Fall 2020

Learning Programming Through Robots: A Mixed-Methods Study on the Effects of Educational Robotics on Programming Comprehension and Motivation of Preservice Teachers

Alex Geoffrey Fegely

LEARNING PROGRAMMING THROUGH ROBOTS: A MIXED-METHODS STUDY ON THE EFFECTS OF EDUCATIONAL ROBOTICS ON PROGRAMMING COMPREHENSION AND MOTIVATION OF PRESERVICE TEACHERS

by

Alex Geoffrey Fegely

Bachelor of Science
Temple University, 2012

Master of Education
Coastal Carolina University, 2014

Education Specialist
Coastal Carolina University, 2017

Submitted in Partial Fulfillment of the Requirements

For the Degree of Doctor of Education in

Curriculum and Instruction

College of Education

University of South Carolina

2020

Accepted by:

Hengtao Tang, Major Professor

Ismahan Arslan-Ari, Committee Member

Lucas Vasconcelos, Committee Member

Anna Clifford, Committee Member

Cheryl L. Addy, Vice Provost and Dean of the Graduate School

DEDICATION

This dissertation is for my wife. Thank you for caring for me and sending me Uber Eats when you were at work, and I was not eating during all-day writing sessions. This dissertation is also for my parents. Thank you for your support with my move to South Carolina and my academic career. I would not be here without everything you have done for me.

ACKNOWLEDGEMENTS

Thank you to my chair, Dr. Hengtao Tang, for your patience, guidance, and insights throughout this process. Thank you for taking me on this summer and being my mentor throughout this process. I have learned so much from you. Thank you to my committee members, Dr. Ismahan Arslan-Ari, Dr. Lucas Vasconcelos, and Dr. Anna Clifford, for dedicating your time to make this the best possible dissertation through your recommendations and critical analyses. Thank you to Dr. Fatih Ari who helped guide this study in its earliest days. Thank you to Dr. Tammi Kolski and Dr. William Morris for your support. Finally, thank you to Dr. Michael Grant for your confidence in me.

ABSTRACT

The purpose of this action research was to evaluate the effect educational robotics have on the programming comprehension and motivation of preservice teachers. Computer science is increasingly being integrated into K-8 curricula across the country. However, there are few teachers trained to teach basic computer science concepts. Core subject teachers are being asked to shoulder the load of integrating computer science concepts into their instruction. Educational robotics have gained attention for their potential to aid users with comprehension and motivation while learning to program. This convergent parallel mixed methods research thus investigated (1) the effect of educational robotics on preservice teachers' comprehension of programming concepts, and (2) how and to what extent that educational robotics influence preservice teachers' motivation related to programming. This study utilized educational robotics to teach preservice teachers (*N* = 18) programming. Data were obtained through a pretest/posttest Programming Comprehension Assessment, a pre/post Programming Motivation Survey, individual interviews, and field notes. Paired sample *t*-tests, Wilcoxon signed-ranks tests, and inductive analysis were used to analyze the data. Quantitative data exhibited significant score increases from pretest to posttest, and significant motivation increases from pre-survey to post-survey. Qualitative data revealed five themes; (1) participants perceived that a problem-based robotics curriculum improved their intrinsic motivation toward programming, (2) participants agreed that knowing programming as a skill had advantages as a teacher, (3) participants experienced self-determination towards programming in the face of robotics challenges, (4) participants perceived that the gradually increasing level of difficulty in the robotics curriculum improved their self-efficacy about programming from initially low levels, and (5) participants perceived programming as a viable fit in their future classrooms. The findings of this study indicate that preservice teachers’ comprehension of programming concepts and motivation related to programming can be improved through educational robotics. This research has implications for informing preservice teacher educators integrating programming concepts into their instruction. Recommendations are provided for programming curriculum design.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

National Context

Computer science is being integrated into K-8 curricula at an increasing rate nationally (Burke, Schep, & Dalton, 2016; Wilson, Sudol, Stephenson, & Stehlik, 2010). However, the number of teachers trained to teach basic computer science concepts from kindergarten to 12th grade in America's public-school system is low (Burke et al., 2016; Google Inc. & Gallup Inc., 2016; Mannila et al., 2014; Wilson et al., 2010). The nation's shortage of teachers knowledgeable in computer science concepts is bottlenecking our country's economy and stunting the economic potential of America's youth (Burke et al., 2016; Wilson et al., 2010). As of 2018, there are more than half a million unfilled computing jobs in the United States (United States Department of Labor, 2018). Meanwhile, computer science majors earn the second-highest initial salary among college graduates (National Association of Colleges and Employers, 2018). Consequently, 91% of parents want their children to learn computer science while even more – 93% – want their children's school to teach computer science (Google Inc. & Gallup Inc., 2016). According to a survey by Google Inc. and Gallup Inc. (2016) in which over 12,000 principals and superintendents were polled, only 40% of elementary principals and 59% of middle school principals offered at least one computer science course in their school. In the same study, 73% of principals and 71% of superintendents either strongly agreed or agreed that computer science education should be integrated into the core subjects to alleviate this problem (Google Inc. & Gallup Inc., 2016).

Wilson et al. (2010) pointed to two primary reasons why even America's youngest and most tech-savvy teachers do not meet student, parent, and economic demands for computer science instruction in the classroom: unpreparedness and apprehension. In their report, Wilson et al. (2010) detailed an ominous national climate in which "very few pre-service teacher preparation programs have the current capacity or coursework developed to prepare computer science teachers" (p. 12). Although few preservice education programs around the country prepare teachers to implement computer science concepts in their teaching (Wilson et al., 2010), a lack of opportunities for preservice teachers to learn effective computer science pedagogy is not the only obstacle facing the nation (Israel, Pearson, Tapia, Wherfel, & Reese, 2015). The national dearth of teachers with computer science competency is often attributed to a pervasive impression of intimidation among teachers vis-a-vis learning and teaching unfamiliar computing concepts (Curzon, Cutts, & Bell, 2009; Grover & Pea, 2013; Meerbaum-Salant, Armoni, & Ben-Ari, 2013). Teachers experience anxiety developing and performing instruction on unfamiliar computer topics in front of their classes (Curzon et al., 2009; Grover & Pea, 2013). Teachers' lack of confidence parallels with low self-efficacy and motivation (Sandholtz & Ringstaff, 2014) and negatively impacts teachers' effectiveness (Babaei & Abednia, 2016; Kreijns, Van Acker, Marjan, & Van Buuren, 2013; Bandura, 1997; Paraskeva, Bouta, & Papagianni, 2008). Thus, Israel et al. (2015) noted that teachers of younger students might erroneously feel that computer science can only be taught through high-level computer programming languages like C++ or Java.

Due to the intimidating reputation of computer science, teachers may be less likely to implement any programming instruction in younger students' courses at all, denying students the chance to develop their knowledge of programming languages and computer science (Israel et al., 2015).

With America's lack of a formal plan for training teachers in computer science (Burke et al., 2016; Google Inc. & Gallup Inc., 2016), researchers have suggested remedies to make learning programming less intimidating (Sengupta, Kinnebrew, Basu, Biswas, & Clark, 2013; Fessakis, Gouli, & Mavroudi, 2013; Good, 2011). A study by Sengupta et al. (2013) showed that in-service teachers who initially demonstrated apprehension about learning computer programming found basic block-based programming languages to be valuable. Other studies have shown that teachers' positive self-efficacy on technology concepts correlates to improved instructional practices with technology (Ertmer & Ottenbreit-Leftwich, 2010; Ertmer, Ottenbreit-Leftwich, Sadik, Sendurur, & Sendurur, 2012). According to Good (2011), less difficult block-based programming languages designed to "lower the computational floor" (p. 18) can be used to build novice programmers' motivation and self-efficacy with programming (Fessakis et al., 2013). Therefore, block-based programming languages can be leveraged to cut through preservice teachers' initial apprehension of computer science concepts before they enter the field, thus cultivating teachers that are more competent with computer science.

Papert (1980) published the seminal research on programming instruction with his Logo programming language and on-screen turtle drawing activities. Since then, the pairing of basic programming languages and robotics have become more prominent in

America’s schools, with the toy brick company Lego advancing to the forefront of public prominence (Martin, Mikhak, Resnick, Silverman, & Berb, 2000; Martin et al., 2011; Martin & Resnick, 1993). Dodds, Greenwald, Howard, Tejada, and Weinberg (2006) reported that “A key advantage of the most popular platforms,” such as Lego, “is the variety of ways in which students can program them” (p. 12). Thus, robotics kits flip students’ typical experience of learning how to operate technologies into learning how to create technologies (Burke & Kafai, 2014; Casler-Failing, 2017). Numerous studies have demonstrated that students as young as four can construct robots from kits and program the robots to perform simple tasks (Bers, 2008; Bers, Ponte, Juelich, Viera, & Schenker, 2002; Cejka, Rogers, & Portsmore, 2006; Kazakoff, Sullivan, & Bers, 2013; Strawhacker & Bers, 2015) while studies on preservice teachers have suggested positive results related to robotics, programming comprehension, and motivation (Jaipal-Jamani & Angeli, 2017; Kim et al., 2015; Kucuk & Sisman, 2018; Ortiz, Bos, & Smith, 2015). It can be inferred from these noteworthy studies that educational robotics can provide a promising method for both teaching programming and motivating preservice teachers to use programming.

Local Context

South Carolina released its K-8 Computer Science and Digital Literacy Standards in May of 2017 (South Carolina Department of Education, 2017). A survey of 158 K-12 South Carolina teachers by Burke et al. (2016) reported that the primary obstacles of teaching computer science in the state are a lack of time and dedicated computer science courses. With few schools offering dedicated computer science courses for K-8 students, non-computer science teachers have been asked to integrate computer science teaching into other subjects (Burke et al., 2016; Google Inc. & Gallup Inc., 2016). Thus, preservice teachers must be prepared to integrate content from The South Carolina Department of Education's (2017) K-8 Computer Science and Digital Literacy Standards such as "Standard 4: Develop a program to express an idea or address a problem" and "5.AP.4.1. Use a visual language to design and test a program that solves a simple task" (p. 23-32). The *Running on Empty* report (Wilson et al., 2010) implored federal, state, and local governments to "Create pre-service and professional development opportunities for computer science teachers" and "Expand professional development opportunities and recruit new computer science teachers" (p. 10). To date, however, South Carolina's Department of Education has not advanced formal guidelines for colleges to integrate these computer science standards into current preservice teacher education programs.

The South Carolina K-8 Computer Science and Digital Literacy Standards (South Carolina Department of Education, 2017) are currently being integrated into an undergraduate educational technology class at the university where this study takes place. I implemented a programming unit of instruction that utilized educational robotics. The aim of this unit was to both prepare K-8 preservice teachers to integrate programming into their instruction and motivate them to use programming through creative educational robotics programming activities.

Statement of the Problem

New K-8 Computer Science and Digital Literacy Standards have been introduced in the state of South Carolina (South Carolina Department of Education, 2017). With few K-8 schools offering stand-alone computer science courses, principals are relying upon teachers of other subjects to integrate computer science concepts into their classes (Burke et al., 2016; Google Inc. & Gallup Inc., 2016). Therefore, K-8 preservice teachers of non-computer science subjects must be prepared and motivated to integrate content from the South Carolina Department of Education's (2017) standards such as "develop a program to express an idea or address a problem" and "use a visual language to design and test a program that solves a simple task" into their instruction (p. 23-32). However, studies have shown that teachers can experience difficulties with traditional abstract methods of learning programming (Bower et al., 2017; Grover & Pea, 2013; Israel et al., 2015; Ortiz et al., 2015; Resnick et al., 2009). For these reasons, teachers need to be able to both comprehend programming concepts and be motivated to use and teach programming.

Purpose Statement

The purpose of this action research was to evaluate the effect educational robotics have on programming comprehension and motivation of preservice teachers at a medium-sized liberal arts university in the southeastern United States.

Research Questions

1. What is the effect of educational robotics on preservice teachers' comprehension of programming concepts?
2. How and to what extent does educational robotics influence preservice teachers' motivation related to programming?

Researcher Subjectivities & Positionality

Peshkin (1988) explained that a researcher's subjectivities "have the capacity to filter, skew, shape, block, transform, construe, and misconstrue what transpires from the outset of a research project to its culmination in a written statement" (p. 17). By outlining my positionality and subjectivities before delving into my research, I can assess the assumptions I have about my participants and what perceptions I believe my participants will have about me. From this reflection, I can understand how subjectivity and positionality principles influence this study.

I am a lecturer and instructional technology specialist within the research location. I have experience with educational technology as a former K-12 public-school student and later, as a high school teacher and college instructor. While a student, I was motivated by using technology for as many projects as I could, creating podcasts, educational videos, and other technology-focused projects. During graduate school, I worked as a web developer and graphic designer. As a teacher at a STEM high school, I found it rewarding to integrate my students' interests in engineering and computer science with social studies class content by including programming and 3D modeling assignments. Currently, I have experience with educational technology as a doctoral student, college instructor, and instructional technology specialist. I have also co-directed a grant that taught middle school and high school science and math teachers in a low socioeconomic school system on how to integrate programming and robotics concepts into their instruction. From these experiences, I have solidified the belief that educational technology is an integral part of K-12 and college education. In my judgment, to fully prepare our students for the future economy, computer science concepts must be integrated into school curricula at the earliest opportunity.

An adage states that you are not who you are, nor are you who you think you are. You are, in fact, who you think others think you are. Action research is a collaboration between the researcher and participants (Creswell, 2014; Mertler, 2017). The researcher and the participants work closely together; therefore, it is paramount to understand participants' perceptions of the researcher in order to see the study with a more authentic view.

My positionality in this study is best described by Dwyer and Buckle (2009) as that of an "insider-outsider" from "the space between" (p. 60). I perceive my participants to mainly assign my status to be that of an insider, which Dwyer and Buckle (2009) explained as "sharing the characteristic, role, or experience under study with the participants" (p. 55). As a former education major and teacher, I share my participants' background, life calling, and ideology. As an alumnus of this study's research location, I share many of the same experiences as my participants both inside and outside of school. As a university lecturer who teaches my participants every day, I am an insider with them through our shared experience of my class. I come from a middle-class family, as many of my students do. Although I may not share the exact same experiences as all my students, I feel as though I come from a background similar enough to empathize and relate. However, I realize that my participants may ascribe my status to be that of an outsider because I hold grading power over them. In addition, I am much older than they are, and I am not currently an undergraduate student sitting with them in class. Due to my shared background with my students as an insider and my outsider power position within the study, I cannot be one or the other (Dwyer & Buckle, 2009). Instead, I am a hybrid insider-outsider.

Being an insider-outsider for my study is a double-edged sword. Dwyer and Buckle (2009) noted that insiders enjoy quick and more open acceptance into the participant population than do outsiders. I identify with my participants' day-to-day lived experiences, and my participants may ascribe more trust to me than to an outsider. They may be more open and truthful with their responses, especially in the case of my individual interview qualitative data collection. On the negative side, as an insider, I may be inherently biased due to not being removed from the participant population (Dwyer & Buckle, 2009; Merriam et al., 2001). As Merriam et al. (2001) noted, as a partial insider I may not be "curious enough to raise provocative questions" (p. 411). Therefore, I must be conscientious about removing myself as much as necessary from my participants' standpoint and ask tough questions to exercise the perspective more commonly associated with an outsider.

As the researcher, I must establish how my interpretations are influenced by my personal value system (Mertens, 2009). My personal paradigm aligns with the pragmatist standpoint. As Hathcoat and Meixner (2015) have described, I will utilize a "plurality of methods to address valued aims of inquiry" in my study (p. 435). From my pragmatist view, my relationship with my participants will impact the results of my research. Corresponding to my insider-outsider role, pragmatists choose an appropriate depth of relationship with their participants relevant to the goals of the research (Mertens, 2009). Ontologically, my study will utilize the multiple viewpoints of my participants in quantitative and qualitative metrics to thoroughly understand the problem and present subsequent solutions (Frels & Onquegbuzie, 2013). To curtail my power influence over the participants, I will position myself within the action research study and classroom as an insider-outsider collaborator. I aim to present myself as a helpful scaffold for student learning as opposed to the traditional powerful teacher role in order to cultivate trust (Herr & Anderson, 2005). Considering my participants' diverse viewpoints, I must appropriately separate myself from my deep-seated beliefs that computer science concepts are relevant to K-8 students and can be creatively linked to most subject areas by teachers. I must take the stance closer to a participant wherein my life experience with educational technology and computer science has not yet crystallized in order to respect and value my participants' perceptions.

Definition of Terms

Block-based Programming

This study utilized Weintrop's (2016) definition to operationalize the term block-based programming. Weintrop's (2016) definition explains that block-based programming languages "leverage a programming-primitive-as-puzzle-piece metaphor" through on-screen programming environments in which users engage the language by "dragging blocks into a canvas and snapping them together to form scripts" to write an executable computer program (p. 58).

Career Motivation

This study used Arwood's (2004) characterization of career motivation. Arwood (2004) describes that career motivation is exhibited when learners understand the subject being learned as relevant to their future careers.

Educational Robotics

Educational robotics is a term used to identify versions of robotics designed for teaching or learning. Ortiz et al. (2015) provided the definition of educational robotics which will guide this study: "Educational robotics is a specific application of K–12 engineering education and offers students physical manipulatives that are familiar and easy to work with as they participate in the engineering design process" (p. 43).

Educational Robotics Practices

This study used Catlin's (2012) definition to operationalize educational robotics practices. Catlin (2012) characterizes educational robotics practices as an instructional strategy that uses educational robotics for instructional purposes.

Intrinsic Motivation

Intrinsic motivation was operationalized by Ryan and Deci's (2000) description of the term. Ryan and Deci (2000) define intrinsic motivation as a learner's desire to learn about a topic due to their inherent interest and "innate psychological needs for competence and autonomy" with the topic (p. 65).

Motivation

This study utilized Pintrich and Schunk's (1996) definition of motivation. Pintrich and Schunk (1996) operationalize the term motivation as "the process whereby goal-directed activity is instigated and sustained" (p. 4).

Motivation to Integrate Programming into Teaching

Motivation to Integrate Programming into Teaching (MTIPIT) was defined based on research on teacher motivation and its combination of intrinsic, extrinsic, and altruistic factors (Brookhart & Freeman, 1992; Han & Yin, 2016; Sinclair, 2008). This study operationalized the term based on Han and Yin's (2016) characterization of teacher motivation. In this study, MTIPIT is defined as the reasons an individual chooses to use and teach programming based on intrinsic and contextual factors.

Programming

Ceruzzi’s (1998) definition of computer programming was used to operationalize the term programming in this study. Computer programming is the process of designing and creating instructions for computers to perform specific tasks (Ceruzzi, 1998).

Programming Comprehension

Ala-Mutka’s (2004) definition of programming comprehension best aligns with the goals and instruments utilized in this study and will be used to operationalize the term programming comprehension. Ala-Mutka (2004) describes programming comprehension as the “ability to track code to build a mental model of the program and predict its behavior” (p. 5).

Robots

The robots used in this context are Lego EV3 educational robots running the EV3-G programming language that are developmentally appropriate for the K-8 learners that preservice teachers who participate in the study will have in the classroom (Martin et al., 2000; Martin et al., 2011; Martin & Resnick, 1993). The EV3 educational robotics kits are part of a Lego universe that “extends the traditional Lego bricks with a central control unit (the RCX), as well as motors and various kinds of sensors” (Koller & Kruijff, 2004, p. 1).

Self-determination

In this study, self-determination will be operationalized by Black and Deci’s (2000) definition of the term. Black and Deci (2000) define self-determination as the control learners have over their learning.

Self-efficacy

Bandura’s (1997) research on self-efficacy will be used in this study. Self-efficacy is defined in this study as learners’ confidence in their ability to achieve the learning task (Bandura, 1997).

CHAPTER 2

LITERATURE REVIEW

The purpose of this action research was to evaluate the effects educational robotics have on programming comprehension and motivation of preservice teachers at a medium-sized liberal arts university in the southeastern United States. This review of literature addresses two research questions. The research questions in this study are (1) what is the effect of educational robotics on preservice teachers' comprehension of programming concepts? and (2) how and to what extent does educational robotics influence preservice teachers' motivation related to programming?

In order to form a comprehensive foundation of knowledge on the topics of programming and educational robotics as they pertain to teacher education, four main paths of inquiry were formed to guide my literature search: (1) programming in K-12 education, (2) programming in teacher education, (3) educational robotics in K-12 teaching, and (4) educational robotics in teacher education. The search terms for each of these four paths of inquiry were varied, and database filters were utilized to identify full-text, peer-reviewed articles from academic journals that represented the most relevant and rigorous literature. The *ERIC* database was my most-used tool to identify pertinent articles for this literature review. A small amount of pertinent literature was found through searches on *Education Source* and *Google Scholar* that did not appear in the *ERIC* database. I also accessed *ProQuest Dissertations and Theses* to identify dissertations related to my research. Ancestral searches through the references of germane literature were used to strengthen the foundation of this literature review. *Google Scholar* and *ResearchGate* were used to access many of these ancestral studies not found on the educational research databases.

This resulting literature review is organized into four key sections, including (1) programming, (2) educational robotics, (3) impact of educational robotics on programming comprehension, and (4) impact of educational robotics on motivation related to programming. The first section overviews the literature on programming to provide the reader with a foundational understanding of programming and how it fits into education. The next section explains the use of educational robotics as learning tools for novices being introduced to programming. The final sections offer syntheses of studies involving programming and educational robotics. Special attention is paid to teacher education and what these studies found in relation to the impacts of educational robotics on programming comprehension and motivation.

Programming

Programming is a major construct identified in this study's research questions. In this section, programming and its associated aspects will first be defined. Next, block-based programming languages and the ways in which learners interact with such programming languages will be explained. Then, programming's context in education will be detailed. Finally, studies that uncovered difficulties experienced by in-service and preservice teachers while learning to program will be shared. These details on programming will provide readers with a foundational understanding of the central construct being evaluated in this study.

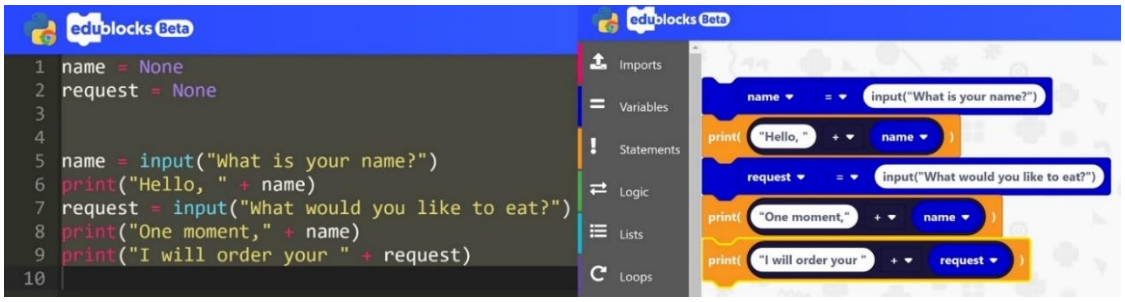
Defining Programming

Programming is a main construct in this study. At its root, Böhm and Jacopini (1966) have explained that programming "is where flow diagrams are introduced with different purposes and defined in connection with the descriptions of algorithms or programs" (p. 366). Ceruzzi, (1998), defined computer programming more broadly as the process of designing and creating instructions for computers to perform specific tasks, known as programs. Programs have also been described by Dijkstra (1976) as "algorithms intended for automatic execution on computers" (p. 8). Programs are created with programming notation techniques, commonly referred to as programming languages (Dijkstra, 1976). Programming includes processes of computational thinking, and misconceptions discussed in the literature note that teachers and students believe the two to be the same (Lu & Fletcher, 2009; Qualls & Sherrell, 2010). Yamazaki, Sakamoto, Honda, Washizaki, and Fukazawa (2015) proposes that "computational thinking is a common concept to various programming languages" (p. 157). Various definitions of computational thinking include aspects about how its processes are fundamental to programming, including problem-solving, concurrency, sequences, variable representation, loops, conditionals, calculation, and abstraction (Kafai & Burke, 2014; Sengupta et al., 2013; Yamasaki et al., 2015). Computational thinking has been described by Yadav, Good, Voogt, and Fisser (2017) as "decomposing problems, using algorithms to solve problems, and abstracting and automating the problem-solving approach" (p. 1051).

Block-based Programming

In this section, research detailing block-based programming's functions will be presented. This section will include descriptions of how users write programs in block-based programming. Then, the educational advantages of block-based programming exhibited in the literature will be described.

