of California Polytechnic State University, Obispo, San Luis.

Bustillo, J., & Garaizar, P. (2016). Using Scratch to foster creativity behind bars: Two positive experiences in jail. *Thinking Skills and Creativity*, 19, 60–72. https://doi.org/10.1016/j.tsc.2015.08.003

Cohen, J.W. (1988). *Statistical power analysis for the behavioral sciences* (2. Edition). Hillsdale, NJ: Lawrence Erlbaum Associates.

Compeau, D. R., & Higgins, C. A. (1995). Computer self-efficacy: Development of a measure and initial test. *MIS quarterly*, 189-211.

Davidson, K., Larzon, L., & Ljunggren, K. (2010). *Self-Efficacy in Programming among STS Students*. Technical Reports from Computer Science Education course of Upssala University

Demirtaş, H., Cömert, M., & Özer, N. (2011). Öğretmen adaylarının özyeterlik inançları ve öğretmenlik mesleğine ilişkin tutumları. *Eğitim ve Bilim*, *36*(159).

Eguchi, A. (2010). What is educational robotics? Theories behind it and practical implementation. In Gibson D., & Dodge B. (eds.), *Proceedings of Society for Information Technology & Teacher Education International Conference 2010* (pp. 4006-4014). Chesapeake, VA: AACE.

Erol, O., & Kurt, A. A. (2017). The effects of teaching programming with scratch on pre-service information technology teachers' motivation and achievement. *Computers in Human Behavior*, *77*, 11-18. https://doi.org/10.1016/j.chb.2017.08.017

Fortus, D., Krajcik, J., Dershimer, R. C., Marx, R. W., & Mamlok- Naamand, R. (2005). Design-based science and real world problem- solving. *International Journal of Science Education,* 27(7), 855–879.

Gerecke, U., & Wagner, B. (2007). The challenges and benefits of using robots in higher education. *Intelligent Automation and Soft Computing, 13*(1), 29–43.

Grubbs, M. (2013). Robotics intrigue middle school students and build STEM skills. *Technol Eng Teach*, 72(6), 12–16

Harel, I., & Papert, S. (1991). Software design as a learning environment. *Interactive Learning Environments*, 1(1), 1-30.

Jaipal-Jamani, K., & Angeli, C. (2017). Effect of robotics on elementary preservice teachers' self-efficacy, science learning, and computational thinking. *Journal of Science Education and Technology,* 26(2), 175-192.

Jang, Y., Lee, W., & Kim, J. (2015). Assessing the usefulness of object-based programming education using arduino. *Indian Journal of Science and Technology*, *8*(S1), 89-96.

Jenkins, T. (2002). *On the difficulty of learning to program*. Proceedings of 3rd annual conference of the LTSN-ICS, 53-58, Loughborough, United Kingdom.

Kafai, Y. B. (2006). Playing and making games for learning instructionist and constructionist perspectives for game studies. *Games and Culture*, 1(1), 36-40.

Karsten, R., & Roth, R. M. (1998). Computer self-efficacy: A practical indicator of student computer competency in introductory IS courses. *Informing Science*, *1*(3), 61-68.

Ke, F. (2014). An implementation of design-based learning through creating educational computer games: A case study on mathematics learning during design and computing. *Computers & Education,* 73, 26–39. https://doi.org/10.1016/j.compedu.2013.12.010

Korkmaz, O. (2016). The effect of Lego Mindstorms Ev3 based design activities on students' attitudes towards learning computer programming, self-efficacy beliefs and levels of academic achievement. *Baltic Journal of Modern Computing*, *4*(4), 994–1007. doi:10.22364/bjmc.2016.4.4.24

Korkmaz, Ö., & Altun, H. (2013). Engineering and ceit student's attitude towards learning computer programming. *The Journal of Academic Social Science Studies International Journal of Social Science,* 6(2*),* 1169-1185.

Kurebayashi, S., Kamada, T., & Kanemune, S. (2006). Learning computer programming with autonomous robots. In *International Conference on Informatics in Secondary Schools-Evolution and Perspectives* (pp. 138-149). Springer, Berlin, Heidelberg.

Kutluca, T., & Ekici, G. (2010). Examining teacher candidates' attitudes and self-efficacy perceptions towards the computer assisted education. *Hacettepe University Journal of Education*, *38*(38).

Lamb, A., & Johnson, L., (2011), Scratch: computer programming for 21st century learners. *Teacher Librarian, 38* (4), 64-68.

Lin, J. M.C., & Liu, S.F. (2012), An investigation into parent-child collaboration in learning computer programming. *Educational Technology & Society, 15* (1), 162–173.

Lin, C. H., Liu, E. Z. F., & Huang, Y. Y. (2012). Exploring parents' perceptions toward educational robots: Gender and socioeconomic difference. *British Journal of Educational Technology, 43*(1), E31-E34.

Liu, E. Z. F., Lin, C. H., & Chang, C. S. (2010). Student satisfaction and self-efficacy in a cooperative robotics course. *Social Behavior and Personality, 38*(8), 1135-1146.

Liu, E. Z-H., Lin, C-H., Feng, H-C., & Hou, H-T. (2013). An analysis of teacher-student interaction patterns in a robotics course for kindergarten children: A pilot study. *The Turkish Online Journal of Educational Technology,* 12(1), 9-18.

Liu, A., Newsom, J., Schunn, C. & Shoop, R. (2013). Students learn programming faster through robotic simulation. *Tech Directions*, *72*(8), 16–19.

Maio, G., & Haddock, G. (2009). *The psychology of attitudes and attitude change*. SAGE Publications Limited.

Major, L., Kyriacou, T., & Brereton, O. P. (2012). Systematic literature review: teaching novices programming using robots. *IET Software*, *6*(6), 502. doi:10.1049/iet-sen.2011.0125

Mishra, P., & Girod, M. (2006). Designing learning through learning to design. *The High School Journal*, 90(1), 44e51

Papert, S. (1980). *Mindstorms: Children, computers and powerful ideas*. New York: Basic Books.

Ramalingam V., & Wiedenbeck S. (1998). Development and validation of scores on a computer programming self efficacy scale and group analyses of novice programmer self-efficacy. *Journal of Educational Computing Research*, 19, 4, 365–379.

Rubio, M. A., Hierro, C. M., & Pablo, A. P. D. Y. (2013). Using arduino to enhance computer programming courses in science and engineering. In *Proceedings of EDULEARN13 conference* (pp. 1-3).

Schwartz, J., Stagner, J., & Morrison, W. (2006). Kid's programming language (KPL). In *ACM SIGGRAPH 2006 Educators program* (p. 52). ACM.

Şişman, B., & Küçük, S. (2018). Pre-Service Teachers' Flow, Anxiety And Cognitive Load Levels In Robotics Programming. *Eğitim Teknolojisi Kuram ve Uygulama*, *8*(2), 125-156.

Tai, D.W.S., Yu, C.H. Laive, L.C. & Lin, S.J. (2003). A study on the effects of spatialability in promoting the logical thinking abilities of students with regard to programming language. *World Transactions on Engineering and Technology Education, 2(2)*, 251-254.

de Vries, E. (2006). Students'construction of external representations in design-based learning situations. *Learning and Instruction*, 16, 213–227.