Writing block-based programs. There are educational versions of programming languages that offer varying scaffolds to novice programmers while they learn to write programs (Sáez-López, Román-González, & Vázquez-Cano, 2016; Weintrop, 2016; Weintrop & Wilensky, 2017). Block-based programming is a subset of programming languages that are part of the visual programming language family (Weintrop, 2016). Visual programming differs from more traditional text-based programming because visual programming allows learners to create programs in a multidimensional programming environment (Myers, 1990). Weintrop (2016) described block-based programming languages as those which "leverage a programming-primitive-as-puzzle-piece metaphor" through on-screen programming environments in which users engage the language by "dragging blocks into a canvas and snapping them together to form scripts" in order to write an executable computer program (p. 58). As shown in Figure 2.1, students assemble programs by dragging and dropping pictorial representations of programming commands in block-based environments (Sáez-López et al., 2016; Weintrop, 2016; Weintrop & Wilensky, 2017). Such blocks represent text-based programming staples like Boolean phrases, conditions, loops, and variables, among other



```
name = None
request = None


name = input("What is your name?")
print("Hello, " + name)
request = input("What would you like to eat?")
print("One moment," + name)
print("I will order your " + request)
```

Figure 2.1. Differences between text-based and block-based programming languages.

functions (Meerbaum et al., 2013; Weintrop, 2016). Scratch (e.g., Malan & Leitner, 2007; Meerbaum-Salant et al., 2013; Resnick et al., 2009), and Alice (e.g., Cooper, Dann, & Pausch, 2000; Kelleher, Pausch, & Kiesler, 2007; Meerbaum-Salant et al., 2013; Werner, Campe, & Denner, 2012) are two examples of block-based programming environments which have been widely studied in education and are categorized in a group known as structured editors (Donzeau-Gouge, Huet, Lang, & Kahn, 1984). Due to the unique language and editing environment characteristics described above, block-based programming languages are often used to introduce novices to programming.

Advantages of block-based programming. Different modalities have been indicated to make learning easier for different learners (Antle, 2007; Manches & Price, 2011; Scaife & Rogers, 2005; Weintrop & Wilensky, 2017). Common text-based programming languages have been reported to be challenging to learn because of the specific grammar and syntax requirements for each command (Alkaria & Alhassan, 2017; Falloon, 2016; Wilson & Moffat, 2010). Block-based programming languages remove the frustrating syntax and related errors likely to be encountered by novice programmers because the blocks have the grammar essential to programming languages built-in (Alkaria & Alhassan, 2017; Weintrop & Wilensky, 2015). With block-based programming environments, blocks of programming commands can only be connected if the sequences make sense and are functional (Alkaria & Alhassan, 2017; Falloon, 2016; Kim, Yuan, Vasconcelos, Shin, & Hill, 2018; Weintrop & Wilensky, 2015; Wilson & Moffat, 2010). Weintrop (2016) succinctly explained that "If two blocks cannot be joined to form a valid syntactic statement, the environment prevents them from snapping together, thus preventing syntax errors but retaining the practice of assembling programs" (p. 59). As blocks cannot be snapped together unless they work as chunks of commands, novice programmers can modify their program and correct their mistakes before running the program unsuccessfully.

With text-based programming's typical obstacles removed, block-based programming can help learners explore abstract computer science concepts sooner in their educational progression than learners using text-based programming languages (Bers, Flannery, Kazakoff, & Sullivan, 2014; Kim et al., 2018; Lye & Koh, 2014). In text-based programming languages, novices must master the grammar of programming before moving on to Boolean phrases, loops, variables, and more complex concepts (Malan & Leitner, 2007; Wilson & Moffat, 2010). Studies have indicated that novices – both children (Howe, 1981; Levin & Kareev, 1980; Papert, Watt, diSessa, & Weir, 1979; Pea, 1983) and adults (Bonar & Soloway, 1982) – can be expected to learn to write only basic text-based programs which are grammatically correct. Although novices of all ages can be expected to write simple but grammatically correct programs (Bonar & Soloway, 1982; Howe, 1981; Pea, 1983), such programs are basic and do not necessarily represent comprehension of programming, only knowledge of the grammatical arranging of commands (Pea & Kurland, 1984). Research has suggested that block-based programming, on the other hand, is designed to accelerate novice programmers past the time-consuming and often frustrating grammar and syntax of learning text-based programming languages, allowing them more time to learn and experiment with higher-order programming concepts (Malan & Leitner, 2007; Wilson & Moffat, 2010). Malan and Leitner (2007) noted that a block-based programming language should be the first programming language learned by college-level novice programmers because block-based programming allows learners "not only to master programmatic constructs before syntax but also to focus on problems of logic before syntax" (p. 1). Instead of focusing on the minutia of text-based programming grammar and syntax, learners of block-based programming can focus on more complex thinking skills – like problem-solving – earlier, therefore creating more functionally full-bodied programs (Malan & Leitner, 2007; Wilson & Moffat, 2010). For these reasons, block-based programming has numerous instructional advantages over text-based programming languages when teaching novices.

Programming in Education

This section provides the underpinnings for why programming is the central construct in this study. Then, a brief overview of research on programming in K-12 education will be shared in order to provide the context for how programming appears in schools and why teachers are being prepared to integrate it into their instruction. Finally, how block-based programming is being used in undergraduate and teacher education will be presented to explain how teachers are experiencing block-based programming.

Programming in K-12 education. The genesis of programming in K-12 education dates back to Papert's programming language, Logo, with which students programmed an on-screen turtle to draw shapes (Abelson & DiSessa, 1986; Feurzeig, Papert, Bloom, Grant, & Soloman, 1969; Resnick, 2007; Resnick, Ocko, & Papert, 1988).

Logo’s approach to programming sparked the development of block-based programming languages such as Scratch and Alice that are commonly used today (Falloon, 2016; Mikropoulos & Bellou, 2013; Resnick, 2007; Weintrop, 2016). Due to its success as a teaching tool, block-based programming is widely being introduced in elementary and middle school classes (Werner et al., 2012; Resnick et al., 2009).

Research has shown block-based programming to have positive effects on the core subjects (Burke, 2012; Fessakis et al., 2013; Moreno-Leon & Robles, 2015; Sáez-López et al., 2016; Sengupta et al., 2013). In science classrooms, research has shown significant gains in student understanding of kinematics and ecology (Sengupta et al., 2013) and the development of enthusiasm and commitment to computer science in sixth grade (Sáez-López et al., 2016). Research into block-based programming’s effect on math skills indicated that students developed their problem-solving and mathematical understanding (Fessakis et al., 2013). Moreno-Leon and Robles (2015) even contended that a math class is the best fit for programming instruction among the general subject areas. In English, block-based programming has been used to teach literacy through digital storytelling (Burke, 2012), and research indicated that there are motivational effects of integrating programming into English instruction (Sáez-López et al., 2016). Such findings undergird principals’ and superintendents’ views that computer science should be integrated into the core subjects (Google Inc. & Gallup Inc., 2016).

Programing in post-secondary education. Block-based programming is being used not only to introduce young novices to programming, but adult learners as well (Alkaria & Alhassan, 2017; Malan & Leitner, 2007; Wilson & Moffat, 2010). According to a study by Malan and Leitner (2007), block-based programming has been used to introduce computer science class students at Harvard to programming. In this study, block-based programming instruction motivated Harvard students to learn to program and familiarized them with important computer science concepts that would transfer over to Java, a more grammar and syntax-heavy text-based programming language. Similarly, studies specific to in-service teachers (Alkaria & Alhassan, 2017; Wilson & Moffat, 2010) indicated that participants' attitudes toward teaching computer science concepts increased as a result of block-based programming professional development. Furthermore, preservice teachers' attitudes and motivation to integrate computer science concepts into their teaching improved as a result of block-based programming instruction (Yadav, Zhou, Hambrusch, & Korb, 2014; Yadav, Zhou, Mayfield, Hambrusch, & Korb, 2011). These studies represent the crux of educational research on adult learners being introduced to programming through block-based languages.

Teachers' Difficulties in Learning Programming

Various researchers have pointed out that studies on programming in education historically have heavily focused upon students, not teachers (Barr & Stephenson, 2011; Grover & Pea, 2013; Yadav et al., 2011). Other researchers have critiqued the small amount of literature on programming relating to comprehensively examining the difficulties experienced by preservice or in-service teachers while learning programming (Bower et al., 2017; Yadav et al., 2011). Most recently, Kucuk and Sisman (2018) emphasized that there continues to be a limited effort by researchers to study the experiences of preservice teachers learning to program. With the reality of the current state of the available literature related to preservice teachers' difficulties learning to program in mind, research on in-service teachers – the population preservice teachers will become upon entering the workforce – will be presented along with the small amount of research on the difficulties experienced by preservice teachers. Through this method, a comprehensive explanation of the literature available on in-service teachers, in addition to preservice teachers, will paint a more informed picture of difficulties these linked populations face while learning to program.

In-service teachers' challenges. Research has shown that in-service teachers can experience difficulties as technology advances and computer concepts become a more substantial part of the K-12 curriculum (Bower et al., 2017; Grover & Pea, 2013; Israel et al., 2015; Resnick et al., 2009). First, research suggests that teachers have difficulties adapting their teaching to teach computer concepts because they are not comfortable using and developing lessons around new technologies (Curzon et al., 2009; Meerbaum-Salant et al; Schanzer, 2015). Exacerbating this problem, teachers have misconceptions about computer science, which repel them from learning and then teaching computer science concepts like programming in classrooms (Bower et al., 2017; Milton, Rohl, & House, 2007). Teachers lack confidence in teaching computer science topics because they are often not computer science majors and therefore do not feel credentialed enough to teach the subject in their classrooms (Bender, Schaper, Caspersen, Margaritis, & Hubwieser, 2016; Israel et al., 2015). In fact, Bower et al. (2017) reported that 78% of teacher participants (*N* = 69) had a low level of self-confidence about teaching computational thinking in their classrooms after taking part in full-day learning activities on basic computer science topics such as dissecting problems, recognizing patterns, abstraction, and algorithms. Most significantly, teachers report a lack of confidence teaching computer science content due to their views of the perceived level of difficulty and abstractness attributed to the subject (Bower et al., 2017; Grover & Pea, 2013; Israel et al., 2015; Resnick et al., 2009). These are all reasons in-service teachers have difficulties with learning programming, which can inhibit them from integrating programming into their instruction.

Preservice teachers' challenges. There is emerging research on preservice teachers being trained to use programming in teacher preparation classes. For example, research reported that preservice teachers experienced issues with programming concepts like identifying variables, defining conditions, and identifying errors (Kim et al., 2015, 2018). A study by Ortiz et al. (2015) noted that 12% of preservice teacher participants did not feel prepared to integrate this type of instruction into their teaching after going through training. This population of preservice teachers echoed the sense of feeling intimidated by the abstract math concepts required to teach programming (Ortiz et al., 2015). These studies imply that preservice teachers, like in-service teachers, experience difficulties with programming concepts.

Educational Robotics

Educational robotics are an important tool in programming education. This section will overview educational robotics, a main construct in the research questions of this dissertation. This section is broken into four parts. First, how studies characterize key educational robotics terms will be explained. Next, how educational robotics are used and how educational robotics relate to block-based programming will be described. Then, theoretical frameworks for educational robotics practices that are found in the literature will then be shared. To conclude, difficulties experienced by teachers using educational robotics will be disclosed. The four elements in this section are designed to provide readers with a summary of literature on educational robotics in education.

Defining Educational Robotics

Educational robotics was defined by Eguchi (2012) broadly as ‘‘the use of robotics as a learning tool’’ (p. 3). Ortiz et al. (2015) provided a more specific definition of educational robotics, or “a specific application of K–12 engineering education and offers students physical manipulatives that are familiar and easy to work with as they participate in the engineering design process” (p. 43). Catlin (2012) characterized educational robotics practices as instructional strategies that use robotics for instructional purposes. These examples provide a general characterization of educational robotics.

Educational Robotics for Teaching and Learning

Having a frame of reference for how educational robotics have been used for teaching and learning is essential background information for understanding educational robotics practices. This section has two focuses. This section will describe (1) how block-based programming and educational robotics are combined, and (2) the advantages of implementing educational robotics practices for programming education that are found in the literature.

Pairing programming with educational robotics. The genesis of educational robotics started with Papert’s Logo programming language (Alimisis et al., 2007; Casler-Failing, 2017). Logo’s turtle concept inspired Perlman’s (1974) TORTIS programming, which, for the first time, included educational manipulatives that could be programmed. Resnick et al. (1988) later paired Lego gears, motors, and sensors with a computer running the Logo programming software. Today, there are numerous types of educational robotics kits available to educators, many which pair robotics pieces with block-based programming environments like Lego Mindstorms robots and Lego EV3-G programming language, or mBlock robots and Scratch programming language (Dodds et al., 2006; Gunbatar & Karalar, 2018; Weintrop, 2016). With the growing popularity of programming initiatives in schools, the use of educational robotics as a programming vessel is becoming widespread in education (Dodds et al., 2006; Rogers, Wendell, & Foster, 2010).

Students or instructors can build educational robots to accomplish specific tasks. For example, sensors or lifting devices may be built onto the chassis of the manipulative in order to navigate through an obstacle course and pick up an object (Bers et al., 2002; Martin et al., 2011; McNally, Goldweber, Fagin, & Klassner, 2006). Educational robots can run based on commands written in block-based programming languages (Alimisis et al., 2007; Petre & Price, 2004). Programming for the educational robotics can be composed on computers or mobile devices in a block-based programming environment and uploaded to the controller unit of each robot either wirelessly by Bluetooth or physically by USB connection (McGill, 2012; Petre & Price, 2004).

Dagdilelis, Sartatzemi, and Kagani (2005) and Staszowski and Bers (2005) offered similar outlines for pairing block-based programming with educational robotics activities in the classroom. Since both block-based programming and educational robots can be constructed, deconstructed, and modified, students can design both their robots and the programs running on the robots to accomplish different tasks (Dagdileliset al.; 2005; Staszowski & Bers, 2005). Dagdilelis et al. (2005) outlined a more technical and action-oriented structure of (1) constructing a robot, (2) writing a program using a visual programming language, (3) transmitting that program to the educational robot, and (4) running the program. Dagdilelis et al. (2005) noted that steps two and four are often repeated many times as students solve problems and modify their educational robotics designs and programs. Staszowski and Bers' (2005) listed five major occurrences that happen while students are engaged in activities that combine programming and educational robotics: (1) design, (2) building, (3) building concepts, (4) programming, (5) programming concepts. These occurrences take more of a big picture view of the process and note mental exercises of building concepts and programming concepts. Dagdilelis et al. (2005) and Staszowski and Bers (2005) include the commonalities of building a robot to perform a certain task and then programming a robot to execute the required commands.

Advantages of educational robotics. There are numerous benefits of educational robotics, which have been noted in the literature. For example, Huang, Yang, and Cheng (2013) studied the impact of using educational robotics on programming achievement. Their findings indicated that students who learned programming through educational robotics demonstrated higher programming achievement than those who learned programming through flowcharts. Educational robotics can be considered as manipulatives for learning to program in the style of Montessori (Brosterman, 1997). While Montessorian manipulatives were designed to help students better understand numbers, educational robotics help students understand abstract science, math, and computer science lesson content (Bers, 2010; Bers et al., 2002; Bers & Portsmore, 2005; Brosterman, 1997). For example, educational robotics enhance the traditional programming learning experience by breaking down the barrier between the computer screen where block-based programs live and the real, physical world where these intangible programs can be acted out physically (Mikropoulos & Bellou, 2013). Real-world application allows students to make connections between the content being studied and how the content is used outside of the classroom (Adams, Miller, Saul, & Pegg, 2014). Since educational robotics can be used to reduce the level of abstractness of science and mathematics concepts (Nugent, Barker, Grandgenett, & Adamchuk, 2010), educational robotics has been demonstrated to be effective in the teaching of STEM concepts (Altin & Pedaste, 2013; Barker, Nugent, & Grandgenett, 2014). Students can actively learn in a student-centered approach by physically interacting with gears, motors, and sensors, among other aspects, through the construction of their own robots (Bers, 2008; Wang & Ching, 2003). As synthesized in Table 2.1, fine motor skills, STEM knowledge, physics knowledge, mathematics skills, and programming understanding have improved in participants as outcomes of educational robotics practices in the classroom. Successful outcomes relating to the use of educational robotics like those highlighted in this paragraph have led to educational robotics' emerging popularity in schools and the field of education.

Table 2.1. *Significant Educational Robotics Findings in K-12 Education*

Study	Population	Significant Findings
Bers et al. (2014)	Kindergarten	Participants were interested and could learn many of the robotics and programming concepts in the curriculum. Educational robotics develop students' fine motor skills.
Lindh & Holgersson (2007)	Elementary and middle school	Educational robotics improved elementary and middle school students' math performance and STEM knowledge.

Table 2.1. *Significant Educational Robotics Findings in K-12 Education* Continued.

Study	Population	Significant Findings
Mikropoulos & Bellou (2013)	Elementary and middle school	Educational robotics can be used to aid students in developing physics knowledge through constructionist robotics activities.
Karahoca, Karahoca, & Uzunboylub, (2011)	Elementary and middle school	Improvement demonstrated in students' self-confidence and mathematics learning.
Casler-Failing (2017)	Middle school	Educational robotics increased student engagement and aided in the learning of ratios and proportional reasoning skills.
Castledine & Chalmers (2011)	Middle school	Educational robotics helped students reflect on problem-solving and allowed students to exercise higher-order thinking skills.
Dagdilelis et al. (2005)	High school	The correct usage of basic programming concepts was better understood with the use of educational robotics.

Theoretical Frameworks for Educational Robotics Practices

Educational robotics practices utilize robots as mindtools (Jonassen, 2000) and adhere to the principles of constructivism and constructionism (Alimisis, 2013; Kucuk & Sisman, 2018). In fact, Mikropoulos and Bellou (2013) reported in their research that most educational robotics studies followed a mixed constructivist-constructionist theoretical framework. This section covers three aspects common to educational robotics theoretical frameworks found in the literature. These common aspects are (1) the use of robots as mindtools to aid student learning, and the utilization of mindtools within (2) constructivist theoretical frameworks, and (3) constructionist theoretical frameworks.

Constructivism. Educational robotics practices for programming align with Piaget's (1967, 1973) theory of constructivism (Harel & Papert, 1991; Mikropoulos &

Bellou, 2013; Petre & Price, 2004). According to Piaget (1967, 1973), constructivism is the building of abstract knowledge structures in one's mind through concrete experiences. Some researchers even suggest that educational robotics represent one of the most effective examples of the application of constructivist theory (Kaya, Newley, Deniz, Yesilyurt, & Newley, 2015; Papert, 1993).

In the constructivist view of learning, the mental creation of knowledge necessitates the use of hands-on activities (Alimisis, 2013; Piaget, 1973; Ucgul, 2013). As the manipulative is used to create concrete representations during the creation of abstract mental models, educational robotics fit within the constructivist framework (Mikropoulos & Bellou, 2013). Furthermore, Petre and Price (2004) emphasized, "In robotics, students' learning is concrete, associated with phenomena they create, observe and interact with," and it is through the physical manipulatives that "the abstractions they derive (or apply later) are grounded and relevant," (p. 148). With their ability to be used as physical manipulatives which can illuminate abstract concepts, educational robotics can be used as a constructivist mindtool for learning.

Constructionism. Both a learning theory and educational strategy, constructionism builds on Piaget's (1967) theory of constructivism by emphasizing the construction of hands-on products. Born from Papert's (1980) constructionist framework, the term constructionism was explained by Kafai and Resnick (1996) as "two types of intertwined construction" wherein "a designer comes to understand not only objective constraints but also subjective meaning" (p. 2). The first type of construction is physical and occurs when students construct their own learning artifacts through hands-on activities (Papert, 1980; Papert, 1993). The meaning-construction described by Kafai and

Resnick (1996) is the second type of entwined construction. On a mental level, constructionism, like constructivism, theorizes that learning is not as simple as the instructor transferring knowledge to the student (Papert, 1980, 1993). Rather, learning occurs when students construct, deconstruct, and reconstruct understanding in their minds based on their learning experiences aided by physical construction (Kafai & Resnick, 1996; Mikropoulous & Bellou, 2013; Papert, 1993; Resnick & Silverman, 2005). As students construct their learning artifacts, they learn by continually creating and updating knowledge in their minds.

A key difference between constructionism and constructivism is that more emphasis is placed on students constructing learning artifacts through hands-on activities in constructionism (Kafai & Resnick, 1996; Papert, 1993). Kafai and Resnick (1996) argued that the difference between constructivism and constructionism is that "Constructionist theory goes beyond Piaget's constructivism in its emphasis on artifacts, asserting that meaning-construction happens particularly well when learners are engaged in building external and sharable artifacts" (p. 2). The learning artifacts in constructionism that are created by students "are subject to the test of reality; if they don't work, they are a challenge to understand why and to overcome the obstacles," Papert (1999, p. XIII) stressed. Therefore, constructivism is the idea that knowledge is built in one's brain, while constructionism is more situated and pragmatic with the idea that knowledge is built through constructing tangible learning artifacts outside of the brain (Papert, 1990).

Due to the buildable nature of many educational robotics kits and the block-based programs, they are often operated with, constructionism is heavily associated with the combination of educational robotics and block-based programming. The utilization of constructionism in educational robotics theoretical frameworks is fitting, as Kafai and Resnick (1996) have affirmed, because “Constructionist theory suggests a strong connection between design and learning” as it “asserts that activities involving making, building, or programming – in short, designing – provide a rich context for learning” (p. 2). Chambers and Carbonaro (2003) asserted that “mindtools, in the form of robotics, represents a constructionist approach to using technology” by aiding students in “representing knowledge, manipulating virtual and concrete objects, and reflecting on what they have designed and built” (p. 212). While constructionism has been used in the theoretical frameworks for studies on the use of educational robotics with preservice teachers (Hadjiachilleos, Avraamidou, & Papastavrou, 2013; Kabatova & Pekarova, 2010), more numerous studies have focused the early childhood, elementary, and middle levels (Bers, 2010; Erwin, Cyr, & Rogers, 2000; Meerbaum-Salant et a;., 2013; Papert, 1993). The construction of the physical manipulatives and the programming of commands in educational robotics activities align with the constructionist learning theory, which postulates that depth of learning is tied in large part to the physical construction of learning artifacts.

Learning through collaboration within a community of learners is a pillar of constructionist theory (Papert, 1980; Huang et al., 2013). Accordingly, the collaboration of students in small groups for building and programming educational robotics is a core part of numerous studies’ instructional frameworks (Bakke, 2013; Bers & Portsmore, 2005; Castledine & Chalmers, 2011; Chambers & Carbonaro, 2003; Kabatova & Pekarova, 2010; Mikropoulos & Bellou, 2013). Backing this aspect of constructionist frameworks, researchers have identified that using educational robotics in conjunction with collaboration leads to positive results (Denis & Hubert, 2001; Huang et al., 2013; Wang, 2001). For example, Denis and Hubert (2001) and Eguchi (2007, 2013) found that constructionist robotics activities developed participants' collaboration skills. Eguchi (2013) noted that 100% of students (*N* = 18) reported learning teamwork skills through the collaborative element of the constructionist robotics activities used by the researchers. Participants have also found the collaborative component of constructionist robotics activities to be beneficial for brainstorming and receiving feedback on programming ideas (Petre & Price, 2004; Sisman & Kucuk, 2019). Constructionist frameworks for robotics activities have garnered positive results by encouraging teamwork and the modification of participants' understanding through the processes of feedback and reflection between participants, their peers, and their instructors (Denis & Hubert, 2001; Eguchi, 2013; Petre & Price, 2004; Sisman & Kucuk, 2019).

Backing the spectrum of constructivist-constructionist educational robotics frameworks described in the above two sections is the use of educational robotics as mindtools. Jonassen (2000) popularized the term mindtools to describe computer-enabled tools that can be built or modified that aid in the facilitation of higher-order thinking skills. Students use the robots as aids to think with – helping them create mental models – and not from (Bers et al., 2002; Chambers & Carbonaro, 2003; Mikropoulos & Bellou, 2013; Smith, 2013). The robots themselves are not what is being studied when mindtools are utilized – although a better understanding of the nuts and bolts of the robots may be an additional value – because the focus is on the use of the manipulatives to illustrate the abstract concepts often in the realms of science and math (Bers et al., 2002). Mikropoulos and Bellou (2013) explained five reasons why educational robotics are commonly used as mindtools. These reasons included (1) the construction of knowledge through project-based assignments which utilize real-world models, (2) providing a safe avenue for failure and discovery in a real-world environment, (3) allowing for learning through the scientific method, (4) allowing students to partake in manipulatives-based reflection, and (5) learning through collaboration and feedback in a community of learners (Mikropoulos & Bellou, 2013). To this end, Mikropoulos and Bellou (2013) reported that mindtools have a functional duty within both constructivist and constructionist frameworks.

Teachers' Difficulties of Integrating Educational Robotics into Education

There are several barriers, limitations, and difficulties users experience while learning with educational robotics (Bruciati, 2004; Kim et al., 2018; Kucuk & Sisman, 2018; Major, Kyriacou, & Brereton, 2014; McNally et al., 2006). These issues can be grouped into three categories: (1) financial barriers, (2) physical limitations, and (3) mental difficulties. The following paragraphs in this section will outline the financial, physical, and mental difficulties that have been described in the literature relating to educational robotics.

Financial barriers. Costs associated with purchasing, maintaining, and even storing educational robotics may make the manipulatives an unjustifiable tool for teaching programming in some contexts (Greenley & Tidwell, 2002; Major et al., 2014). If obtaining robots for each student is unattainable in a school's budget, this can lead to students working in groups (Smith, 2013). Although the benefits of a group dynamic for educational robotics frameworks have been outlined above, a study by Kucuk and Sisman (2018) highlighted that preservice teachers expressed difficulties adapting to the group structure of educational robotics activities. Moreover, the cost inherent to educational robotics may discourage institutions from letting students take the manipulatives outside of the classroom (Major et al., 2014; McNally et al., 2006). If institutions have enough computers or mobile devices for their students, classes can perform similar programming exercises without the additional cost of educational robotics kits by using online simulators (Major et al., 2014; McNally et al., 2006). For these various reasons, educational robotics have inherent financial barriers.

Physical limitations. As they are not directly necessary for programming education, educational robotics can create distractions for students and teachers (Major et al., 2014; McNally et al., 2006). Mechanical failure is one added issue when integrating educational robotics into programming education, which may impact both teachers and students (Major et al., 2014). For teachers, instructional time and preparation time can be lost to constructing the robots and setting up obstacle courses for students to program the robots through (Major et al., 2014). In addition, Kucuk and Sisman (2018) noted that their preservice teacher participants experienced difficulties with the physical aspects of the educational robotics activities, including problems with understanding the design steps, as well as losing interest in designing the robots. Preservice teachers also experienced difficulties connecting motors and sensors to ports and arranging the proper blocks of programming (Kucuk & Sisman, 2018). Similar data were gathered in a study by Sisman and Kucuk (2019) in which preservice teachers experienced difficulties with connecting the correct sensors to ports and assembling the educational robots because of the small parts. Substantiating Kucuk and Sisman's (2018) and Sisman and Kucuk's (2019) findings, a study by McGill (2012) with a population of non-computer science majors learning programming reported participants' frustration with the physical aspects of robots, including parts, sensors, and connectivity issues. As noted by these studies, the physical aspect of educational robotics can cause difficulties for some learners.

Mental difficulties. Educational robotics may lead to mental difficulties for some learners (Bruciati, 2004; Kim et al., 2018; Kucuk & Sisman, 2018). Notably, some of the preservice teachers in Kucuk and Sisman's (2018) study continued to report issues understanding complex programming processes. Another problem that has been observed in preservice teacher educational robotics studies deals with debugging (Kim et al., 2018). This research showed that during activities that combined block-based programming and educational robotics, many preservice teachers feared being embarrassed by writing code that would not run properly on the robots (Kim et al., 2018). Consequently, preservice teachers erred on the side of caution and wrote more basic programs (Kim et al., 2018). Sisman and Kucuk (2019) reported similar findings in which preservice teachers felt debugging was a time-consuming and often frustrating process. In addition, researchers (Bruciati, 2004; Kucuk & Sisman, 2018) caution that intrinsic cognitive load may be increased by adding educational robotics to programming exercises. As noted by these researchers, the added mental impacts of educational robotics can cause difficulties for some learners.

Impact of Educational Robotics on Programming Comprehension

There is a need to prepare preservice teachers to integrate STEM learning into their future instruction (Kim et al., 2017). This section will begin by defining programming comprehension. Then, cognitive learning theories will be explained. After that, programming comprehension frameworks will be detailed. Next, a synthesis on the topic of programming comprehension and teachers will be shared. To finish, an overview of the different ways programming comprehension has been measured relating to this study's population will be examined.

Defining Programming Comprehension

Comprehension can be demonstrated by students by comparing, interpreting, describing, or organizing (Bloom, Engelhart, Furst, Hill, & Krathwohl, 1956). Programming comprehension has been described by Ramalingam and Wiedenbeck (1997) as "the process of understanding a program written by oneself or someone else, normally for the purpose of doing some further task with the program which requires understanding" (p. 125). Ala-Mutka (2004) described programming comprehension as the "ability to track code to build a mental model of the program and predict its behavior" (p. 5). Programming comprehension, Ramalingam and Widenbeck (1997) have asserted, consists of the skills people use to collaborate, modify and streamline programs as "most programming does not involve writing a program from scratch but instead starts from the basis of existing programs" (p. 125).

Cognitive Learning Theories

Learning theories help researchers explain the mental processes of how people learn (Harasim, 2012). Cognitive learning theories are the basis of cognitive models that explain how information is obtained from the learner's environment and then processed into comprehension and long-term knowledge (Kalyuga, 2010; Sweller, 1994). This section will overview germane theories related to programming comprehension: schema theory and information processing.

Schema theory. To begin, schema theory explains how learners create models in their minds using an interconnected network of nodes organized through relationships among similar concepts (Johnson-Laird, 1983). Kalyuga (2010) explained schema as the relationships, categories, patterns, and overall meaning the mind ascribes to different information. Multiple schemas can be used in conjunction with each other in a hierarchical structure (Kalyuga, 2010). Short-term, or working memory, temporarily stores the information that is currently being used by the processor, is limited to a small number of ideas, and is responsible for the coordination of information and thinking (Atkinson & Shiffrin, 1968; Baddeley, 1992; Miller, 1956). Long-term memory is larger in capacity and contains all the knowledge a learner can call upon in order to give context to or understand a new idea (Klatzky, 1980; Smith, Shoben, & Rips, 1974). In Klatzky's (1980) Network Model of Long-term Memory, long-term memories are likened to a mental dictionary with concepts filed by association while different nodes house conceptual associations and work in concert to form memories. The feature comparison model of long-term memory (Smith et al., 1974) differs from Klatzky's (1980) network model. In the feature comparison model, defining characteristics are compared in memory recall (Smith et al., 1974). The propositional models of long-term memory (Norman & Rumelhart, 1975) mix aspects of the previously described models in which nodes take stored basic background information and combine that input with a proposition using a subject and a predicate. The parallel distributed processing models of long-term memory (McClelland, 2011), Driscoll (2005) explained, differs in that "multiple cognitive operations occur simultaneously as opposed to sequentially" (p. 95). Schema are often organized by semantic concepts (Navarro-Prieto & Canas, 2001;

Ormerold, 1990), and larger concepts can be combined, called chunks. Chunks contain large amounts of associated information that are interconnected through concepts and extensions (Sweller, 1994). Then, the chunks of automatic processing interact to create new schemas as new material is learned (Sweller, 1994). Schema theory helps explain how people handle and comprehend information.

Information processing models. The information processing model, or IPM, is a theory which explains how learners process information (Newell & Simon, 1972). In the IPM, learners are like computers – or more fitting with this study, robots using sensors – and obtain information through their receptors, like the eyes and ears (Newell & Simon, 1972). The information that is obtained by the receptors is then sent to the processor, whose function it is to understand the information (Newell & Simon, 1972). Similar information is stored within a learner's memory using different silos, or nodes (Newell & Simon, 1972). Nodes are arranged starting with the name of the concept and extend into the nature of the concepts associated with that name (Kristensen & Osterbye, 1994). From there, nodes are further associated by intention, or the facets the concepts have in common.

Similarly, Atkinson and Shiffrin's (1968) Multi Store Model of Memory has many overlapping ideas about the comprehension of knowledge as Newell and Simon's (1972) IPM. In the Multi Store Model of Memory, information is obtained from the environment through the senses like a computer, and it is then processed in a linear fashion (Atkinson & Shiffrin, 1968). Driscoll (2005) explained that Atkinson and Shiffrin's (1968) model utilizes the structure of a "multistore, multistage theory of memory," (p. 74-75) where information is absorbed through the receptors and then flows through a metamorphosis from each state of sensory, working, and then long-term memory. In the Multi Store Model of Memory, information is encoded visually, by sound, or by meaning (Atikinson & Shiffrin, 1968).

Programming Knowledge Frameworks

Programming comprehension can be explained through learning theory. In this section, the types of programming knowledge that researchers have identified learners use when writing programs are discussed. Then, frameworks that explain how learners come to comprehend programming will be detailed.

Syntactic, semantic, and strategic knowledge. Types of programming knowledge can be divided into three different categories: syntactic, semantic, and strategic (Bucks, 2010; Mayer, 1979; McGill & Volet, 1997). Syntactic programming knowledge includes the vocabulary, grammar, and organizational rules used in a specific programming language (Mayer, 1979). Syntactic programming knowledge is unique to each programming language in much the same way English and Spanish have different rules about vocabulary, grammar, and syntax (Bucks, 2010). Semantic, described by Bucks (2010) as conceptual programming knowledge, on the other hand, includes programming ideas or functions which are transferrable between programming languages (Soloway & Ehrlich, 1984). Both syntactic and semantic knowledge contribute to strategic knowledge when creating or understanding a program in a certain context and aid in one's ability to problem-solve in programming (Bucks, 2010). Strategic knowledge pertains to the problem-solving skills used to complete a programming problem (McGill & Volet, 1997). The three categories of programming knowledge include syntactic, semantic, and strategic knowledge and contribute to programming comprehension.

Frameworks of programming comprehension. There are multiple frameworks to explain how programming is comprehended. Two prominent frameworks are those of Mayer (1981) and Pennington (1986). These frameworks will be explored in this section.

Mayer's model. Mayer (1981) used the IPM (Newell & Simon, 1972) to explain programming comprehension. In the IPM, the cognitive processes which take place in the mind are represented by a computer (Newell & Simon, 1972). In Mayer's (1981) model, the learner experiences the new information and processes it using short-term memory. While new information is being processed in the short-term memory, links are searched for within the long-term memory in order to give context and a previous understanding of the information (Mayer, 1981). The connected long-term memories are brought into the short-term memory, and then the mind updates the existing mental model relevant to the concept or adds the new information (Mayer, 1981). Bayman and Mayer (1983) investigated this model and found that most participants had an incomplete understanding of programs they were tested on at the conclusion of an introductory programming course. The researchers determined that the novice programmers needed concrete models of the programs in order to develop the necessary mental models for comprehension (Bayman & Mayer, 1983).

Pennington's model. Pennington's (1986) framework of programming comprehension expands upon classic language comprehension frameworks by borrowing the idea of layered mental representations. Surface form representation, textbase representation, and situational modeling are all aspects Pennington (1986) borrows from traditional text comprehension models. Surface form representation is the first layer, which consists of a word for word recall of the text (Ramalingam & Wiedenbeck, 1997).

Next is the textbase representation, which includes abstractions from surface form representation (Pennington, 1986). Finally, there is the situation model in which the reader adds context to the text based on the reader's previous knowledge or experiences with the content which the text describes (Pennington, 1986; Ramalingam & Wiedenbeck, 1997).

Pennington's (1986) programming comprehension framework divides into five levels: program, domain, operations, function, and state. The traditional concept of textbase representation aligns with Pennington's (1986) program model while the traditional situation model aligns with Pennington's (1986) domain model (Ramalingam & Wiedenbeck, 1997). The program model includes operations knowledge – understanding of basic pieces of programming – and flow control knowledge or understanding of loops or if/then statements (Ramalingam & Wiedenbeck, 1996). The domain model includes variables and the changing of data (Pennington, 1986). Operations knowledge includes basic operations in a single line of programming, function knowledge includes knowledge of the outcome of the program, while state knowledge consists of understanding how all the pieces of the program work together (Pennington, 1986; Ramalingam & Wiedenbeck, 1997).

Two studies utilized Pennington's (1986) framework. In the first study, professional programmers reviewed short programs within their programming language expertise (Pennington, 1986). Then, participants underwent a memory test based on program, domain, operation, functions, and state elements, and results showed that operations knowledge was well represented while domain knowledge was poorly represented (Pennington, 1986). In a second study, professional programmers were given a longer program and then underwent a memory test based on program, domain, operation, functions, and state (Pennington, 1987). Next, the participants modified the program and answered new questions. Although the results of the first phase included errors in domain knowledge, the results of the second phase of the study had the highest scores on the domain knowledge (Pennington, 1987). Pennington (1987) interpreted these results as showing that program and domain knowledge are different and that comprehension-based activities likely increase domain knowledge.

Programming Comprehension and Educational Robotics

This section is divided into two focuses. First, educational robotics studies that have assessed programming comprehension among in-service and preservice teachers will be shared. Then, the ways in which programming comprehension of preservice teachers has been measured will be detailed.

Educational robotics' impact on teachers' programming comprehension.

Research in the areas of preservice and in-service teachers' comprehension of programming and robotics is emerging (Eguchi, 2013; Jaipal-Jamani & Angeli, 2017; Kay, Moss, Engelman & McKlin, 2014; Kim et al., 2018; Kucuk & Sisman, 2018; Perritt, 2010; Sullivan & Moriarty, 2009). These studies evaluate different aspects of programming comprehension through educational robotics. Since there are so few studies in the area of teachers' programming comprehension and robotics, each of the following paragraphs will be dedicated to detailing either a study of in-service teachers' or preservice teachers' programming comprehension.

In-service teachers. Through professional development sessions, the effects of educational robotics on in-service teachers' programming comprehension have been studied (Kay et al., 2014; Sullivan & Moriarty, 2009). These studies provided as-needed robotics professional development to train teachers in computer science concepts (Kay et al., 2014; Sullivan & Moriarty, 2009). The following paragraphs will synthesize these studies.

Kay et al. (2014) evaluated educational robotics' effects on programming comprehension among in-service K-12 teachers with no prior programming experience (*N* = 41). Over the course of three days of Lego robotics programming workshops, participants learned how to write basic programs for the robots and the skills necessary to start their own robotics clubs (Kay et al., 2014). Results indicated a statistically significant increase in programming knowledge and skills, with 90% of participants reporting that they felt they were competent or skilled in programming (Kay et al., 2014). The researchers stated that these results suggested that programming understanding among in-service teachers increased with the use of educational robotics (Kay et al., 2014).

Sullivan and Moriarty (2009) evaluated the robotics and programming knowledge of 20 in-service middle and high school teachers. The in-service teachers participated in professional development workshops at a robotics fair and were assessed with a pretest/posttest robotics and programming content knowledge instrument as well as a self-assessment survey (Sullivan & Moriarty, 2009). Results indicated statistically significant differences between the pretest and posttest, with all participants reaching a general knowledge of all assessment concepts (Sullivan & Moriarty, 2009). The self-assessment data indicated that the participants' content knowledge related to robotics and programming increased significantly because of the workshops (Sullivan & Moriarty, 2009).

Preservice teachers. Researchers have begun to study preservice teachers' comprehension of programming in different contexts (Eguchi, 2013; Jaipal-Jamani & Angeli, 2017; Kim et al., 2018; Kucuk & Sisman, 2018). Numerous researchers point out that robotics instruction is becoming more common in preservice teacher preparation around the world (Bruder & Wedeward, 2003; Hadjiachilleos et al., 2013; Kay et al., 2014; Kaya et al., 2015; Kim et al., 2015; Majherová & Králík, 2017; Sullivan & Moriarty, 2009). However, there is limited research on using educational robotics for training preservice teachers in teacher preparation courses (Kucuk & Sisman, 2018; Jaipal-Jamani & Angeli, 2017). Few researchers have studied programming comprehension of preservice teachers through the lens of educational robotics activities (Eguchi, 2013; Jaipal-Jamani & Angeli, 2017; Kim et al., 2018). Each of these studies will be detailed in the following paragraphs.

Eguchi (2013) studied 18 preservice teachers participating in an educational robotics course. Participants worked in groups sharing one robot and one computer (Eguchi, 2013). Participants were evaluated through observations (Eguchi, 2013). For the observations, participants were evaluated while teaching groups of classmates how to program their robots through difficult programming tasks (Eguchi, 2013). Each group was successful in teaching the other groups during the observations (Eguchi, 2013). Eguchi (2013) contended that teaching "indicates their mastery of the programming skills required in class since teaching is the highest form of learning" (p. 9).

Jaipal-Jamani and Angeli (2017) evaluated 21 elementary preservice teachers' understanding of science and computational thinking concepts as a result of robotics activities in a science teaching methods course. The robotics activities accounted for six hours of contact time in which participants learned about algorithms, debugging, control structures, and writing sequences of programming (Jaipal-Jamani & Angeli, 2017). The researchers found statistically significant differences between the pre and posttest knowledge assessment scores, indicating that robotics activities were an effective strategy for increasing participants' abilities to write algorithms and debug programs (Jaipal-Jamani & Angeli, 2017).

Kucuk and Sisman (2018) studied 15 preservice teachers' experiences while learning programming and robotics. The participants learned programming and robotics in collaborative groups through a 13-week course, which met for four hours per week (Kucuk & Sisman, 2018). The robotics activities involved participants composing original programs for the robots (Kucuk & Sisman, 2018). Participants of the study indicated that they felt the educational robotics programming course improved their programming skills (Kucuk & Sisman, 2018).

Kim et al. (2018) assessed 19 preservice teachers' debugging techniques and common errors while using block-based programming. Debugging constitutes strategic programming knowledge (McGill & Volet, 1997), which combines both syntactic and semantic programming knowledge. In this study, preservice teachers participated in 12 hours of robotics learning modules wherein they built and programmed robots (Kim et al., 2018). In their research, Kim et al. (2018) revealed that preservice teachers have difficulty locating and fixing errors in block-based programs.

Measuring preservice teachers' programming comprehension. Studies have used different measures to evaluate the programming comprehension of preservice teachers, both with and without the intervention of educational robotics (Eguchi, 2013; Jaipal-Jamani & Angeli, 2017; Kucuk & Sisman, 2018; Kim et al., 2018; Yadav et al., 2014). However, few of these studies have comprehensively measured and reported the impacts of different interventions, such as robotics, on preservice teachers' programming comprehension (Kim et al., 2015). For instance, a study by Arlegui, Pina, and Moro (2013) on training teachers to use educational robotics provided only anecdotes about what participants learned. In another example, a study by Bers and Portsmore (2005) focused on partnerships between preservice teachers and engineering students learning programming with educational robotics. The following paragraphs will first detail the qualitative measures that have been used to assess preservice teachers' comprehension of programming; then, the quantitative measures will be described.

Qualitative measures. Various studies have utilized qualitative methods with which to evaluate preservice teachers' learning and comprehension of programming (Eguchi, 2013; Kim et al., 2018; Kucuk & Sisman, 2018). In an educational robotics intervention, Kucuk and Sisman (2018) used preservice teachers' responses to interview questions about their feelings on changes in their programming comprehension as a result of the study. Preservice teachers' grasp of programming concepts was also evaluated by Eguchi (2013). In this study, preservice teachers were evaluated through teaching observations performed by the instructor. In a thorough investigation, Kim et al. (2018) measured preservice teachers' comprehension of block-based programming by evaluating their debugging skills. Like Eguchi (2013), Kim et al. (2018) relied on observational data.

To do this, Kim et al. (2018) reviewed video recordings of students' debugging processes and used a coding instrument based on Vessey's (1985) debugging paths in conjunction with Katz and Anderson's (1987) error-locating techniques. This study did not implement educational robotics and focused on participants' debugging processes in a block-based programming environment. The researchers focused on the debugging process citing the ideas of researchers such as Brennan and Resnick (2012), Grover et al. (2015), and Pea and Kurland (1984), who agree that students who create programs that simply run do not necessarily understand programming. Programs that run do not necessarily demonstrate programming comprehension because the program may run by chance due to students tinkering and rearranging programming blocks until a successfully functional arrangement of blocks is found (Brennan & Resnick, 2012; Kim et al., 2018). Therefore, Kim et al. (2018) investigated programming comprehension through the lens of debugging instead of through methods that evaluate if students can simply arrange programming blocks into functional formations (Kim et al., 2018). These studies utilized different qualitative measures to investigate programming comprehension among preservice teacher participants.

Quantitative measures. Only one study uncovered in this literature review carefully assessed preservice teachers' programming comprehension through quantitative measures. In an educational robotics study, Jaipal-Jamani and Angeli (2017) used two measures to gauge preservice teachers' programming comprehension. These measures included: (1) a questionnaire to measure preservice teachers' science knowledge which also included 3 Likert-type questions to assess participants' perceived programming knowledge, and (2) a worksheet to assess participants' comprehension on sequencing, control structures, and debugging (Jaipal-Jamani & Angeli, 2017). These two different measures were used to investigate preservice teachers' programming comprehension through quantitative methods.

Impact of Educational Robotics on Motivation Related to Programming

Few studies have examined motivation in relation to learning programming (DeClue, 2003; Feldgen & Clua, 2004; Jenkins, 2001; Kelleher et al., 2007). The section will focus on motivation related to programming. This section will begin with definitions of motivation and teacher motivation, as well as descriptions of contributing factors to motivation and motivation frameworks. To close, a synthesis of literature on the motivational impact of educational robotics on teachers will be presented.

Motivation

Johns (1996) describes motivation as the extent to which persistent effort is sustained toward a specific goal. Motivation combines mental and physical processes and presents as one's determination to spend time and effort on a task and can be divided into two general categories of motivation: intrinsic and extrinsic (Cullen & Greene, 2011; Deci & Ryan, 2000). Intrinsic motivation applies to internal drive to complete tasks based on personal desire (Deci & Ryan, 2000; Maslow, 1943; Skinner, 1954). Extrinsic motivation, on the other hand, applies to external rewards such as pay given for completing tasks (Taylor, 1916). According to research by Sinclair (2008), teachers' intrinsic motivation is greater than extrinsic motivation to teach. As cited in Han and Yin (2016), Dörnyei and Ushioda (2011) divide teacher motivation into multiple components. Han and Yin (2016) explained these components as (1) teachers' inherent interest in teaching, (2) lifelong commitment, and (3) discouraging factors based on teachers' negative experiences. Motivation is abstract, complex, and includes numerous indicators (Ball, 1977; Jenkins & Davy, 2002; Law, Lee, & Yu, 2010). These numerous aspects of motivation will be explained here. Then, frameworks dealing with motivation will be outlined.

Indicators of motivation. Researchers have put forward numerous indicators of motivation which fall into general categories like engagement (Singh, Granville, & Dika, 2002), extrinsic motivation (Amabile, Hennessey, & Tighe, 1994; Law et al., 2010; Taylor, 1916), interest (Dewey, 1913; O'Keefe & Harackiewicz, 2017), intrinsic motivation (Deci & Ryan, 2000; Maslow, 1943; Skinner, 1954), self-efficacy (Bandura, 1997), and value (Martin, 2007). These general indicators of motivation will be described below.

Engagement. Flow theory states that the natural curiosity activated in learners is vital for keeping learners intrinsically motivated (Egbert, 2003; Huang, Backman, & Backman, 2010). Engaging learning tasks are required in order to maintain flow (Csikszentmihalyi, 1975, 1990, 2000) within intrinsic motivation.

Behavioral engagement refers to learners' attention, effort, and persistence (Kim et al., 2017; Skinner, Kindermann, & Fuller, 2009). Contributing to the classroom, concentration, and observable effort constitute behavioral engagement (Skinner et al., 2009). The presence of behavioral engagement can be observed as on-task involvement and participation (Fredricks, Blumenfeld, & Paris, 2004; Kim et al., 2015, 2017; Skinner et al., 2009). A lack of behavioral engagement can be observed through learners' dearth of attention or expression of dissatisfaction with a task.

Cognitive engagement centers on a learner's investment in a task (Fredricks et al., 2004). Cognitive engagement is linked to the way in which learning tasks are structured, and the learning strategies involved (Kim et al., 2017). Motivation and self-regulated learning are related to cognitive engagement (Fredricks et al., 2004).

Emotional engagement refers to the positive or negative feelings learners have about the learning task which motivates students toward finishing learning tasks (Kim et al., 2017; Skinner et al., 2009). High emotional engagement has been shown to indicate motivated involvement in learners while low emotional engagement has been shown to indicate withdrawal from a learning task (Skinner et al., 2009). Engagement is linked to flow and, thus, also indicates intrinsic motivation (Martin, 2007, 2012).

Extrinsic motivation. Extrinsic motivation includes the motivating factors external to learning like awards, recognition, or punishments (Amabile et al., 1994; Law et al., 2010; Taylor, 1916). When people perform a task because of extrinsic motivation, it may not be because they take enjoyment in the task itself, rather they are focused on obtaining a reward (Cullen & Greene, 2011). High course grade aspirations and the desire to score well on projects are examples of extrinsic motivation in education (Glynn, Brickman, Armstrong, & Taasoobshirazi, 2011). Similarly, career aspirations and the drive to obtain the desired job represent extrinsic motivation (Glynn et al., 2011).

Interest. Interest plays an important role in motivation (Deci, 1992; O'Keefe & Harackiewicz, 2017). Interest is tied to the content of the learning task and reflects a level of increased attention and effort (Krapp, Hidi, & Renninger, 1992; Renninger & Hidi, 2011). In an academic context, Singh et al. (2002) note that engagement and interest are linked within motivation as engagement is "active involvement, commitment, and attention as opposed to apathy and lack of interest" (p. 324). Interest represents intrinsic motivation and is tied to flow theory (Chan & Ahern, 1999; Csikszentmihalyi, 1975, 1990, 2000; Yonghiu, 2010). Flow is a level of learning absorption which sustains learners' motivation over long periods of time (Csikszentmihalyi, 1975, 1990, 2000; Chan & Ahern, 1999; Yonghiu, 2010).

Intrinsic motivation. Intrinsic motivation is the internal drive people have to complete tasks based on personal desire (Deci & Ryan, 2000; Maslow, 1943; Skinner, 1954). When people demonstrate intrinsic motivation, they have a commitment to goal attainment based on an internal enjoyment in completing the task (Amabile et al., 1994; Law et al., 2010). Deci and Ryan (2000) link intrinsic motivation to self-regulation, persistence, and high performance, among other related indicators and outcomes.

Self-efficacy. Self-efficacy is one's belief in one's ability to succeed (Bandura, 1997) and signals highly versatile motivation (Bandura, 1997; Martin, 2007; Pajares, 1996). Self-efficacy is built through successes with experiences completing similar tasks related to the task at hand (Bandura, 1997). Learners who have high self-efficacy in relation to a learning task's content are likely to have more determination and adapt better in the face of adversity when experiencing initial difficulty with a learning task and follow-through (Bandura, 1997). Self-efficacy is an indicator of motivation and is also linked to expectancy-value (Martin, 2007).

Value. In learning theory, to what level learners believe that a task is useful, pertinent, and manageable to them is categorized as the general concept of value (Belland, Kim, & Hannafin, 2013). Task value is used to describe learners' perceptions of how important, interesting, and useful a task is (Wigfield & Eccles, 2000). Value promotes intrinsic motivation (Belland et al., 2013). Learners who perceive a learning task as having a high task value produce more effort toward completing the task at hand (Belland et al., 2013). Expectancy-value suggests that behavior is an outcome of the perceptions an individual has for their expected level of success combined with their perceptions of the value associated with completing the task (Fishbein & Ajzen, 1972).

Motivation models. Anderson and McLoughlin (2007) have remarked how today's programming students are impatient and expect immediate success while beginning to learn to program. Jenkins (2001) argued that students' motivation relating to programming could be divided into four categories: intrinsic, extrinsic, social, and achievement. Jenkins (2001) noted that many undergraduates are motivated by the extrinsic promise that learning programming will expand their money-making potential. However, Jenkins (2001) argued that intrinsic motivation was required for learners to successfully learn how to program. There are several frameworks for motivation, including those by Keller (1987), Svinicki (2010), and Vollmeyer and Rheinberg (2006). These frameworks are shared in this section.

Keller's (1987) ARCS model of motivation, for instance, is based on four components of motivation: attention, relevance, confidence, and satisfaction. In Keller's (1987) framework, attention can be harnessed by surprise or inquiry. Relevance can be formed by using real-world examples (Keller, 1987). Confidence can be created by showing a learner that they can succeed with the learning task (Keller, 1987). Satisfaction in Keller's (1987) framework links to a learner's feelings that the task is inherently rewarding. The ARCS framework points to attention, relevance, confidence, and satisfaction as factors that can promote and sustain a learner's motivation (Keller, 1987).

Svinicki (2010) touted a combined theory of motivation comprised of three factors: the value of the task, the ability to influence the outcome of the task, and self-efficacy. Value of task is based on multiple different factors, including (a) how interesting the task is to the learner, (b) the relationship between the long-term goals of the learner and the task, (c) the learner's perceived usefulness of the task, (d) how the task is valued by the learner's peers, and (e) how important others view the task (Svinicki, 2010). The ability to influence the outcome of the task is the learner's perception of if they can control the outcome of the task (Svinicki, 2010). A learner's self-efficacy is a learner's belief that they can succeed (McGill, 2012). Svinicki's (2010) combined theory of motivation aligns with the established theories of self-determination, expectancy-value, and behavioral, cognitive, and achievement goal orientation.

The cognitive-motivational model uses the expectancy-value model and has four factors of motivation (Vollmeyer & Rheinberg, 2006). These factors consist of the probability of success, anxiety related to failure, natural interest, and level of challenge (Vollmeyer & Rheinberg, 2006). Anxiety in the cognitive-motivational model is tied to fear of failure, while the challenge links to whether or not the learner wants to have success with the task are aligned to expectancy-value (McGill, 2012). The cognitive-motivational model factors work in combination with a learner's level of engagement and concentration (Vollmeyer & Rheinberg, 2006).

Motivation Related to Programming and Educational Robotics

Research indicates that participants with high levels of motivation spend more time on learning, engaging learning materials with higher intensity, cooperate more with peers, and are more open to learning and using new knowledge (Levin & Long, 1981;

Martin, 2007). Today's programming learners are motivated in ways unlike any other generation (Guzdial & Soloway, 2002; Trees, 2010). Literature supports the motivational impacts of educational robotics on novices learning programming in a variety of contexts (Apiola, Lattu, & Pasanen, 2010; Cheng, 2017; McGill, 2012; Osborne, Thomas, & Forbes, 2010; Petre & Price, 2004). For example, comparative research by Yamazaki et al. (2015) with a mixed middle and high school population reported that utilizing educational robotics increased positive responses to motivation questions compared to game-based programming application control data. Research by Kim et al. (2015, 2018) showed that preservice teachers must maintain high levels of intrinsic motivation to succeed while learning programming. The following paragraphs explain the current literature specific to preservice and in-service teachers' programming motivation and the impacts of educational robotics.

Teachers' programming motivation. Negative feelings new teachers develop about science concepts negatively influence their ability to become effective teachers (Appleton, 2003; Bryan, 2003; Davis, Petish, & Smithey, 2006). Various modalities for motivating novice programmers who may be struggling with programming have been investigated, from multimedia modalities to educational robotics (Kolling & Rosenberg, 2001; Rich, Perry, & Guzdial, 2004; Yamazaki et al., 2015). McGill (2012) pointed out, "It is important to investigate empirically whether or not learning environments actually have an effect on student motivation since many of these systems were built for that specific purpose" regarding different products for programming motivation (p. 2). Nevertheless, numerous researchers studying the motivational effects of educational robotics did not define motivation or provide details about their instruments' questions, validity, and reliability (Adams, 2010; Cliburn, 2006; Lauwers, Nourbakhsh, & Hamner, 2009). Thus, previous research pertaining to motivation specific to preservice teachers is indistinct.

Several researchers have recently studied educational robotics and programming with in-service and preservice teacher populations (Jaipal-Jamani & Angeli, 2017; Kaya et al., 2015; Sisman & Kucuk, 2019). Educational robotics interventions have been effectively used to enhance preservice teachers' motivation to integrate programming into their curricula (Jaipal-Jamani & Angeli, 2017; Kaya et al., 2015). Jaipal-Jamani and Angeli's (2017) study reported that over 85% of their preservice teacher participants were motivated to use robotics in their teaching. Similarly, Kaya et al.'s (2015) study exploring the views of 11 preservice teachers on engineering concepts reported that 100% of their participants decided to integrate block-based programming and educational robotics into their elementary science classes. A study by Sisman and Kucuk (2019) adds that preservice teachers were most motivated by educational robotics and the idea that they could learn to teach their future students how to program educational robots.

Teachers' programming motivation based on motivational indicators. Studies have demonstrated improvements to in-service and preservice teachers' motivation through educational robotics interventions (Jaipal-Jamani & Angeli, 2017; Kay et al., 2014; Kim et al., 2015; Kucuk & Sisman, 2018; Osborne et al., 2010; Perritt, 2010). For instance, Kay et al. (2014) found that in-service teachers' confidence in their programming skills increased in a statistically significant manner after they completed educational robotics activities, including robot construction and programming. Perritt (2010) concluded that confidence built through educational robotics activities increased preservice and in-service teachers' motivation to implement educational robotics and programming into their instruction. Sullivan and Moriarty (2009) found that educational robotics instruction improved in-service teachers' perceptions of the value of programming educational robotics in the classroom, implying that participants are motivated to utilize programming in the classroom. For preservice teacher populations, research indicated that developing self-confidence with programming educational robotics is the key to motivating preservice teachers to use programming (Kim et al., 2015; Osborne et al., 2010). Similarly, several researchers have shown that preservice teachers' engagement and confidence in STEM concepts increased after being involved in educational robotics activities (Jaipal-Jamani & Angeli, 2017; Kim et al., 2015; Kucuk & Sisman, 2018). Furthermore, preservice teachers' interest and self-efficacy in STEM concepts increased after they completed educational robotics activities (Adams et al., 2014; Jaipal-Jamani & Angeli, 2017; Kim et al., 2015; Ortiz et al., 2015).

Measuring Motivation

Motivation has many interrelated indicators (Bandura, 1997; Dewey, 1913; Martin, 2007; O'Keefe & Harackiewicz, 2017; Singh et al., 2002). In this section, general instruments for gathering data on motivation in education will first be described. Then, more specific instruments that have been designed to evaluate programming motivation will be shared.

Educational motivation instruments. Numerous instruments exist for measuring participants' general motivation in relation to the field of education. Students' motivation can be measured with the Motivated Strategies for Learning Questionnaire, or MSLQ (Pintrich, 1999; Pintrich & De Groot, 1990). Landry's (2003) Student Motivation Scale includes items inspired by Pintrich and De Groot (1990) to measure undergraduate students' motivation to complete their studies in the face of obstacles (Martin, 2003). Similarly, Sinclair, Downson, and McInerney (2006) devised the Motivational Orientations to Teach Survey, or MOT-S, which includes 80 motivational questions aimed to assess the teaching motivation of preservice teachers. Other motivation instruments include the Questionnaire of Current Motivation, which is designed to measure initial motivational and uses the cognitive-motivational factors of the probability of success (Vollmeyer & Rheinberg, 2006), anxiety related to failure, natural interest, and level of challenge (Rheinberg, Vollymeyer, & Burns, 2001). Keller's (1983, 1987) ARCS-based Instructional Materials Motivation Survey instrument measures the impact of integrating a tool designed for increasing motivation into one's instruction. Glynn et al. (2011) created the Science Motivation Questionnaire II, which evaluates the general science motivation of college learners through the subscales of intrinsic motivation, self-determination, self-efficacy, career motivation, and grade motivation.

Evaluating motivation towards programming. Specialized instruments directly related to programming concepts and educational robotics have been inspired by the more general motivation instruments described above. This section will first highlight qualitative measures of programming motivation. Then, this section will describe quantitative measures of programming motivation.

Qualitative measures. There are different qualitative measures for motivation related to programming and educational robotics (Jaipal-Jamani & Angeli, 2017; Kaya et al., 2015; Kim et al. 2015; Kucuk & Sisman, 2018; Yadav et al., 2014). Kim et al. (2015) used an adapted version of Black and Deci's (2000) learning self-regulation questionnaire, or SRQ-L, to measure autonomous and controlled motivation. In the study, Kim et al. (2015) used surveys and interviews to gather data on preservice teachers' motivation while using educational robotics. Yadav et al. (2014) measured preservice teachers' motivation to integrate computational thinking programming exercises into their future classrooms by using open-ended questions focusing on three categories, including computational thinking, the relationship of computational thinking to other disciplines, and integrating computational thinking into the classroom. Similarly, Kaya et al. (2015) studied preservice teachers' experiences in learning programming through educational robotics and measured participants' motivation through qualitative data gathered through reflective essays. Kucuk and Sisman (2018) gathered data on preservice teachers' motivation through interview questions like "How have you felt cognitively and emotionally while working on the robotics programming activities?" (p. 307). Jaipal-Jamani and Angeli (2017) studied preservice teachers' interest and self-efficacy relating to programming concepts and robotics. In this study, Jaipal-Jamani and Angeli (2017) utilized a questionnaire about participants' self-efficacy with computational thinking and robotics as well as a questionnaire in which participants self-rated their confidence with teaching block-based programming educational robotics lessons. Ortiz et al. (2015) gathered qualitative data on preservice teachers' motivation during educational robotics activities through observations, participants' comments, and reflective essays.

Quantitative approaches. Other studies have taken quantitative approaches to investigate the effects of educational robotics on motivation (McGill, 2013; Wang, Mei, Lin, Chiu, & Lin, 2009). A prime example is McGill's (2013) instrument, which borrows aspects of Keller's (1987) ARCS model and Wiedenbeck's (2005) computer self-efficacy scale. McGill's (2013) instrument is comprehensive and is specialized for educational robotics motivation. McGill's (2013) study measured four components of motivation: attention, relevance, confidence, and satisfaction. This instrument investigated the motivational effects of educational robotics on a population of non-computer science majors using Wiedenbeck's (2005) computer programming self-efficacy scale measured the confidence of participants as they completed programming tasks. McGill (2012) measured motivation through quantitative data gathered with Keller's (1987) instructional materials motivation survey. Other examples are the Wang et al.'s (2009) motivation questionnaire and experience questionnaire. The motivation questionnaire evaluated students' feelings related to programming motivation before and after instruction and includes the three subscales of motivation to learn programming, self-efficacy, and perception of programming (Wang et al., 2009). The experience questionnaire, which was given after instruction, included two subscales for classroom experience and classroom atmosphere (Wang et al., 2009).

Chapter Summary

This literature review examined applicable literature on the topics of programming in K-12 education, educational robotics, comprehension, and motivation. Programming is the process of designing and creating special instructions for computers to run, known as programs (Ceruzzi, 1998). Block-based programming languages can help propel novices past the traditional difficulties of text-based programming languages in order to explore abstract computer science concepts quickly (Bers et al., 2014; Kim et al., 2018; Lye & Koh, 2014; Malan & Leitner, 2007; Wilson & Moffat, 2010). Although block-based programming has demonstrated positive motivational effects with preservice teachers (Yadav et al., 2011; Yadav et al., 2014), programming is still inherently abstract. Both in-service and preservice teachers attribute their lack of confidence toward teaching computer science content to their perspectives that programming is difficult and abstract (Bower et al., 2017; Grover & Pea, 2013; Ortiz et al., 2015; Resnick et al., 2009). Educational robotics have been shown to make learning abstract concepts more concrete (Altin & Pedaste, 2013; Barker et al., 2014; Mikropoulos & Bellou, 2013; Nugent et al., 2010). Block-based programming and educational robotics pair well together because of the constructible nature of each medium (Dagdilelis et al., 2005; Staszowski & Bers, 2005). Numerous studies undergird the benefits of pairing educational robotics with programming (Bers et al., 2002; Bers & Ponte, 2005; Huang et al., 2013). Commonly, educational robotics practices use robots as mindtools (Jonassen, 2000) within constructivist and constructionist learning frameworks (Alimisis, 2013; Kucuk &Sisman, 2018; Mikropoulos & Bellou, 2013). Programming comprehension is the ability to predict what a program will do by utilizing mental models (Ala-Mutka, 2004) and includes syntactic, semantic, and strategic knowledge (Bucks, 2010; Mayer, 1979; McGill & Volet, 1997). The effects of educational robotics on the programming comprehension of in-service (Kay et al., 2014; Sullivan & Moriarty, 2009) and preservice teachers (Eguchi, 2013; Jaipal-Jamani & Angeli, 2017; Kim et al., 2018; Kucuk & Sisman, 2018) have been studied with varying results. Qualitative measures of preservice teachers' programming comprehension (Eguchi, 2013; Kim et al., 2018; Kucuk & Sisman, 2018) are more common than quantitative measures (Jaipal-Jamani & Angeli, 2017). Recent studies of in-service and preservice teacher populations have shown that educational robotics can be motivational (Jaipal-Jamani & Angeli, 2017; Kaya et al., 2015; Sisman & Kucuk, 2019). There are numerous education-specific motivational instruments, but few tailored to programming education (McGill, 2013; Wang et al., 2009). In conclusion, educational robotics can be used to make abstract concepts like programming more concrete (Altin & Pedaste, 2013; Barker et al., 2014; Mikropoulos & Bellou, 2013; Nugent et al., 2010) and have been shown to have motivational effects with teacher populations (Jaipal-Jamani & Angeli, 2017; Kaya et al., 2015; Sisman & Kucuk, 2019).

CHAPTER 3

METHOD

The purpose of this action research was to evaluate the effect of educational on the programming comprehension and motivation of preservice teachers at a medium-sized liberal arts university in the southeastern United States. The research questions for this study were:

1. What is the effect of educational robotics on preservice teachers' comprehension of programming concepts?
2. How and to what extent does educational robotics influence preservice teachers' motivation related to programming?

Research Design

This study utilized action research. According to Mertler (2017), action research is typically carried out by practitioners with a "vested interest in the teaching and learning process" of a specific population and setting (p. 4). The main advantage of action research is its specificity. Greenwood and Levin (2007) described action research as "context bound" (p. 63). This means that action research is specific to the class and participants taking part in the study (Creswell, 2014; Mertler, 2017; Rudestam & Newton, 2007). Action research fits my context because I was not only the researcher in this study but also the instructor. I had a highly contextualized problem specific to my course that needed to be addressed. Although the results of an action research study such as mine cannot be widely generalized to other instances and settings, the results of the study are tailored to the research questions and environment being investigated. Further, an action research type intervention is more appropriate for my teaching context than a true experimental design with control and experimental treatments. In my action research study, all the participants received the benefits of the study. What differentiates action research from more traditional lines of inquiry are both the process and the end goal (Mertler, 2017). While traditional lines of inquiry are typically performed by outsiders withdrawn from the study's subjects with the goal of documenting teaching or learning, action research is typically performed by insiders, such as myself, in collaboration with the participants being studied with the end goal of improving teaching and learning (Zeni, 1998). Accordingly, the goal of this action research was designed to pinpoint actionable steps to improve teaching practices and student outcomes.

Greenwood and Levin (2007) described one advantage of action research as it is a "pragmatic" system to solve "real-life problems holistically" (p. 63). Mertler (2017) affirmed that action research solves problems holistically by stating that action research tends to align more harmoniously with mixed methods than with singularly qualitative or quantitative strategies. In addition, Morgan (2014) explained that mixed methods fit best with a pragmatist paradigm. As mentioned in the Researcher Subjectivities and Positionality section of this dissertation, my personal paradigm aligns with a pragmatist standpoint. Thus, mixed methods were selected for this study to provide a holistic and best-aligned method for evaluating the research questions. While the quantitative data in this study were employed to point toward the effect of the intervention on programming comprehension and motivation, qualitative data were harnessed to report the experiences and opinions of the participants. Analyzing two different forms of data, Mertler (2017) argued, “leads to greater credibility in the overall findings” (p. 107). By analyzing two different styles of data, I was able to discover information that would have otherwise been overlooked if only one data collection method was utilized. For my study, mixed methods merged quantitative data and qualitative data, which eliminated biases of a single data collection method, which showed the full picture of the phenomena at hand (Creswell, 2014). The mixed methods design was chosen so I could triangulate if the data gathered from the motivation survey are more complex than one data collection style would detect (Almalki, 2016). Triangulation is a process of corroboration using evidence from different sources, different types of data, or different methods of data collection (Buss & Zambo, 2014; Creswell, 2014; Patton, 2002). I compared data side-by-side from the surveys and individual interviews to determine if the quantitative data supported the qualitative data.

I utilized a convergent parallel mixed methods design for my action research. Creswell (2014) explained convergent parallel mixed methods design as a technique in which the researcher gathers quantitative and qualitative data at the same time then analyzes the results of the study separately in order to see if the triangulation of results “confirm or disconfirm” each other (p. 219). The first reason convergent design was used for this study is outlined by Creswell and Plano Clark (2018), who have described convergent design as an intuitive and efficient strategy for researchers new to performing mixed methods. Another reason convergent design was used in this study was time. Creswell and Plano Clark (2018) noted that convergent parallel mixed methods are often used when the researcher “has limited time available for collecting data in the field.” (p. 68). The small window of time available to dedicate to this study within the class schedule necessitated the convergent accumulation of quantitative and qualitative data. Further, convergent design enabled me to compare participants' feelings gathered through qualitative questioning with the data gathered from my standpoint through surveys (Creswell, 2014; Creswell & Plano Clark, 2018). Coming full circle, Creswell and Plano Clark (2018) linked the pragmatic nature of action research described by Greenwood and Levin (2007) to convergent design with the statement, "assumptions of pragmatism are well suited for guiding the work of merging the two approaches [quantitative and qualitative] into a larger understanding" (p. 69). Because this study utilized both surveys and individual interviews to analyze motivation, the perspectives of both the participants and I were united.

Setting and Participants

This study took place at a medium-sized liberal arts university in the southeastern United States. This study occurred within an educational technology course that preservice teachers must take to graduate as education majors. In this course, students were taught how to utilize computers, multimedia, mobile technologies, interactive whiteboards, apps, and websites, among other educational technologies. There were no prerequisite classes for this course. Therefore, students came into the course with various levels of experience with technology. The setting of this study was a large digital learning lab complete with personal computers for each student, a SmartBoard, two projectors, and associated screens. In addition to the computer clusters offered in this room, there were spaces for collaboration activities in the room. There were 12 Lego EV3 robotics kits for the class along with 24 laptops with the Lego programming software, so each student could write his or her own programs. Each laptop was Bluetooth enabled in order to communicate the programs to the Lego EV3 robots.

This study included a purposeful sample of participants. As Creswell (2014) explained, purposeful sampling allows the researcher to select the participants who will "best help the researcher understand the problem and the research question" (p. 189). The inclusion criteria stipulated that the participants needed to be preservice teachers with education majors. Therefore, out of the two sections of the course taught by me, the section of the course with the fewest non-education major students was selected to preserve the highest population value for the study. Out of the 23 students in the class, there were two non-education majors whose data were removed from the study to avoid threats to validity. Of the eligible 21 education majors, three participants dropped out of the class during the study. These participants' data were removed prior to analysis. An ultimate total of 18 undergraduate preservice teachers made up the sample for this study. As shown in Table 3.1, these undergraduate preservice teacher participants represented all the education majors offered by the university: early childhood education (2), elementary education (9), middle level education (3), special education (2), and physical education (2). The participants included 15 females and three males. The participants' ages ranged from 18 to 23, with a mean age of 19 (*SD* = 1). The participants included freshmen (6), sophomores (11), and one junior. Four of the participants reported their technology comfort level as basic, 12 intermediate, and two advanced. Only one participant had limited prior programming experience and prior programming instruction. Two participants reported having limited prior experience programming a robot and prior robotics instruction.

Table 3.1. *Participants' Demographic Information*

Age	Gender	Classification	Education Major	Tech. Comfort Level	Prog. Exp.	Robo. Exp.
19	Female	Sophomore	Elementary	Intermediate	No	No
18	Female	Sophomore	Elementary	Intermediate	No	No
21	Female	Junior	Elementary	Intermediate	No	No
19	Female	Sophomore	Elementary	Basic	Yes	Yes
19	Female	Sophomore	Elementary	Advanced	No	No
20	Female	Sophomore	Special	Intermediate	No	No
23	Female	Sophomore	Physical	Intermediate	No	No
18	Female	Freshman	Elementary	Intermediate	No	No
19	Female	Freshman	Early Childhood	Basic	No	No
18	Female	Freshman	Early Childhood	Basic	No	No
19	Male	Sophomore	Physical	Intermediate	No	No
19	Female	Sophomore	Elementary	Intermediate	No	No
18	Female	Freshman	Middle	Basic	No	No
20	Male	Sophomore	Middle	Intermediate	No	No
20	Female	Sophomore	Special	Intermediate	No	No
18	Female	Freshman	Elementary	Intermediate	No	No
18	Female	Sophomore	Elementary	Advanced	No	No
18	Male	Freshman	Middle	Intermediate	No	Yes

Note. Prog. Exp. means programming experience and Robo. Exp. means robotics experience.

Intervention

This study utilized an educational robotics intervention that spanned four weeks of lessons. The lessons included in this intervention used mindtools to teach programming through a constructivist framework (Jonassen, 2000; Piaget, 1967) in a collaborative environment. These lessons were inspired by a robotics curriculum previously developed by the research setting's physics and education faculty, including myself. This robotics curriculum was created as part of a federal No Child Left Behind Improving Teacher Quality Higher Education grant for a grant titled PRISM – Partnership for Robotics Integration using Science and Math (South Carolina Commission on Higher Education, 2016). Activities and challenges were abridged and tailored to the specific goal of teaching programming through robotics. Lego EV3 robots running the EV3-G block-based programming language were chosen for this intervention because of Lego robotics' popularity in schools at the K-8 levels (Martin et al., 2000; Martin et al., 2011; Martin & Resnick, 1993). Participants were paired randomly for the intervention. Marzano (2007) recommended cooperative pairs for learning activities involving problem-solving in order to allow learners to collaboratively discuss and reflect upon the problems they are given. Classes met twice per week for one hour and fifteen minutes per period. Each lesson was aligned to both the South Carolina Computer Science and Digital Literacy Standards for grades K – 8 (South Carolina Department of Education, 2017) as well as course standards.

The robotics intervention was divided into four week-long units. These units were (1) Basic Procedures, (2) Advanced Procedures, (3) Control Structures, and (4) Variables. This sequence of these units was based on the robotics curriculum created as part of a PRISM grant (South Carolina Commission on Higher Education, 2016). The Basic Procedures unit focused on the core syntactic programming skills needed to write functional programs. The Advanced Procedures unit focused on semantic and strategic programming skills needed to write programs which navigated the robots around obstacles. The Control Structures unit focused on writing programs utilizing flow control based on predetermined parameters, such as if/then statements and loops. The Variables unit focused on integrating variables into the flow control of advanced programs. These units are shown in Table 3.2 with two main topics per unit. Each unit consisted of demonstrations, learning activities, and challenges. These units will be described in detail in the following sections.

Table 3.2. *Robotics Intervention Units*

Unit	Topics
Basic Procedures	Syntactic knowledge of the programming language Odometry Programming for seconds/rotations/degrees
Advanced Procedures	Semantic programming knowledge Pseudocoding Strategic programming knowledge Programming turning
Control Structures	Flow control Loops If/then statements
Variables	Variables Combining variables with control structures

Intrinsic motivation involved learners' desire to learn about the topic due to their own internal self-interests (Eccles, Simkins, & Davis-Kean, 2006; Ryan & Deci, 2000, 2020). Researchers have shown that physically interacting with robots can impact intrinsic motivation (Apiola et al., 2010). Likewise, problem-solving, as found in the challenges, has been shown to impact intrinsic motivation (Kucuk & Sisman, 2018).

Career motivation is an idea that posits that learners who demonstrate motivation in a subject see that subject's relevance to their future careers (Arwood, 2004; Glynn, Taasoobshirazi, & Brickman, 2009). Career motivation aligned with the instructional portion of the lessons where participants were explained how to write programs and how programming concepts could be integrated into their future teaching.

Self-determination has been defined by Black and Deci (2000) as the control learners have over their learning. Similarly, self-efficacy is described as students' confidence in their ability to achieve the learning task (Bandura, 1997; Lawson, Banks, &

Logvin, 2007). Self-determination is brought about through confidence-building (Ryan & Deci, 2000, 2020), and self-efficacy is brought about through learners experiencing success (Bandura, 1997). These two categories of motivation aligned with the learning activities and challenges in the lessons, which could boost learners' confidence through success.

Motivation to Integrate Programming into Teaching (MTIPIT) was built on previous research about teacher motivation, which included a combination of intrinsic, extrinsic, and altruistic factors (Brookhart & Freeman, 1992; Han & Yin, 2016; Sinclair, 2008). MTIPIT encompassed learners' feelings about including programming instruction and activities in their professional teaching, built through their experiences with all the different aspects of the programming lessons (Brookhart & Freeman, 1992; Han & Yin, 2016; Sinclair, 2008).

The units of the Programming Motivation Survey were aligned to the various aspects of the lesson plans, as delineated in Table 3.3.

Table 3.3. *Programming Motivation Survey Subscale and Lesson Aspect Alignment*

Subscale	Lesson Aspect
Intrinsic Motivation	Using robots Learning activities Challenges
Career Motivation	Programming instruction Lectures on programming integration
Self-Determination	Learning activities Challenges

Table 3.3. *Programming Motivation Survey Subscale and Lesson Aspect Alignment* Continued.

Subscale	Lesson Aspect
Self-Efficacy	Learning activities Challenges
MTIPIT	Programming instruction Lectures on programming integration Using robots Learning activities Challenges

Basic Procedures

The first week focused on basic programming procedures. In this unit, participants became familiar with how programs are composed. As outcomes of these lessons, their associated activities, and challenges, participants were able to test and debug a program, create functioning programs, calculate values for programs, and used three different methods of programming to solve a problem. Table 3.4 details the alignment of the lesson plans to state standards, and the course's student learning outcomes.

Table 3.4. *Basic Procedures Lesson Plan Alignment*

Lesson Plan	SC State Computer Science Standard	Lesson Objectives
Basic Procedures Class 1	Standard 1: Recognize that many daily tasks can be described as step-by-step instructions (i.e., algorithms). Standard 4: Develop a program to express an idea or address a problem	Test and debug a program Create a functioning program

Table 3.4. *Basic Procedures Lesson Plan Alignment* Continued.

Lesson Plan	SC State Computer Science Standard	Lesson Objectives
Basic Procedures Class 2	Standard 1: Recognize that many daily tasks can be described as step-by-step instructions (i.e., algorithms). Standard 4: Develop a program to express an idea or address a problem	Calculate values for a program Use different methods of programming to solve a problem

During the first class of the Basic Procedures unit, participants were familiarized with the syntax of the programming language and given step-by-step instructions for writing programs with different methods in the EV3-G block-based programming language. The instructor highlighted the functionality and customizability of each type of programming block throughout the presentation. Instructional possibilities and curricular connections with science and math were highlighted. The instructor demonstrated programming functions on an example robot. Participants were instructed to follow along throughout the training and write and execute programs, as shown by the instructor when appropriate. The instructor demonstrated a basic debugging process. Then, participants were given free time in their pairs to experiment with the robots and become comfortable with programming them. As an exit ticket for dismissal, participants shared one discovery their pair made while programming their robot during the experimentation time. More details on this class period's activities are in a lesson plan, as Figure A.1 in Appendix A.

In the next class period, the formal in-class robotics programming activities began. Participants were introduced to odometry and calculating values for their programs. Participants learned how odometry could be used to solve problems. Pairs of participants first used trial and error and then used odometry in their programs. Once participants completed the odometry activity, they were given a challenge. For this challenge, they were instructed to program their robots to travel one meter using three different programming methods. Their programs must move the robots based on (1) an amount of time, (2) revolutions, and (3) degrees. For full details on this class period, see the lesson plan located in Figure A.2 in Appendix A. An example solution for the One Meter Challenge is available as Figure A.3 in Appendix A.

Advanced Procedures

The second week focused on more advanced programming procedures. In this unit, participants became familiar with more customized programs designed to accomplish specific tasks. As outcomes of these lessons, their associated activities, and challenges, participants were able to predict the outcome of a program, modify a simple program, and create a program to solve a problem. Table 3.5 details the alignment of the lesson plans in this unit to state standards and course student learning outcomes.

Table 3.5. *Advanced Procedures Lesson Plan Alignment*

Lesson Plan	SC State Computer Science Standard	Lesson Objectives
Advanced Procedures Class 1	Standard 1: Design, evaluate, and modify simple algorithms (e.g., steps to make a sandwich; steps to a popular dance; steps for sending an email).	Predict the outcome of a program Modify a simple program

Table 3.5. *Advanced Procedures Lesson Plan Alignment* Continued.

Lesson Plan	SC State Computer Science Standard	Lesson Objectives
Advanced Procedures Class 2	Standard 3: Decompose problems into subproblems and write code to solve the subproblems (i.e., break down a problem into smaller parts).	Predict the outcome of a program Create a program to solve a problem

The first class of the Advanced Procedures unit focused on more difficult programming, including turning. Participants were introduced to pseudocode. Then, participants were presented with step-by-step instructions for writing programs for turning the robots using the block-based programming editor and the EV3-G programming language. The instructor highlighted the functionality and customizability of each type of programming block throughout the presentation, as well as instructional possibilities and curricular connections. The instructor demonstrated the different programming functions for turns on an example robot. Based on given program examples, participants predicted the outcome of programs before they were performed by the robot. Participants wrote more advanced programs to make their robots follow lines through courses designed with colored tape, as illustrated in Figure 3.1. After this instruction, pairs worked on a learning activity in which they modified a given program in order to move their robots around the box that their robots came in. For full details on this class period, see the lesson plan located in Figure A.4 in Appendix A. For a potential programming solution to the challenge for this lesson, see Figure A.5 in Appendix A.

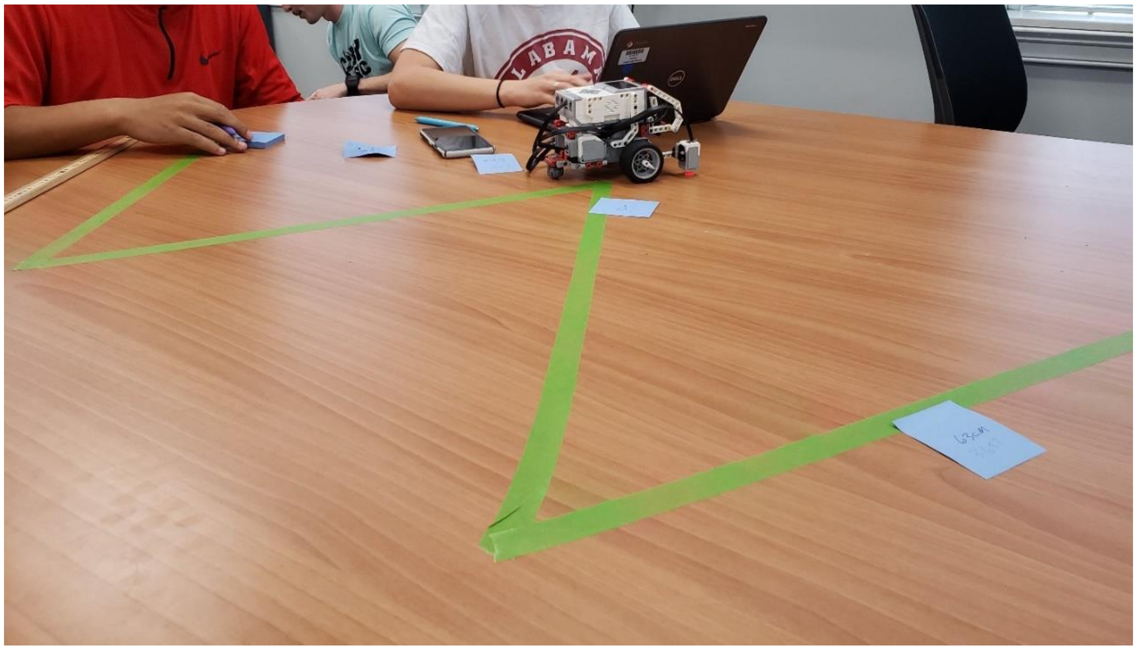


Figure 3.1. Line following activity

The second class of the Advanced Procedures unit began with a pseudocode warmup activity. In this activity, students designed paper airplanes and then wrote instructions for a partner to create an identical model. Throughout this activity, participants learned how exact their algorithms needed to be for the computer to execute them when they are writing advanced programs properly. The next part of the class period revolved around a challenge. To begin, the instructor led the students in a pseudocode demonstration for following a path. Then, the challenge was introduced. In the challenge, pairs programmed their robots through a maze made from electrical tape. Before placing their robot in the maze, partners were required to write their programs from a schematic and calculations lens, as shown in Figure 3.2. Once partners showed the instructor their program, they could run it in a maze and make necessary modifications. There were multiple copies of the maze set up on the floor throughout the classroom and neighboring hallway, as displayed in Figure 3.3, so multiple pairs of students could share each maze in order to ensure efficiency. For full details on this class period, see the lesson plan located in Figure A.6 in Appendix A. A schematic for the maze is available in Appendix A as Figure A.7.

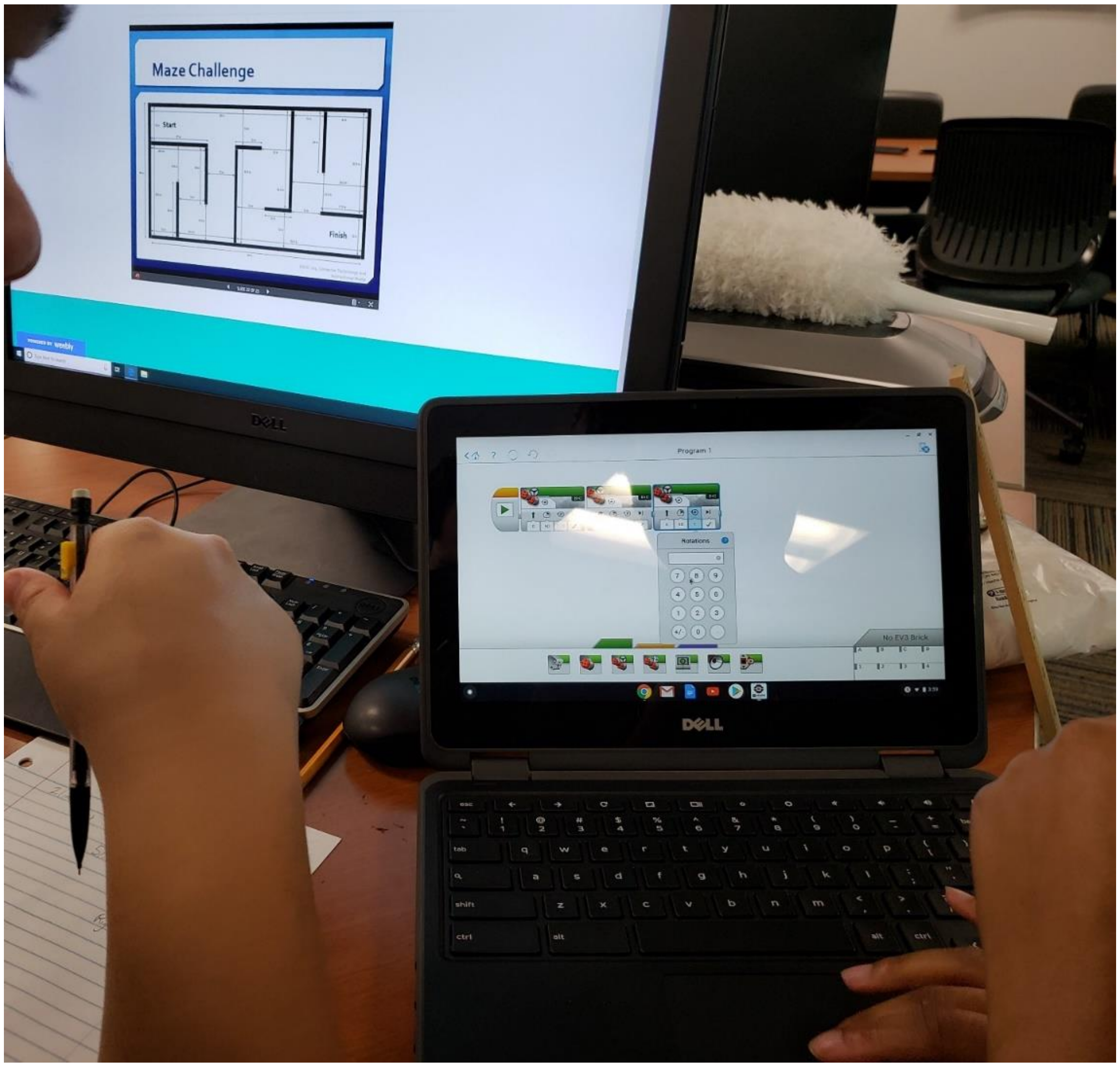


Figure 3.2. Partners write a maze program.

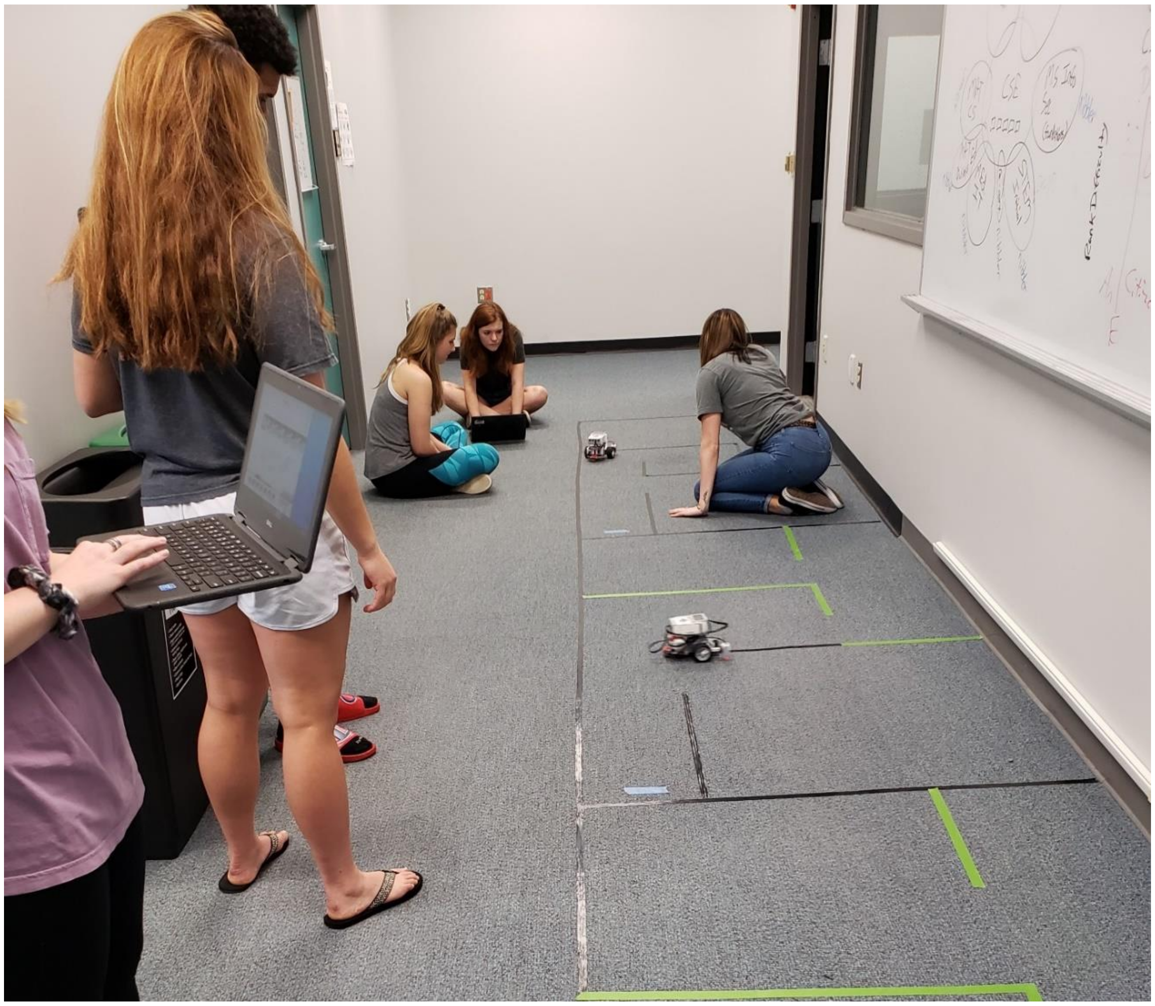


Figure 3.3. Participants test their programs in the mazes.

Control Structures

The third week of the robotics intervention focused on the programming of different control structures. In particular, the participants were introduced to programming loops and if/then statements. As outcomes of these lessons, their associated activities, and challenges, participants were able to predict the outcome of programs, create programs using control structures, and modify programs using control structures.

Table 3.6 details the alignment of the lesson plan to state standards and course student learning outcomes.

Table 3.6. *Control Structures Lesson Plan Alignment*

Lesson Plan	SC State Computer Science Standard	Lesson Objectives
Control Structures 1	Standard 2: Use and compare simple coding control structures (e.g., if-then, loops).	Predict the outcome of a program that uses control structures Create a program using control structures
Control Structures 2	Standard 2: Use and compare simple coding control structures (e.g., if-then, loops).	Modify a simple program using control structures Create a program using control structures

During the first class of the Control Structures unit, participants were presented with information on what control structures are and how they control the flow of programs. Then, participants were given step-by-step instructions for writing loops into programs using the block-based programming editor. The instructor highlighted the functionality and customizability of different types of loops throughout the presentation. Instructional possibilities for looping and curricular connections for control structures, in general, were identified. The instructor demonstrated the different programming functions on an example robot, and participants predicted the actions of the robot based on the given loops in the program. The learning activity for this unit required pairs to program their robots to move in a slithering motion, making a hissing sound at the end of the program after the required loops. For full details on this class period, see the lesson plan located in Figure A.8 in Appendix A. An example programming solution to the Slithering One Meter Challenge is available as Figure A.9 in Appendix A.

In the second class of the Control Structures unit, the instructor reinforced the utilization of control structures by providing more details on loops and if/then statements. Videos on different programming connections to different subjects were shared by the instructor. Then, the Lap Loop Challenge was given to participants. In this challenge, pairs modified their Lap Activity programs by deleting superfluous programming, which could be written in a more succinct fashion with loops. The objective was to modify their programs in order to successfully move their robot around their box three times using the loop, playing a different sound after each loop was completed. For full details on this class, please see the lesson plan located in Figure A.10, and the potential solution to the Lap Loop Challenge demonstrated in Figure A.11 in Appendix A.

Variables

The fourth week of the robotics intervention focused on how variables were used in programming. Participants learned that variables are containers for changing value information in programs. This unit also introduced the color sensor. As outcomes of these lessons, their associated activities, and challenges, participants were able to predict the outcome of a program based on given variables, create a program using variables, and modify a program using variables. Table 3.7 details the alignment of the lesson plan to state standards and course student learning outcomes.

Table 3.7. *Variables Lesson Plan Alignment*

Lesson Plan	SC State Computer Science Standard	Lesson Objectives
Variables Class 1	Standard 5: Identify variables and compare the types of data stored as variables.	Predict the outcome of a program based on the given variables. Create a program using variables.
Variables Class 2	Standard 4: Design and code programs to solve problems Standard 5: Identify variables and compare the types of data stored as variables.	Create a program using variables. Modify a program using variables.

The first class of the Variables unit began with an overview of the color sensor. First, the instructor demonstrated how the color sensor was used. Participants were presented with step-by-step instructions for writing programs using variables inside if/then statements in the block-based programming editor. The instructor highlighted the functionality of the color sensor and how it could be used with the different types of programming blocks related to variables, like the variables block, the math block, and the read numeric and write numeric settings. Throughout the presentation, curricular connections and instructional possibilities were shared. The instructor demonstrated the color sensor on an example robot. Then, pairs wrote programs utilizing the color sensor that scanned colors, incrementing a variable each time a predetermined color was detected by the sensor. The instructor then introduced the Red Light Activity. In the learning activity, pairs programmed their robots to speed up when the color sensor detects blue (increasing the speed variable each time), and stop the robot when the color sensor detects red. For full details on this class, see the lesson plan located in Figure A.12 in

Appendix A in addition to the schematic for the Red Light Activity available in Figure A.13.

For the final robotics class, the Variables unit's Color Maze Challenge was shared. For this challenge, the mazes utilized in the Maze Challenge were modified. Red pieces of tape were added to the mazes at points where the robots needed to turn right. Green pieces of tape were added to the mazes at points where the robots needed to turn left. The criteria for the Color Maze Challenge stipulated that every time the robots encountered a red line, they turned right and every time they encountered a green line they turned left and increment a variable by one on the Lego EV3's screen using a variable and the formula (x + 1). The walls of the maze and the finish line were made of black tape, so the robots needed to be programmed to stop if they detected the black tape. Students completed this activity when they successfully navigated their robots to the end of the maze using programming, which utilized movement, control structures, and variables. For more details, see the lesson plan located in Figure A.14 in Appendix A. A schematic for this maze is included in Appendix A as Figure A.15. An example solution for this challenge is also available in Figure A.16 in Appendix A.

Data Collection Methods and Data Sources

Multiple sources of data were utilized to inform the results of this study. These sources were (1) Programming Comprehension Assessment, (2) Programming Motivation Survey, (3) field notes, and (4) individual interviews. Each research question and its associated data sources are represented in Table 3.8. The data sources used in this study are described in detail in the paragraphs below.

Table 3.8. *Research Questions and Data Sources Alignment*

Research Questions	Data Sources
RQ#1: What is the effect of educational robotics on preservice teachers' comprehension of programming concepts?	Pretest and posttest Programming Comprehension Assessment
RQ#2: How and to what extent does educational robotics influence preservice teachers' motivation related to programming?	Pre-instructional and post-instructional Programming Motivation Survey
	Field notes
	Individual interviews

Programming Comprehension Assessment

To assess the construct of programming comprehension, participants completed the researcher-created pretest and posttest Programming Comprehension Assessment found in Appendix B once before the intervention began, and once immediately after the intervention concluded. The pretest and posttest data allowed me to determine participants' comprehension of programming concepts. The assessment was constructed of 20 questions and divided into four subsections with five questions each. Each of the subsections was aligned to the four units of instruction: (1) Basic Procedures, (2) Advanced Procedures, (3) Control Structures, and (4) Variables. As demonstrated in Appendix C, each question was aligned to a South Carolina Computer Science and Digital Literacy Standard (South Carolina Department of Education, 2017) as well as a lesson objective from each lesson.

The questions prompted participants to read, debug, differentiate, problem-solve, and arrange portions of programs. The first five questions focused on basic procedures in programming. For example, participants were asked to arrange pieces of a program so that the program worked and successfully moved the robot. In the Advanced Procedures subsection, participants were asked to predict the outcome of a program, modify a program, or create a program that solved a problem using blocks of programming that included turns. For example, participants were asked to predict where a robot running a given program would end in relation to its starting location after executing the given program. The next subsection aligned with the Control Structures unit of instruction. This subsection focused on utilizing loops and if/then statements to build programs. For example, participants were asked to simplify a program using loops. In this section, for example, participants were asked to choose the string of programming which included variables to produce the desired results. Each question was graded on a nominal scale as either correct or incorrect (Devlin, 2017). Each correct answer was worth one point for a total of 20 possible points. The Programming Comprehension Assessment was designed to take about 30 minutes to complete. The instrument was validated by two experts in programming and robotics (see the full feedback from each reviewer in Appendix D). One expert was part of the team that created the South Carolina K-8 computer science standards while the other is a physics professor and president of a state-wide Lego robotics league. Through the validation process, updates were made to the Programming Comprehension Assessment based on the experts' suggestions. An example of such feedback is exhibited in Figure 3.4. For the result of this feedback, review question #18 in the final Programming Comprehension Assessment in Appendix B.

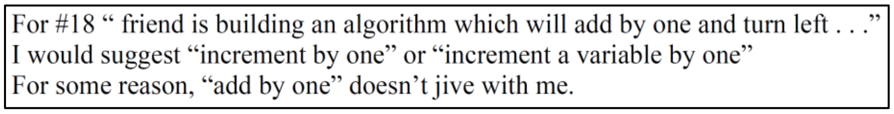

For #18 " friend is building an algorithm which will add by one and turn left . . ."
I would suggest "increment by one" or "increment a variable by one"
For some reason, "add by one" doesn't jive with me.

Figure 3.4. Example feedback from expert.

Programming Motivation Survey

The Programming Motivation Survey (Appendix E) was given before and after instruction. It was designed using a combination of intentionally and carefully selected statements from an existing valid and reliable instrument in addition to researcher-designed statements. The 25-item Likert type scale Programming Motivation Survey was adapted from the Science Motivation Questionnaire II (SMQ-II) created by Glynn et al. (2011). Reliability testing was conducted on the SMQ-II (Glynn et al., 2011) with 340 college student participants. The Cronbach's alpha of the SMQ-II (Glynn et al., 2011) is .92, which indicated a very good reliability score (DeVellis, 2003).

The Programming Motivation Survey had five subscales which are displayed in Table 3.9: (1) Intrinsic Motivation, (2) Career Motivation, (3), Self-Determination, (4) Self-Efficacy, and (5) Motivation to Integrate Programming into Teaching. The subscale of grade motivation from Glynn et al.'s (2011) instrument did not fit this study and was removed. In its place, a researcher-created subscale entitled "Motivation to Integrate Programming into Teaching" was added, which included five statements. In total, 15 of 20 statements from the SMQ-II's (Glynn, 2011) subscales of intrinsic motivation, career motivation, self-determination, and self-efficacy were adapted to focus on programming. The five statements from the subscales I adapted from the SMQ-II that did not fit the focus of the study were replaced with researcher-created statements (Appendix F). After the adaptations were made, the instrument was reviewed by three experts in the fields of programming and education.

Participants responded to items such as "Understanding programming will benefit me in my career" on a five-point Likert type scale from (1) strongly disagree, to (5) strongly agree. As advised by DeVellis (2003), the statements participants responded to were straight-forward in meaning and mixed in random order. Nine demographic information questions accompanied the Likert scale motivation items. These demographic questions gave context to the results and provided descriptive statistics on participants' age, gender, classification, concentrations within the education major, as well as previous experience with programming and robotics. Results were analyzed with either paired sample *t*-tests or Wilcoxon signed-ranks tests depending on their normality in order to compare the pre-survey and post-survey sets of data from the same participants (Mertler, 2017). The Cronbach's alpha for the Programming Motivation Survey in pre- (α = .963) and post- (α = .938) surveys indicated a very good reliability (DeVellis, 2003).

Table 3.9. *Programming Motivation Survey Subscale Alignment*

Statement	Subscale
3. Learning programming is interesting. 17. I am curious about advancing my programming skills. 1. Programming is relevant to my life. 12. Learning programming makes my life more meaningful. 19. I enjoy learning programming.	Intrinsic Motivation
7. Learning programming will help me get a good job. 13. Understanding programming will benefit me in my career. 10. Knowing programming will give me a career advantage. 25. I will use programming problem-solving skills in my career. 23. My career will involve programming.	Career Motivation
5. I put enough effort into learning programming. 11. I spend a lot of time learning programming. 6. I use various strategies to learn programming well. 20. I look for additional resources to improve my skills when learning programming. 16. I concentrate fully on what I do when I work on programming activities.	Self-Determination

Table 3.9. *Programming Motivation Survey Subscale Alignment* Continued.

Statement	Subscale
9. I am confident I will do well on programming tests.	Self-Efficacy
4. I am confident in learning programming.	
15. I believe I can master programming knowledge and skills.	
14. I am confident I will do well on programming activities.	
24. I can write advanced programs.	
22. I can teach programming in my future courses.	MTIPIT
21. I enjoy teaching programming to others.	
18. I plan to incorporate programming into my teaching.	
2. Teaching programming would benefit my students.	
8. Programming activities will enhance my students' learning.	

Field Notes

I maintained brief field notes during each class session. Field notes have been described as essential for rigorous qualitative research and offer an extra layer of detail with which to aid in the construction of thick, rich descriptions (Creswell, 2017; Phillippi & Lauderdale, 2018). When I was not teaching or providing scaffolding to participants, observations related to motivation and behavioral engagement (Fredricks et al., 2004; Kim et al., 2015, 2017; Skinner et al., 2009) were recorded. Examples of such observations included students voicing excitement and frustration programming the robots. Teamwork dynamics between partners were also recorded. For example, there were notes of when one participant within a team was noticeably more engaged with programming the robot than the other. Special notes were made for participants' absences, computer issues, and robot malfunctions. These notes were written in a composition book and coded in Delve and Microsoft Word.

Individual Interviews

Individual interviews were selected as a data collection method because they provided descriptive qualitative data of participants' perspectives on focused topics (Bloomberg & Volpe, 2016; Creswell & Poth, 2018; Mertler, 2017; Mills, 2018). Interviews, in this instance, gathered participants' reflections upon their programming experience throughout the study. This interview data provided further elaboration on participants' experiences, which may not appear in my field notes and quantitative survey data relative to the study's second research question (Creswell, 2014; Creswell & Poth, 2018).

Purposeful sampling was used to select participants for the interviews. One third of the participants (*n* = 6) were purposefully selected for individual interviews about their experiences within the intervention. Interviewees were selected based on my observations of participants' behavioral engagement (Fredricks et al., 2004; Kim et al., 2015, 2017; Skinner, Kindermann, & Fuller, 2009) that were recorded as field notes. Two participants representing high, medium, and low behavioral engagement were selected for individual interviews in order to have a balanced population of interviewees. High behavioral engagement was exhibited as on-task behavior, deep involvement, and active participation (Fredricks et al., 2004; Skinner et al., 2009; Stipek, 2002). For example, Paula demonstrated high engagement in all programming challenges and would actively contribute toward classroom activities and helping other groups. Medium behavioral engagement was intermittent, episodic on-task behavior and mild participation (Fredricks et al., 2004; Skinner et al., 2009; Stipek, 2002). For example, Randy demonstrated engagement, but with only some of the programming activities. He also demonstrated only mild participation with his partner. Low behavioral engagement was exhibited by participants who were routinely off task and made minimal contributions to their partner or the class through participation (Fredricks et al., 2004; Skinner et al., 2009; Stipek, 2002). For example, Jennifer was off-task and did not contribute towards the programming activities, as she let her partner do almost all the work.

I followed the interview protocol found in Appendix G. The interview questions were each aligned to the second research question, and as displayed in Table 3.10, 10 of the interview questions were aligned to the motivation subscales evaluated in the Programming Motivation Survey, while two were designed to directly gather data with which to improve the curriculum. In each interview, I prompted the participant with open-ended questions that guided the discussion. Open-ended questions were used by me to capture the rich detail of participants' attitudes and experiences (Creswell, 2017; Creswell & Poth, 2018; Morgan, 2018). After each question was presented to the participant, I listened to the participant's response. The individual interviews followed a semi-structured protocol (Merriam & Tisdell, 2016; Mertler, 2017). The semi-structured nature of the interviews allowed the flexibility to put forward additional probes when appropriate in order to elicit more detail (Creswell, 2017; Mertler, 2017). Each interview took approximately 30 minutes. The interviews were audio-recorded and transcribed in real-time using Microsoft Dictate in Microsoft Word. Then, I reviewed the resulting transcripts for accuracy and made edits as needed. While reviewing the transcriptions, observations were noted in the researcher journal which helped provide a context in the analysis and coding of the transcript.

Table 3.10. *Individual Interview Question Alignment*

Individual Interview Questions	Alignment
1. What aspects, if anything, interested you in the programming activities? Prompt: Can you explain what you found interesting about those programming activities?	Intrinsic Motivation
2. Tell me about your experiences with the programming activities in the course. Prompt: Which one(s) was(were) most enjoyable? Explain. Prompt: Which one(s) was(were) least enjoyable? Explain.	Intrinsic Motivation
3. How do you think learning programming will influence your career after graduation?	Career Motivation
4. In what ways do you believe learning programming would be valuable to you as a teacher? Prompt: How has your opinion changed since the beginning of this course?	Career Motivation
5. Can you tell me about a time when you felt learning programming was hard? Prompt: Why did you feel this way? Prompt: How did you overcome that situation?	Self-Determination
6. Tell me about a time you put in extra effort over the past four weeks to research additional resources to help you during the programming activities. Prompt: How did you make the decision to seek additional resources?	Self-Determination
7. Tell me about your current state of programming knowledge and skills? Prompt: How do you think it has changed since the beginning of this course?	Self-Efficacy
8. What are your feelings on learning even more advanced programming?	Self-Efficacy

Table 3.10. *Individual Interview Question Alignment* Continued.

Individual Interview Questions	Alignment
9. Where do you position yourself in the continuum of adding or not adding programming activities to your classes? Why?	MTIPIT
10. Tell me about your thoughts on how programming activities would fit into the grade level and subject area you will teach? Prompt: Describe an example programming activity for the grade or subject area you will be teaching.	MTIPIT
11. Which programming activities do you feel were effective in helping you learn programming? Prompt: What suggestions would you make to improve the programming activities in this course?	Perception of the Curriculum
12. Do you have any questions for me?	N.A.

Data Analysis

Quantitative and qualitative data were analyzed (Creswell, 2014; Merriam & Tisdell, 2016). Using both quantitative and qualitative data removed the biases of only utilizing one type of data in order to show a more accurate picture of the phenomenon being investigated (Creswell, 2014; Mertler, 2017). As demonstrated in Table 3.11, each research question was investigated with different sources of data and different analysis methods. First, the quantitative and then the qualitative data analysis processes are described in the following paragraphs.

Table 3.11. *Research Questions, Data Sources, and Data Analysis Method Alignment*

Research Questions	Data Sources	Data Analysis Method
RQ#1: What is the effect of educational robotics on preservice teachers' comprehension of programming concepts?	Programming Comprehension Assessment	Descriptive statistics Paired sample *t*-tests Wilcoxon signed-ranks tests

Table 3.11. *Research Questions, Data Sources, and Data Analysis Method Alignment* Continued.

Research Questions	Data Sources	Data Analysis Method
RQ#2: How and to what extent does educational robotics influence preservice teachers' motivation related to programming?	Programming Motivation Survey	Descriptive statistics Paired sample *t*-tests Wilcoxon signed-ranks tests
	Field notes	Inductive analysis
	Individual interviews	Inductive analysis

Quantitative Data Analysis

Student scores on the pre/post Programming Comprehension Assessment instrument were downloaded from Moodle as Microsoft Excel sheets and formatted for SPSS. Identification numbers were assigned to each participant. Participants who dropped out, non-education majors, and their associated data were removed prior to analysis. The data were uploaded into SPSS for data analysis. The students' scores on the Programming Comprehension Assessments were arranged into units for each of the four topics covered in instruction and compared using paired sample *t*-tests for the parametric data and a Wilcoxon signed-ranks tests for the non-parametric data. The paired sample *t*-tests and Wilcoxon signed-ranks tests were performed on the data in order to examine whether the intervention had an impact on students' Programming Comprehension Assessment scores. These data were depicted in tables, including the overall scores and unit categories along. The tables were accompanied by a text description.

Quantitative data from the pre/post Programming Motivation Survey instrument were downloaded from Moodle as Microsoft Excel sheets and formatted for SPSS. Identification numbers were assigned to each participant. Participants who dropped out, non-education majors, and their associated data were removed prior to analysis. The data were uploaded into SPSS for data analysis. Descriptive statistics were calculated at this time. Student responses to the Likert scale questions were analyzed within their pre-determined subscales. Results were analyzed with either paired sample *t*-tests or a Wilcoxon signed-ranks test depending on their normality in order to compare the pretest and posttest sets of data from the same participants (Mertler, 2017). As suggested by Mertler (2017), an alpha level of .05 was utilized to ascertain if the intervention had a significant impact on their programming comprehension scores. The Cronbach's alpha for this instrument's pretest ($\alpha = .96$) and posttest ($\alpha = .94$) indicated a very good reliability (DeVellis, 2003).

Qualitative Data Analysis

Inductive analysis was used to analyze the qualitative data (Creswell, 2017; Mertler, 2017). In this study, qualitative data came from the individual interviews and field notes. All transcripts and coding files were stored in a password-secured folder. The transcriptions and field notes were broken down through an inductive system of open coding in the first cycle, and pattern coding in the second cycle. Strauss and Corbin (1990) described open coding as "the process of breaking down, examining, comparing, conceptualizing, and categorizing data" (p. 61). Pattern coding is a second cycle coding method in which the researcher takes the first cycle codes and groups them into categories of similar codes (Saldaña, 2016). The pattern codes were then developed into larger categories (Saldaña, 2016). The data were analyzed for themes in the individual interviews and field notes (Braun & Clarke, 2006; Creswell, 2017; Mertler, 2017). These themes centered on representing students' perceptions about motivation related to programming and the educational robotics intervention. In this instance, the open coding led to pattern coding, which developed categories that were used to pinpoint themes that emerged during the data analysis (Bloomberg & Volpe, 2016; Creswell & Poth, 2018; Mills, 2018).

A coding web tool known as Delve and multiple Microsoft Word documents were used. As Creswell (2014) recommended, I recorded codes that were expected, surprising, or interesting related to the research question. Delve was used for the open coding of the data. Because Delve is limited in the movement and manipulation of open codes into pattern codes, the open codes were exported as a Microsoft Word document. Open codes were printed and sorted into pattern codes. Then, the open codes were moved to different Microsoft Word documents holding the different pattern codes that were generated. In new Microsoft Word documents, the pattern codes were aligned into umbrella categories. Then, themes were generated from these categories. The comments feature in Microsoft Word was used to keep notes on codes and the coding process. From this coding process, I reduced the qualitative data into a few of the most relevant categories depicting themes for sharing and further description (Creswell, 2014; Mertler, 2017).

The thematic findings are represented in two different ways. First, a table depicting the different themes uncovered by the interviews is displayed. Second, thick, rich description with selected quotes from the individual interviews and field notes were used to weave together the description of the participants' experiences relative to programming motivation. Interpretations of participants' perspectives were presented to provide context. Further conversation comparing the results of the data analysis relative to research question two, followed in a discussion section.

Procedures and Timeline

The timeline for the procedures for this research included three phases: (1) Introduction, (2) Robotics Intervention, and (3) Data Collection and Analysis. As demonstrated in Table 3.12, the three phases of the study take place over a total of 16 weeks. Each phase is described in the paragraphs below.

Table 3.12. *Timeline and Procedures*

	Phase 1: Introduction (1 week)
Week 1	Getting Started
Class 1	1. Explanation of study 2. Informed consent 3. Pre-Programming Comprehension Assessment and pre-Programming Motivation Survey
Class 2	1. Computer setup 2. Robot setup 3. Robot operation overview & troubleshooting
	Phase 2: Robotics Intervention (4 weeks)
Week 2	Basic Procedures
Class 1	1. Basic Procedures programming demonstration 2. Free time to experiment with programming robots
Class 2	1. Odometry Activity 2. One Meter Challenge
Week 3	Advanced Procedures
Class 1	1. Pseudocoding lap demonstration 2. Lap Activity
Class 2	1. Pseudocoding maze demonstration 2. Maze Challenge
Week 4	Control Structures
Class 1	1. Looping demonstration 2. Slithering One Meter Activity
Class 2	1. Flow control overview 2. Lap Loop Challenge
Week 5	Variables
Class 1	1. Color sensor demonstration 2. Red Light Activity
Class 2	1. Variables overview 2. Maze with Variables Challenge
	Phase 3: Data Collection & Analysis (11 weeks)
Week 6	Data Gathering

Table 3.12. *Timeline and Procedures* Continued.

Class 1 Class 2	1. Post-Programming Comprehension Assessment and post-Programming Motivation Survey 2. Individual interviews
Week 7 & 8	Interview Transcripts - Initial Analysis
Independent	1. Review interview audio with transcripts for accuracy 2. Review transcripts' contents 3. Member checking of transcripts
Week 9 & 10	Comprehension Assessment Analysis
Independent	1. Prepare data for SPSS 2. Paired sample *t*-tests and Wilcoxon signed-ranks tests on comprehension data 3. Reliability analysis
Week 11 & 12	Motivation Survey Analysis
Independent	1. Prepare data for SPSS 2. Paired sample *t*-tests and Wilcoxon signed-ranks tests on motivation data 3. Reliability analysis
Week 13 – 16	Coding of Qualitative Data
Independent	1. Rounds of coding and peer debriefing 2. Categories and themes

Phase 1: Introduction

There were actions that were completed before the study began. Institutional Review Board approval was gained from both my associated university (Appendix H) and the research site (Appendix I). Before the study began, steps were completed to prepare the participants for the study. The week before the Robotics Intervention phase began, there were two class periods dedicated to preparing the participants for the robotics lessons. The events of these two class periods will be described in this section.

First, students were briefed on the study. In this briefing, the purpose of the study, procedures of the study, duration of the study, rights of participants, risks to participants, benefits to participants, confidentiality, and sharing of results were explained to students. Students were given time to ask questions and reflect upon their decision to participate or not. It was explained to students that participating or not participating in this study would not influence their grades, and that participation was optional. At the conclusion of this introduction, informed consent was obtained from the participants. Students were given the research site university's consent form and photography, video, and audio recording release form (Appendix J). Participants signed these two forms, and the forms were collected by me and stored in a secure location. The pretest Programming Comprehension Assessment and the pre-instructional Programming Motivation Survey were given to participants. To access these instruments, participants logged into the course webpage in Moodle and navigated down to the associated week. There participants found the Programming Comprehension Assessment pretest and pre-instructional Programming Motivation Survey. Participants completed the pretest first, followed immediately by the pre-instructional survey.

During the next class period, participants were familiarized with the robots and programming software. Participants were paired and given a robot and laptop. Then, the pairs followed the instructor through the process of how participants were to connect the laptop to the robot using Bluetooth. Then, the instructor described the different parts of the robots. The instructor showed students the different motors and sensors of the robots in a presentation. The functions of each type of motor and sensor were explained. Instructional time was dedicated to showing the participants how to freeze the robot in situations where the robot goes awry. Then, participants were shown how to troubleshoot problems that may occur with the robots. To finish this class period, the instructor showed participants the different sections of the programming software. Participants followed along with the instructor on their laptops. The instructor demonstrated the programming canvas, the programming palettes, and the hardware tab. This basic overview concluded the first week's activities.

Phase 2: Robotics Intervention

Phase 2 was divided into four week-long units. Each week/unit contained two class periods of 1.25 hours for a total of 2.50 hours of instructional time per week/unit. The basic structure of each unit was the same. Each unit began with an instructor-led overview of the concepts in the unit, including context and curricular integration ideas, and taught participants robotics programming concepts. Participants then practiced the new programming concepts through learning activities. Finally, participants completed programming challenges to finish each unit.

Participants began with the Basic Procedures unit. The first class of this unit consisted of a basic overview of programming and a programming demonstration. After that, participants had free time to experiment with programming the robots. In the second class of the unit, participants took part in an odometry learning activity and then the One Meter Challenge. Next, participants moved on to the Advanced Procedures unit. The first class of this unit consisted of a pseudocoding demonstration and the Lap Activity. The second class of this unit began with a pseudocoding activity for following a path and ended with the Maze Challenge. Then, participants took part in the Control Structures unit. The first class of this unit started with a looping demonstration and ended with the Slithering One Meter Activity. The second class of this unit started with a control structures overview which explained loops and if/then statements and ended with the Lap Loop Challenge. Finally, participants completed the Variables unit. The first class of this unit began with a color sensor demonstration and ended with the Red Light Activity. The second class of this unit began with an overview of variables and ended in the Maze with Variables Challenge.

Phase 3: Data Collection and Analysis

In the first part of this phase, participants took the same Programming Comprehension Assessment and then the Programming Motivation Survey that they had taken previously. The instruments were again available in Moodle, the participants' course management system. The next part of this phase required me to obtain qualitative data through individual interviews. The audio from each of these recordings was transcribed with Microsoft Dictate and loaded into Microsoft Word. The cleaned transcripts were shared with the interviewees for member checking. The transcripts were then reviewed by me in order to become familiarized with the content. The transcripts along with the field notes were then uploaded into Delve for coding and inductive analysis. Descriptive statistics, paired sample *t*-tests, and Wilcoxon signed-ranks tests were then performed on the pre/post results of each instrument. The transcripts were coded, and themes were gathered from the data. Finally, participants had the opportunity to critique the findings of this study.

Rigor and Trustworthiness

Researchers must communicate the actions they have taken to assert the rigor and trustworthiness of their findings (Creswell, 2014). There were five strategies employed by me to ensure the rigor and trustworthiness of the qualitative data in this study. The strategies that were used to confirm rigor and trustworthiness in this study are (1) triangulation, (2) member checking, (3) peer debriefing, (4) audit trail, and (5) thick, rich description. These strategies are detailed in the following paragraphs.

Triangulation

Methodological triangulation is the most evident strategy used to ensure rigor and trustworthiness for this study. Methodological triangulation united data from mixed sources and methods (Buss & Zambo, 2014; Creswell, 2014; Patton, 2002). The mixing of quantitative survey data on motivation, qualitative field notes, and qualitative individual interview response data about motivation constituted the mixed methods for research question #2. These mixed sources created a dialogue between the different perspectives offered through the disparate ways of investigating the phenomenon (Maxwell, 2010). After all data were collected and the data sources were analyzed individually, these results were then compared to corroborate findings from each different methodology, ensuring consistency (Bazeley, 2013; Creswell, 2014).

Member Checking

As this study was conducted through the scope of action research, a collaborative member checking process was used. Multiple rounds of member checking were used in this study. Member checking ensures trustworthiness by allowing stakeholders to verify the accuracy of the findings (Creswell, 2014; Merriam, 1998; Mertler, 2017). The first member checking occurred after the individual interviews. Participants were presented with the transcripts of their individual interviews through email. Each email was kept on a separate email chain for each participant as to preserve anonymity. I inquired if the transcripts were reflective of what the participants meant during the individual interviews. Participants had the opportunity to critique or correct me during this time to further establish the trustworthiness of the results (McMillan, 2016; Mills, 2018). No inaccuracies were reported by participants, and three of the six interviewees confirmed their transcripts' accuracy, while the other three did not respond. The themes and categories created were then shared with the participants after the data were coded and analyzed. Again, participants were asked to critique or correct the themes and categories. The accuracy of the themes and categories was confirmed by three of the six interviewees, but no additional insights were provided. The other three interviewees did not respond.

Peer Debriefing

Peer debriefing has been described by Lincoln and Guba (1985) as a discussion with the goal of examining the methodological process, exploring unrealized possibilities in the study, as well as checking and defending the study's findings or interpretations. According to Shenton (2004), peer debriefing with other academics offers "the fresh perspective that such individuals may be able to bring" which can "challenge assumptions made by the investigator" who may become too close to the subject matter to see opportunities for the study's refinement (p. 67). Creswell (2014) echoed this notion by explaining that including the perspectives of other academics to review a study acts as an external evaluation on the rigor and trustworthiness of the methods and interpretations of results. Peer debriefing with my dissertation chair was used to ensure all methods were fundamentally sound, and all interpretations were justified and accurate. Throughout the study, instruments, data, codes, themes, and interpretations were constantly shared and reviewed with the dissertation chair. From peer debriefing insights, the study was refined, and the accuracy of results were improved, adding credibility to the results (Bloomberg & Volpe, 2016; McMillan, 2016).

Audit Trail

The audit trail in this study consisted of a researcher journal that documented both reflections from the intervention as well as decisions that were made during the data analysis process (Creswell, 2017). An audit trail was used to document an ongoing record of events and decisions which occurred during the study and analysis (Lincoln & Guba, 1985; Merriam, 1998). The researcher journal provided a linear timeline of thoughts and events germane to the intervention and data analysis aspects of the study. The insights within my field notes on each class session were incorporated into the researcher journal and elaborated upon. In addition, reflections on the lessons and summarizations of experiences were written immediately following each class session. These passages were used to provide context when reporting the results of the study. Further, I used the researcher journal to remember what was previously done and what needed to be done while working through the data analysis phase. For example, I made notes about which codes were used and why during the thematic analysis of the interview data. In this way, I ensured that there was a written record that supported the thought process behind each code. In turn, these thought processes and decisions could be shared in the dissertation. The researcher journal was an ongoing document written in Microsoft Word.

Thick, Rich Description

Thick, rich descriptions were detailed, illustrative accounts that enabled the reader to better understand the study (Bazeley, 2013; Creswell, 2014; Mertler, 2017). These detailed descriptions allowed the reader to make analyses and begin to draw conclusions (Lincoln & Guba, 1985; Merriam, 1998). In this study, interview responses as well as field notes explaining the phenomena were described and interpreted. The perspectives of the participants were woven into the thick, rich description to add authenticity and support to my inferences. These thick, rich descriptions of the study provided context to the reader.

Plan for Sharing

The results of this study were shared with various audiences. In each of the methods of sharing, copies of the quantitative and qualitative data that were included in each presentation or report were devoid of any identifying characteristics. Information was reported in aggregate, and student pseudonyms in the form of study-specific identification numbers were used for specific examples in order to “limit descriptions of individuals” to the point that “they are not easily identifiable” (Mertler, 2017, p. 271). The results of this study were presented to (a) the study’s participants, (b) the university’s Instructional Technology department, (c) readers of peer-reviewed journals, and (d) attendees of international and national professional conferences. The methods for sharing findings with each of these audiences will be described in the paragraphs below.

Participants

The results of the study were shared with participants through a visual presentation given by me. This presentation occurred after the member checking of the themes of the study. Participants were given the opportunity to comment on the findings in accordance with the action research model wherein participants are collaborators (Creswell, 2014; Mertler, 2017). All questions were answered, and reflections were made during this time.

The University's Instructional Technology Department

The professional stakeholders that are part of the Instructional Technology department all met in the Instructional Technology meeting area. I provided a visual presentation as well as a hard-copy report of the study to the department. This report included an outline of the study's instructional modules, data, findings, as well as a list of suggested updates and improvements. All professional stakeholders collaborated on brainstorming additional updates and improvements to the programming instruction and documented action steps for updating the instructional modules.

Readers of Peer-Reviewed Journals

Articles related to this study's research questions will be written. These articles will be derivative of the dissertation's contents. I will segment the dissertation into different pieces to report the results. Potential journals will be selected based on the advisement of my chair. Although action research is not widely generalizable, the findings of this dissertation will help add to the scarce literature available to academics and practitioners related to preservice teachers learning programming through robotics.

Attendees of Professional Conferences

The results of this study will also be shared at educational technology conferences. A presentation of selected findings is planned for an international conference within the year of the dissertation's successful defense. Other international and national presentations will be planned upon successful defense at the recommendation of my chair.

CHAPTER 4

ANALYSIS AND FINDINGS

The purpose of this action research was to evaluate the effect of educational robotics on the programming comprehension and motivation of preservice teachers at a medium-sized liberal arts university in the southeastern United States. The findings from this study will aid in understanding the impact of educational robotics on preservice teachers’ comprehension and motivation related to programming before they begin their professional practice. The data collection in this study was aligned to two research questions:

1. What is the effect of educational robotics on preservice teachers’ comprehension of programming concepts?
2. How and to what extent does educational robotics influence preservice teachers’ motivation related to programming?

This chapter provides evidence of comprehension and motivation that were gathered from participants during data collection. Of the eligible 21 education majors taking the course, three participants dropped out of the class during the study. These participants’ data were removed prior to analysis. This analysis only included the pre/post Programming Comprehension Assessment and Programming Motivation Survey data from the remaining 18 participants.

This chapter is divided into two sections representing the mixed methods used in this study. The quantitative section reports the pre/post results of the Programming

Comprehension Assessment and the Programming Motivation Survey as well as subsequent data analysis on the participants' responses. The qualitative section documents the findings based on the analysis of the individual interviews. The quantitative results will be reported first in this chapter, followed by the qualitative results. At the end, those two sources of findings will be integrated.

Quantitative Analysis and Findings

This section provides the quantitative results from the instruments utilized in this study. Data were collected before and after the educational robotics intervention using two instruments: (1) Programming Comprehension Assessment and (2) the Programming Motivation Survey. The data presented in this section include participants' overall pre and post results as well as the data for each respective unit or subscale within each instrument. First, the pre/post Programming Comprehension Assessment results will be presented, followed by the pre/post Programming Motivation Survey results.

Programming Comprehension Assessment

The Programming Comprehension Assessment was given to participants before and after the educational robotics programming intervention. The instrument was evaluated by two experts in block-based programming and educational robotics to establish face validity (Salkind, 2010). The Programming Comprehension Assessment included 20 multiple choice questions grouped into four units of five questions representing each of the instructional units in the intervention (Basic Procedures, Advanced Procedures, Control Structures, and Variables). Each multiple-choice question had five answer choices. There was only one correct answer per question. Each question had one point possible for a total possible score of 20 points. Each unit had a total possible score of five points.

Descriptive statistics. First, the data were analyzed using descriptive statistics, and an item difficulty analysis was run based on the participants' average scores on the Programming Comprehension Assessment. From the pretest (*M* = .21, *SD* = .07) to the posttest (*M* = .58, *SD* = .24), participants' overall programming comprehension improved.

An item difficulty analysis, shown in Table 4.1, displays the difficulty of each question on the Programming Comprehension Assessment. Difficulty index values were calculated. Item difficulty levels in this study were equal to the percentage of participants who answered the items correctly, or the items' mean scores (Lord, 1952). Difficulty index values varied between .22 and .83, resulting in a mean difficulty index calculation of *M* = .58. According to Lord (1952), the difficulty for a five-response option multiple choice question with one correct answer choice is ideally .70. Hopkins and Antes (1990) noted that difficulty levels below .14 were very difficult, levels between .15 and .29 were difficult, levels between .30 to .70 were moderate, levels between .71 to .85 were easy, and levels from .86 and above were very easy. According to Hopkins and Antes's (1990) difficulty levels, the Programming Comprehension Assessment included two difficult questions, 12 moderate questions, and six easy questions, with an overall moderate difficulty level (*M* = .58, *SD* = .24). In Table 4.1, the item difficulty and unit difficulty levels are equal to the means outlined.

Table 4.1. *Item Difficulty – Programming Comprehension Assessment Posttest*

Units	Question	*M*	*SD*
Basic Procedures	Q1	.83	.38
	Q2	.44	.51
	Q3	.83	.38
	Q4	.28	.46
	Q5	.56	.51
	Basic Procedures Total	.59	.49
Advanced Procedures	Q6	.56	.51
	Q7	.78	.43
	Q8	.56	.51
	Q9	.67	.49
	Q10	.72	.46
	Advanced Procedures Total	.66	.48
Control Structures	Q11	.39	.50
	Q12	.72	.46
	Q13	.50	.51
	Q14	.44	.51
	Q15	.83	.38
	Control Structures Total	.58	.57
Variables	Q16	.61	.50
	Q17	.61	.50
	Q18	.44	.51
	Q19	.67	.49
	Q20	.22	.43
	Variables Total	.51	.49
Total Programming Comprehension Assessment Difficulty		.58	.24

Note. Mean is equal to item difficulty.

Participants' scores in each of the units in the assessment representing the four different instructional units (Basic Procedures, Advanced Procedures, Control Structures, and Variables), as well as the total scores, were analyzed for the pretest and posttest. First, the Shapiro-Wilk test was used to evaluate the normality of the data. Based on those results, a paired sample *t*-test or a Wilcoxon signed-ranks test was used to analyze the data, respectively.

Shapiro-Wilk normality tests. The Shapiro-Wilk test was used to determine if the data were normally distributed for each unit as well as the total scores. To complete the Shapiro-Wilk tests, the participants' pre and post average scores for each unit and overall total were calculated. Then, the differences between the pretest and posttest for each unit, as well as the overall total for the pretest and posttest, were calculated to create a new variable that represented the difference in scores between the pre and posttest. These differences were then analyzed using Shapiro-Wilk tests.

Shapiro-Wilk test results with *p* values above .05 indicated that the data are normally distributed, while *p* values under .05 indicated that the data are not normally distributed (Gibbons & Chakraborti, 2011). The data (see Table 4.2) were found to be normally distributed for the units of Control Structures (p = .212) and Variables (p = .534), as well as the Total (p = .143) using the Shapiro-Wilk tests ($p > .05$). However, the units of Basic Procedures (p = .017) and Advanced Procedures (p = .042) were found to violate the normality assumption.

Table 4.2. *Shapiro-Wilk Normality Tests – Programming Comprehension Assessment*

Units	*W*	*df*	*p*
Basic Procedures Difference	.87	18	.017*
Advanced Procedures Difference	.89	18	.042*
Control Structures Difference	.93	18	.212
Variables Difference	.96	18	.534
Total Programming Comprehension Assessment Difference	.92	18	.143

Note. * Indicates not normally distributed data ($p < .05$)

The next steps of the data analysis process were guided by the Shapiro-Wilk test results. Either the paired sample *t*-test or Wilcoxon signed-ranks test were used to analyze the data depending on their normality results from the Shapiro-Wilk test, as outlined in Table 4.3. The data for the units and total that were normally distributed were analyzed using paired sample *t*-tests and the data for the units that were not normally distributed were analyzed using Wilcoxon signed-ranks tests (Gibbons & Chakraborti; Pappas & DuPuy, 2004). Cohen's *d* was calculated to determine the effect size for the change in each unit for the parametric data (Cohen, 1988). The effect size of the change in the non-parametric test was reflected by the correlation coefficient *r* (Pallant, 2007; Rosenthal, 1994). To minimize familywise Type 1 error inflation, the Bonferroni correction (Bland & Altman, 1995) level was calculated for the total number of tests conducted on the instrument (5).

Table 4.3. *Data Analysis Method Alignment Based on Normality of Data – Programming Comprehension Assessment*

Shapiro-Wilk Test Results	Units	Data Analysis Method
Normally Distributed	Control Structures Variables Total Programming Comprehension	Paired sample *t*-test
Not Normally Distributed	Basic Procedures Advanced Procedures	Wilcoxon signed-ranks test

Paired sample *t*-tests. Paired sample *t*-tests were conducted to compare participants' scores on the Control Structures, Variables, and Total between pretest and posttest. The paired sample *t*-tests revealed that participants' posttest scores were significantly higher than pretest scores. Participants' comprehension of programming concepts increased from the pretest (*M* = .21, *SD* = .07) to the posttest (*M* = .58, *SD* = .24), *t*(17) = 6.48, *p* < .001, Cohen's *d* = 1.53. Participants' comprehension of control structures increased from the pretest (*M* = .26, *SD* = .17) to the posttest (*M* = .58, *SD* = .26), *t*(17) = 4.68, *p* < .001, Cohen's *d* = 1.10. Participants' comprehension of variables increased from the pretest (*M* = .19, *SD* = .17) to the posttest (*M* = .51, *SD* = .32), *t*(17) = 3.69, *p* < .001, Cohen's *d* = .87.

As demonstrated in Table 4.4, the overall increase in students' total scores on the assessment from pretest to posttest was found to be statistically significant. The units of Control Structures and Variables also demonstrated significant increases from pretest to posttest. As demonstrated in Table 4.4, the effect size for this analysis was found to exceed Cohen's (1988) convention for a large effect (*d* = .80) for these units in addition to the total. To minimize familywise Type 1 error inflation, the Bonferroni correction (Bland & Altman, 1995) level was calculated by dividing the desired alpha level of .05 by total number of comparisons (5) to reveal a new significance threshold of *p* < .01. Both subscales and the total remained significant at the Bonferroni correction alpha level. Specifically, the results suggest that when preservice teachers learn programming through educational robotics, their comprehension of control structures, variables, and programming in general can be increased.

Table 4.4. *Paired Sample t-Tests – Programming Comprehension Assessment Averages*

	Pretest		Posttest					
Units	*M*	*SD*	*M*	*SD*	*t*	*df*	*p*	*d*
Control Structures	.26	.17	.58	.26	4.68	17	< .001*†	1.10
Variables	.19	.17	.51	.32	3.69	17	.002*†	0.87
Total Programming Comprehension	.21	.07	.58	.24	6.48	17	< .001*†	1.53

Note. Units were out of five questions. The total was out of 20.
* Indicates the differences between pretest and posttest is significant *p* < .05.
† Indicates the differences between pretest and posttest is significant at Bonferroni correction level *p* < .01.

Wilcoxon signed-ranks tests. The data that were not normally distributed for the units of Basic Procedures and Advanced Procedures were analyzed using Wilcoxon signed-ranks tests. The Wilcoxon signed-ranks tests were used to produce valid non-parametric results because it is a superior analysis method for data that are non-normal in distribution (Pappas & DePuy, 2004). To complete the Wilcoxon signed-ranks tests, students' average scores for each unit in addition to their average overall scores were calculated for the pretest and posttest. The average scores for each unit as well as the average total scores were then compared using Wilcoxon signed-ranks tests. According to Pallant (2007) and Rosenthal (1994), the effect size for Wilcoxon signed-ranks tests can be calculated by dividing the *Z* value by the root of the total *N* observations resulting in the correlation coefficient *r*. The resulting statistics are displayed in Table 4.5. The medians of Basic Procedures pretest and posttest were .20 and .70, respectively. A Wilcoxon signed-ranks test indicated that that there was a statistically significant effect in Basic Procedures ($Z = -3.30$, $p = .001$, $r = -.55$). The medians of the pretest Advanced Procedures and posttest Advanced Procedures were .20 and .70, respectively. A Wilcoxon signed-ranks test indicated that that there was a statistically significant effect in Advanced Procedures ($Z = -3.43$, $p = .001$, $r = -.57$). The effect size below -.50 indicated a large effect size (Cohen, 1992). To minimize familywise Type 1 error inflation, the Bonferroni correction (Bland & Altman, 1995) level was calculated by dividing the desired alpha level of .05 by total number of comparisons (5) to reveal a new significance threshold of $p < .01$. Both subscales remained significant at the Bonferroni correction alpha level. Specifically, the results suggest that when preservice teachers learn programming through educational robotics, their comprehension of basic and advanced concepts can be increased.

Table 4.5. *Wilcoxon Signed-Ranks Tests – Programming Comprehension Assessment Averages*

	Pretest		Posttest				
Units	*Mdn.*	*SD*	*Mdn.*	*SD*	*Z*	*p*	*r*
Basic Procedures	.20	.15	.70	.29	-3.30	.001*†	-.55
Advanced Procedures	.20	.18	.70	.30	-3.43	.001*†	-.57

Note. Units were out of five questions.
* Indicates the differences between pretest and posttest is significant $p < .05$.
† Indicates the differences between pretest and posttest is significant at Bonferroni correction level $p < .01$.

Programming Motivation Survey

The Programming Motivation Survey was given to participants before and after the robotics programming intervention. The Programming Motivation Survey included 25 five-point Likert scale questions grouped into five subscales of five questions representing each subscale examined in this study (Intrinsic Motivation, Career Motivation, Self-Determination, Self-Efficacy, MTIPIT). Each Likert scale question asked participants to indicate their level of agreement with a statement from 1 (strongly disagree) to 5 (strongly agree). Both the pretest and posttest Programming Motivation Survey were tested for reliability (N = 18). According to DeVellis (2003), a Cronbach's alpha coefficient below .60 is unacceptable, .60 to .69 is undesirable, .70 to .80 is respectable, and .80 and above is very good. The Cronbach's alpha for this instrument's pretest (α = .96) and posttest (α = .94) indicated very good reliability (DeVellis, 2003). The reliabilities of each of the instrument's subscales were also tested, as shown in Table

4.6. The range of Cronbach's alpha levels for these ranged from .80 to .90, which also indicated very good reliability (DeVellis, 2003).

Table 4.6. *Cronbach's Alpha Reliability – Programming Motivation Survey*

Subscales	Pretest α	Posttest α
Intrinsic Motivation	.89	.89
Career Motivation	.88	.80
Self-Determination	.90	.89
Self-Efficacy	.85	.87
MTIPIT	.89	.81
Total Programming Motivation	.96	.94

Descriptive statistics. First, descriptive statistics about the programming motivation survey were presented in Table 4.7. From the pre-survey (*M* = 2.38, *SD* = .84) to the post-survey (*M* = 3.48, *SD* = .64), participants' overall mean motivation improved. The subscale with the largest increase was Self-Determination in which participants' mean motivation improved 28% between pretest and posttest.

Table 4.7. *Descriptive Statistics – Programming Motivation Survey*

Subscales		*M*	*SD*
Intrinsic Motivation	Pre-survey	2.23	0.93
	Post-survey	3.11	0.96
	Difference	1.12	0.03
Career Motivation	Pre-survey	2.94	0.98
	Post-survey	3.72	0.59
	Difference	.78	0.39
Self-Determination	Pre-survey	1.99	0.98
	Post-survey	3.39	0.72
	Difference	1.41	0.26
Self-Efficacy	Pre-survey	2.17	0.82
	Post-survey	3.47	0.84
	Difference	1.30	0.03

Table 4.7. *Descriptive Statistics – Programming Motivation Survey* Continued.

Subscales		*M*	*SD*
MTIPIT	Pre-survey	2.59	1.04
	Post-survey	3.72	0.75
	Difference	1.13	0.29
Total Programming Motivation	Pre-survey	2.38	0.84
	Post-survey	3.48	0.64
	Difference	1.10	0.20

Note. Out of five-point Likert scale.

Students' responses in each of the subscales in the survey (Intrinsic Motivation, Career Motivation, Self-Determination, Self-Efficacy, MTIPIT), as well as the totals, were analyzed from the pre-survey and post-survey. The Programming Motivation Survey data were analyzed for normality, and then one of two tests was used to evaluate if the intervention's results indicated an increase in motivation related to programming. In the same process outlined earlier in this chapter, first, Shapiro-Wilk tests were used to evaluate the normality of the data. From there, either the paired sample *t*-test or Wilcoxon signed-ranks test was used depending on the results of the Shapiro-Wilk tests.

Shapiro-Wilk normality tests. The Shapiro-Wilk test was used to determine if the data were normally distributed for each subscale as well as the total. To complete the Shapiro-Wilk tests, the participants' pre-survey and post-survey Likert scale averages for each subscale as well as the total were calculated. Then, the differences between the Likert scale averages for each subscale as well as the total from the pre-survey and post-survey were found to create a new variable that represented the difference in Likert scale averages between the pre-survey and post-survey. These differences, shown in Table 4.8, were then analyzed using Shapiro-Wilk tests.

The data were found to be normally distributed for the Total (*p* = .796) as well as the subscales of Intrinsic Motivation (*p* = .353), Self-Determination (*p* = .155), Self-Efficacy (*p* = .814), and MTIPIT (*p* = .974) using Shapiro-Wilk tests (*p* > .05). However, the subscale of Career Motivation (*p* = .045) was found to not be normally distributed. Therefore, as demonstrated in Table 4.8, the null hypothesis was retained for Intrinsic Motivation, Self-Determination, Self-Efficacy, MTIPIT, and Total while the null hypothesis was rejected for Career Motivation.

Table 4.8. *Shapiro-Wilk Normality Tests – Programming Motivation Survey*

Subscales	*W*	*df*	*p*
Intrinsic Motivation Difference	.95	18	.353
Career Motivation Difference	.89	18	.045*
Self-Determination Difference	.92	18	.155
Self-Efficacy Difference	.97	18	.814
MTIPIT Difference	.98	18	.974
Total Programming Motivation Difference	.97	18	.796

Note. * Indicates not normally distributed data (*p* < .05).

The next steps of the data analysis process were guided by the Shapiro-Wilk test results. Either the paired sample *t*-test or Wilcoxon signed-ranks test were used to analyze the data depending on their normality results from the Shapiro-Wilk test, as outlined in Table 4.9. The data for the subscales and total that were normally distributed were analyzed using the paired sample *t*-test, and the data for the subscales that were not normally distributed were analyzed using the Wilcoxon signed-ranks test (Gibbons & Chakraborti; Pappas & DuPuy, 2004). Cohen's *d* was calculated to determine the effect size for the change in each unit for the parametric data (Cohen, 1988). The effect size of the change in the non-parametric test was reflected by the correlation coefficient *r*

(Pallant, 2007; Rosenthal, 1994). To minimize familywise Type 1 error inflation, the Bonferroni correction (Bland & Altman, 1995) level was calculated by dividing the desired alpha level of .05 by the total number of comparisons (6) on the instrument's data to reveal a new significance of *p* < .008.

Table 4.9. *Data Analysis Method Alignment Based on Normality of Data – Programming Motivation Survey*

Shapiro-Wilk Test Results	Subscales	Data Analysis Method
Normally Distributed	Intrinsic Motivation Self-Determination Self-Efficacy MTIPIT Total Programming Motivation	Paired sample *t*-test
Not Normally Distributed	Career Motivation	Wilcoxon signed-ranks test

Paired sample *t*-tests. Paired sample *t*-tests were conducted to compare participants' survey responses on the pre-survey and post-survey for the normally distributed subscales of Intrinsic Motivation, Self-Determination, Self-Efficacy, MTIPIT, and the normally distributed Total. To complete the paired sample *t*-tests, participants' average Likert scale agreement levels for each subscale as well as their total results, were calculated on the pre-survey and post-survey. The changes in each of the subscales as well as the overall total were then compared using the paired sample *t*-tests.

The paired samples *t*-tests revealed that participants' posttest scores were significantly higher than pretest scores. Participants' overall programming motivation increased from the pre-survey (*M* = 2.38, *SD* = 0.84) to the post-survey (*M* = 3.48, *SD* = 0.64), *t*(17) = 6.10, *p* < .001, Cohen's *d* = 1.44. Participants' intrinsic motivation increased from the pre-survey (*M* = 2.23, *SD* = 0.93) to the post-survey (*M* = 3.11, *SD* = 0.96), *t*(17) = 4.26, *p* = .001, Cohen's *d* = 1.00. Participants' self-determination increased from the pre-survey (*M* = 1.99, *SD* = 0.98) to the post-survey (*M* = 3.39, *SD* = 0.72), *t*(17) = 7.07, *p* < .001, Cohen's *d* = 1.67. Participants' self-efficacy increased from the pre-survey (*M* = 2.17, *SD* = 0.82) to the post-survey (*M* = 3.47, *SD* = 0.84), *t*(17) = 5.75, *p* < .001, Cohen's *d* = 1.36. Participants' MTIPIT increased from the pre-survey (*M* = 2.59, *SD* = 1.04) to the post-survey (*M* = 3.72, *SD* = 0.75), *t*(17) = 6.10, *p* < .001, Cohen's *d* = 1.20.

As demonstrated in Table 4.10, the overall increase in students' total programming motivation on the survey from pre-survey to post-survey was found to be statistically significant with the paired sample *t*-test *t*(17) = 6.10, *p* < .05. Intrinsic Motivation *t*(17) = 4.26, *p* = .001, Self-Determination *t*(17) = 7.07, *p* < .001, Self-Efficacy *t*(17) = 5.75, *p* < .001, and MTIPIT *t*(17) = 5.09, *p* < .001 also demonstrated significant increases from pre to post. These results suggest that educational robotics did have an impact on preservice teachers' programming motivation. Specifically, the results suggest that when preservice teachers learn programming through educational robotics, their programming motivation can be increased. As demonstrated in Table 4.10, the effect size for this analysis was found to exceed Cohen's (1988) convention for a large effect (*d* = .80) for these subscales in addition to the total. All the subscales and the total were found to be significant at the Bonferroni correction level of *p* < .008.

Table 4.10. *Paired Sample t-Tests – Programming Motivation Survey Likert Scale Agreement*

Subscales	Pretest *M*	Pretest *SD*	Posttest *M*	Posttest *SD*	*t*	*df*	*p*	*d*
Intrinsic Motivation	2.23	0.93	3.11	0.96	4.26	17	.001*†	1.00
Self-Determination	1.99	0.98	3.39	0.72	7.07	17	< .001*†	1.67
Self-Efficacy	2.17	0.82	3.47	0.84	5.75	17	< .001*†	1.36
MTIPIT	2.59	1.04	3.72	0.75	5.09	17	< .001*†	1.20
Total Programming Motivation	2.38	0.84	3.48	0.64	6.10	17	< .001*†	1.44

Note. Out of five-point Likert scale.
* Indicates the differences between pre-survey and post-survey is significant $p < .05$.
† Indicates the differences between pre-survey and post-survey is significant at Bonferroni correction level $p < .008$.

Wilcoxon signed-ranks test. The data that were not distributed normally for the subscale of Career Motivation were analyzed using the Wilcoxon signed-ranks test. To complete the Wilcoxon signed-ranks test, students' average Likert scale agreement levels for the Career Motivation subscale was calculated for the pre-survey and post-survey. The motivation levels were then compared using the Wilcoxon signed-ranks test. The correlation coefficient *r* was calculated to represent the effect size (Pallant, 2007; Rosenthal, 1994). The resulting statistics are displayed in Table 4.11. The medians of pre/post Career Motivation were 3 and 3.72, respectively. A Wilcoxon signed-ranks test indicated that there was a statistically significant effect in Career Motivation ($Z = -3.58$, $p < .001$, $r = -.6$). The effect size below -.50 indicated a large effect size (Cohen, 1992). The subscale was found to be significant at the Bonferroni correction level of $p < .008$. These results suggest that educational robotics positively impact preservice teachers' Career Motivation related to programming. Specifically, the results suggest that when preservice teachers learn programming through educational robotics, their Career Motivation related to programming can be increased.

Table 4.11. *Wilcoxon Signed-Ranks Test – Programming Motivation Survey*

	Pre-survey		Post-survey				
Subscales	*Mdn.*	*SD*	*Mdn.*	*SD*	*Z*	*p*	*r*
Career Motivation	3	0.98	3.72	0.59	-3.58	< .001*†	-.6

Note. Out of five-point Likert scale.
* Indicates the differences between pre-survey and post-survey is significant $p < .05$.
† Indicates the differences between pre-survey and post-survey is significant at Bonferroni correction level $p < .008$.

To summarize, the Programming Comprehension Assessment and Programming Motivation Survey were analyzed based on their associated subscales using either a paired sample *t*-test or a Wilcoxon signed-ranks test depending on their normality results on Shapiro-Wilk tests. Programming Comprehension Assessment data showed that participants' posttest scores on all subscales and the total were significantly higher than their pretest scores. All subscales and totals on the Programming Comprehension Assessment were found to have a large effect size (Cohen, 1988; Field, 2009; Pallant, 2007; Rosenthal, 1994). Programming Motivation Survey data showed that participants' post-survey agreement levels on all subscales and the total were significantly higher than their pre-survey agreement levels. All subscales and totals on the Programming Motivation Survey were found to have a large effect size (Cohen, 1988; Field, 2009; Pallant, 2007; Rosenthal, 1994).

Qualitative Findings and Interpretations

This study utilized two different origins of qualitative data: field notes and individual interviews. This section covers the analysis of (1) field notes, and (2) individual interviews.

Field Notes

Field notes were written in-situ when possible during the instruction and immediately after teaching the instructional units. Field notes were used to provide thick, rich descriptions and inform the selection of the individual interview participants (Creswell, 2017; Phillippi & Lauderdale, 2018). To maintain an audit trail (Creswell, 2017), a linear timeline of thoughts and events that were part of the intervention was kept in a researcher journal. My field notes on each class session were incorporated into the researcher journal and elaborated upon. In addition, notes on why codes were used and changed were also included in this audit trail. Inductive analysis was used to evaluate field notes along with the interview transcripts (Braun & Clarke, 2006).

Individual Interviews

At the conclusion of the study, one third of the participants were purposefully selected for individual interviews about their experiences in the intervention. Interviewees were selected based on my observations of participants' behavioral engagement that were also recorded in my field notes (see Table 4.12). Behavioral engagement was defined as on-task involvement and participation (Fredricks et al., 2004; Kim et al., 2015, 2017; Skinner et al., 2009). These individual interviews each took approximately 30 minutes in length and took place in my office during the class meeting schedule after the intervention was completed. The interview questions focused on the second research question and were delivered through a semi-structured interview format (see Appendix G). Each interview was open-ended in format, and I prompted the participant with a question, listened to the participant's response, and asked follow-up prompts from the interview protocol as needed.

Table 4.12. *Interviewees' Demographic Information*

Pseudonym	Age	Gender	Class.	Education Major	Prog. Exp.	Robo. Exp.	B. Engage.
Paula	21	Female	Junior	Elementary	No	No	High
Mariah	18	Female	Sophomore	Elementary	No	No	High
Randy	18	Male	Freshman	Middle	No	Yes	Medium
Katy	18	Female	Freshman	Elementary	No	No	Medium
Jennifer	19	Female	Sophomore	Elementary	No	No	Low
Simon	20	Male	Sophomore	Middle	No	No	Low

Note. Class. means Classification, Prog. Exp. means programming experience, Robo. Exp. means robotics experience, and B. Engage. means behavioral engagement.

Transcripts of the interviews were made in real-time with the Microsoft Dictate audio transcribing tool in Microsoft Word, and the interviews were also audio recorded using a video camera facing a wall to record the interviews' audio but not video. Transcriptions were checked for accuracy by me against recordings. Updates and formatting changes were made to accurately reflect the experiences and responses of each participant. The beginning parts where I explained the project and informed the participant of their rights were removed from the beginnings of the transcripts, and closing remarks were removed. In three instances with Simon, his responses were muffled or otherwise unintelligible. Notes were made within the transcript in these instances. For example, when his response for one question was muffled to the point which the microphone could not pick it up to be accurately transcribed in Microsoft Word, and the backup recording could not be used, and a note was included in brackets:

Q: Which ones were at least enjoyable?

A: [Muffled response]

Q: OK, alright, so like the pseudocoding activities?

A: Yeah.

The transcriptions were each contained in their own Microsoft Word documents. After each transcript was cleaned for formatting and clarity, the finalized transcripts were emailed privately to each participant to ensure their accuracy through member checking. Participants were asked to respond back, noting any perceived inaccuracies in the transcripts. Three of the six interview participants responded back and confirmed their transcripts. Then, coding was performed.

Analysis of qualitative data. Participants' responses in the transcripts as well as my field notes were examined through inductive analysis (Creswell, 2017; Mertler, 2017). Before formal coding began, I reviewed each transcript numerous times over a period of two weeks to become familiarized with the transcripts' contents. The transcripts and field notes were then uploaded into the Delve coding web tool.

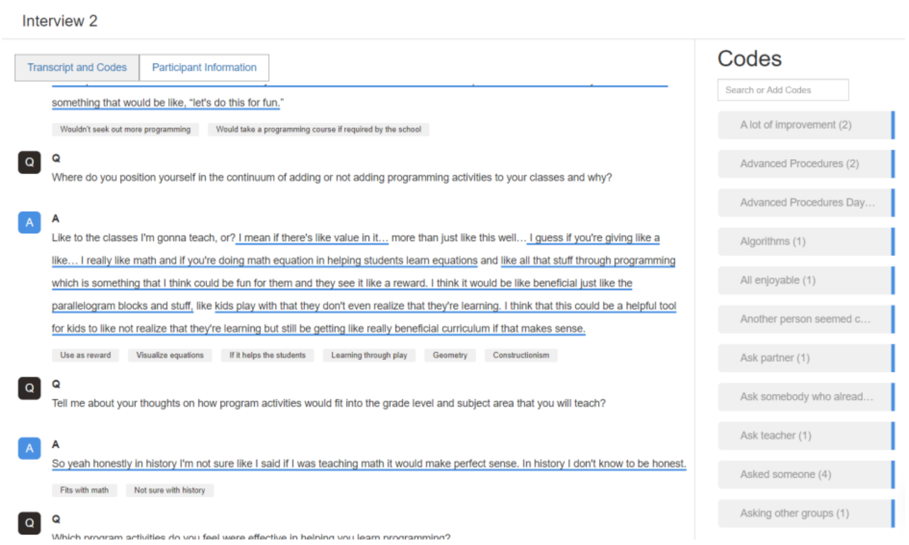
Two cycles of coding were performed. Each cycle consisted of multiple rounds of coding. Open coding was performed in the first cycle, followed by pattern coding in the second cycle (Saldaña, 2016). Table 4.13 shows the total numbers of final codes applied. These cycles and their rounds will be described in the sections below. Then, how the themes were identified will be described.

Table 4.13. *Summary of Qualitative Data Sources*

Data Sources	Final Open Codes Applied
Field Notes	16
Interview Transcripts	164
Total of Sources	180

First cycle coding. For first cycle coding, two rounds of open coding were used to separate the qualitative data into discrete parts to analyze similarities and differences

(Saldaña, 2016). The transcripts and field notes were analyzed sentence-by-sentence in this open coding cycle (see Figure 4.1 and 4.2). Each of these rounds will be explained in the paragraphs below.


Interview 2

Transcript and Codes | Participant Information

something that would be like, "let's do this for fun."

Wouldn't seek out more programming | Would take a programming course if required by the school

Q
Where do you position yourself in the continuum of adding or not adding programming activities to your classes and why?

A
Like to the classes I'm gonna teach, or? I mean if there's like value in it… more than just like this well… I guess if you're giving like a like… I really like math and if you're doing math equation in helping students learn equations and like all that stuff through programming which is something that I think could be fun for them and they see it like a reward. I think it would be like beneficial just like the parallelogram blocks and stuff, like kids play with that they don't even realize that they're learning. I think that this could be a helpful tool for kids to like not realize that they're learning but still be getting like really beneficial curriculum if that makes sense.

Use as reward | Visualize equations | If it helps the students | Learning through play | Geometry | Constructionism

Q
Tell me about your thoughts on how program activities would fit into the grade level and subject area that you will teach?

A
So yeah honestly in history I'm not sure like I said if I was teaching math it would make perfect sense. In history I don't know to be honest.

Fits with math | Not sure with history

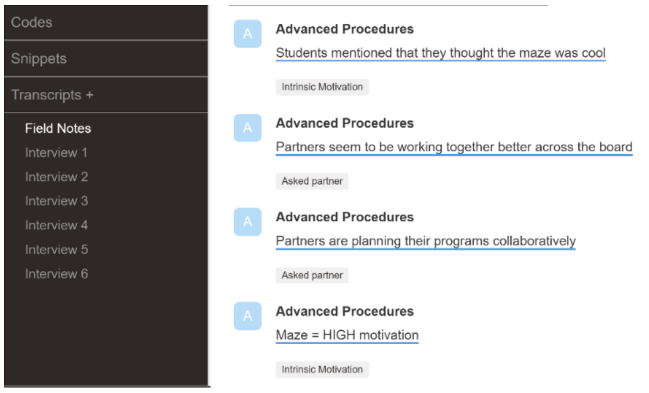
Q
Which program activities do you feel were effective in helping you learn programming?

Codes

Search or Add Codes

- A lot of improvement (2)
- Advanced Procedures (2)
- Advanced Procedures Day…
- Algorithms (1)
- All enjoyable (1)
- Another person seemed c…
- Ask partner (1)
- Ask somebody who alread…
- Ask teacher (1)
- Asked someone (4)
- Asking other groups (1)

Figure 4.1. Open coding in the Delve web tool.



A
Advanced Procedures
Students mentioned that they thought the maze was cool
Intrinsic Motivation

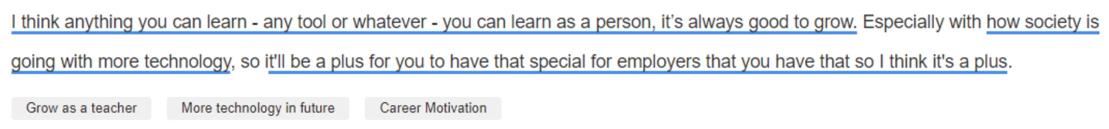
A
Advanced Procedures
Partners seem to be working together better across the board
Asked partner

A
Advanced Procedures
Partners are planning their programs collaboratively
Asked partner

A
Advanced Procedures
Maze = HIGH motivation
Intrinsic Motivation

Figure 4.2. Open coding of field notes in Delve.

Codes which summarized the experience of the participant in the transcript or my observations in the field notes were assigned to the qualitative data (Bloomberg & Volpe, 2016). Notes about the meanings of codes and topics of interest to review in further rounds of coding were kept in the researcher journal as a timeline of thoughts and events that occurred during the coding process (Lincoln & Guba, 1985; Merriam, 1998). In some instances, more than one code was applied for different aspects of a sentence through a coding process known as splitting (Saldaña, 2016). According to Saldaña (2016), splitting is a "meticulous line-by-line coding" technique that is used to provide more specific codes to transcripts (p. 229). For example, Figure 4.3 illustrates how the codes of *More technology in future* and *Career Motivation* were applied to the second sentence.

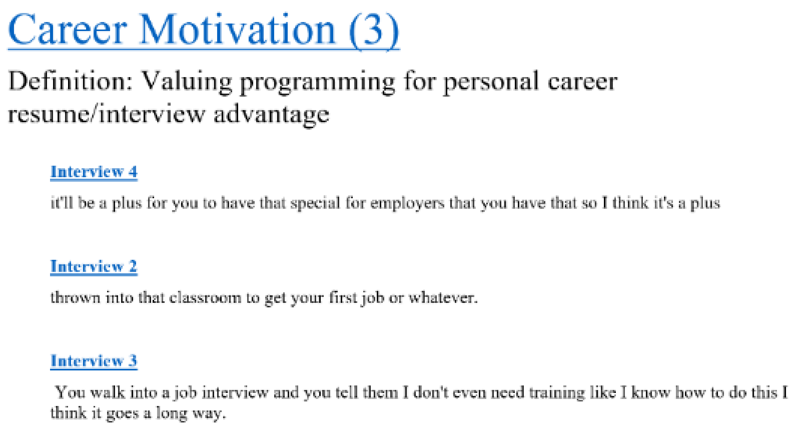


I think anything you can learn - any tool or whatever - you can learn as a person, it's always good to grow. Especially with how society is going with more technology, so it'll be a plus for you to have that special for employers that you have that so I think it's a plus.

Figure 4.3. Split coding in Delve.

The first round of coding resulted in 193 preliminary codes. After peer debriefing (Lincoln & Guba, 1985; Shenton, 2004) with the dissertation chair, seven revisions were made to these first codes. For example, the code *Math thought process* was changed to *Translates math* after a peer-debriefing conversation where it was decided that the code could be updated to better describe the excerpt, which noted the translation of math from an abstract form to a concrete one for students.

A second round of open coding was performed where the experiences of the participants were captured. During this round of coding, some codes were discarded or combined to encapsulate participants' responses more accurately (Saldaña, 2016). Figure 4.4 shows an example of the coding schemes. For example, the Round 1 code of *All enjoyable* was discarded and its contents were combined with the code *Enjoyed all* *activities.* The code *STEM/technology jobs going to become more important* was subsumed into the *More technology in future* code in the second round. During this round, codes were also renamed to align more directly to the second research question. For example, the code *Improve skills to become more employable* from Round 1 was updated to the code of *Career Motivation* in Round 2 to better reflect the subscales used to evaluate the second research question. All changes to codes were recorded with analytic memos. This second round of coding resulted in 180 unique codes. I met with the dissertation chair, and peer-debriefing (Lincoln &Guba, 1985; Shenton, 2004) was again performed to review the analytic memos on the changes and to ensure the integrity of each of my codes.



Career Motivation (3)

Definition: Valuing programming for personal career resume/interview advantage

Interview 4

it'll be a plus for you to have that special for employers that you have that so I think it's a plus

Interview 2

thrown into that classroom to get your first job or whatever.

Interview 3

You walk into a job interview and you tell them I don't even need training like I know how to do this I think it goes a long way.

Figure 4.4. Example of coding schemes.

Second cycle coding. The second cycle consisted of two rounds of pattern coding. Pattern coding is used to condense large amounts of data into smaller units to develop categories and then themes (Saldaña, 2016). In this cycle, pattern coding was used to filter the first cycle codes down into pattern codes, shown in Figure 4.5.

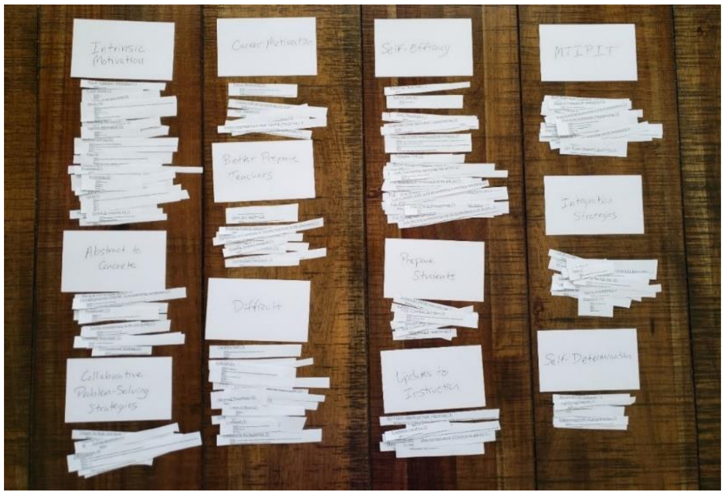


Figure 4.5. Sorting of open codes into pattern codes.

Each pattern code consisted of multiple sub-codes from the first cycle. I aligned each open code to a pattern code category based on a definition, as shown in final form in Table 4.14. For example, the pattern code of *Programming Embodies Abstract Concepts* contained codes that illustrated participants' perceptions about taking abstract formulas or equations and seeing them embodied through programming. To categorize codes into pattern codes, I first exported the finalized first cycle codes out of Delve. Then, as depicted in Figure 4.4, I compared open codes to align the open codes with the evolving pattern codes (Lincoln & Guba, 1985). A total of four codes from the open coding cycle could not be categorized due to their insufficient usefulness or insignificance for describing participants' experiences, and they were discarded (e.g., *prefer exactness*) (Saldaña, 2016). During the pattern coding process, a note was made in my journal to keep track of decisions that were made about the codes' meanings and relationships (Bazeley, 2013; Mertler, 2017). Peer debriefing (Lincoln & Guba, 1985; Shenton, 2004) was again performed with my dissertation chair, which led to more specific pattern code titles as well as the reorganization of different sub-codes to align to more suitable pattern codes. For example, the pattern code *Self-Efficacy* was split into pattern codes *Low Self-Efficacy* and *Increased Self-Efficacy*; the pattern code of *Mazes* was discarded, and its sub-codes were added to the *Challenges* pattern code. Updates to the verbiage of the codes' definitions were also made. For example, the *Robotics Visualize Abstract Concepts* pattern code's use of the word "sentiments" in the definition "Codes which illustrated participants' sentiments about taking abstract formulas or equations and seeing them visualized through robotics" was updated to "perceptions" in order to remove confusion relating to the two definitions of "sentiments." This change was recorded in the researcher journal notes as follows:

> The term *sentiments* in the definition for the *Robotics Visualize Abstract Concepts* pattern code was updated to the term *perceptions* due to the recommendation that *sentiments* may confuse readers with its two different definitions (attitude/perception toward something versus feelings of tenderness).

These peer debriefing (Lincoln & Guba, 1985; Shenton, 2004) changes filtered 176 of the unique open codes from the first cycle into the 20 pattern codes. After peer debriefing (Lincoln & Guba, 1985; Shenton, 2004) and a second round of pattern coding, these were finalized into 22 pattern codes. These 22 finalized pattern codes are displayed in Table 4.14. Once the pattern codes were well-defined, peer debriefing (Lincoln & Guba, 1985; Shenton, 2004) was again performed, and the individual codes that comprised each pattern code were again analyzed for alignment and duplicity. For example, the open codes *Fits with math*, *Use with math*, and *Geometry* were moved into the pattern code *Single Subject Integration Strategies*.

Table 4.14. *Cycle 2 – Final Pattern Codes*

Pattern Codes	Pattern Code Definitions	Example Excerpt
Advantages in Job Seeking	Codes that denoted job, resume, and career skill value (preservice teacher-centered)	"…so, it'll be a plus for you to have that special for employers that you have that, so I think it's a plus." – Randy
Autonomy	Codes highlighting participants' preferences to solve problems in their own ways	"Students asked if they have to solve a particular way (rotations, degrees, seconds) or if they were allowed to change – preferred degrees." – Field Notes
Better Educator	Codes about how participants perceived learning programming could better them to grow as educators for their students	"I think anything you can learn - any tool or whatever - you can learn as a person, it's always good to grow." – Randy
Blank Slate	Codes acknowledging participants' initially non-existent understanding of programming	"So, I had like a blank slate and now I kind of understand..." – Katy
Challenges	Codes highlighting participants' enthusiasm for the challenges	"I guess the challenges were fun…" – Randy
Collaboration Strategies	Codes highlighting participants' collaborative strategies (partner, other group, etc.)	"I just worked with my partner and like used her insight use my insight together…" – Paula
Cross-Curricular Integration Strategies	Codes representing participants' cross-curricular subject integration ideas for their future classrooms	"You could do like longitude and latitude. But you could do that…voyages of different explorers. You could talk about the mileage, and you could actually kind of have like on a scale, and I didn't think about it that way, but it was pretty interesting." – Randy

Table 4.14. *Cycle 2 – Final Pattern Codes* Continued.

Pattern Codes	Pattern Code Definitions	Example Excerpt
Decisively Committed to Integrate	Codes associated with participants' decisive commitment to integrating programming	"I don't know exactly how it would fit in, but I know I could definitely like find a way once I get their curriculum. Like I would love to find a way." – Katy
Difficulty	Codes noting difficulty or confusion with the programming activities and challenges	"When we got into the more difficult stuff like the loops." – Katy
Extra Effort	Codes which demonstrated participants' extra effort while learning programming	"Well I know that my partner for this Googled like formulas…" – Jennifer
Foundational Knowledge	Codes noting the basic or foundational content was important (i.e. Basic Procedures, lectures, etc.)	"It was most valuable starting with the basics everything just leading up to the final thing just everything adding together was the most effective thing for me personally." – Mariah
Help-Seeking	Codes which demonstrated participants' strategies for getting help when experiencing a problem (i.e. another group)	"If I wasn't sure about something, I would go ask somebody who got it already, got finished [with] the course." – Simon
Hesitant to Integrate	Codes associated with participants' hesitant feelings about integrating programming or feelings that they needed to learn more before integrating programming into teaching	"So, yeah, honestly in history I'm not sure like I said if I was teaching math it would make perfect sense. In history I don't know to be honest." – Jennifer

Table 4.14. *Cycle 2 – Final Pattern Codes* Continued.

Pattern Codes	Pattern Code Definitions	Example Excerpt
Increased Self-Efficacy	Codes that related to increased self-efficacy (i.e., feelings of confidence toward learning more programming or self-efficacy when not understanding the programming activity or challenge)	"Yeah, a little bit more confidence. I think it [confidence] has definitely grown since we started with the programming." – Randy
Interest	Codes for excerpts demonstrating interest	"Probably when we learned to get them to talk. I think it was cool how… I think it added more of a sense of like depth to it, maybe? Not just in moving around like they were like moving and talking and it was like really interesting to see like a box do that really." – Katy
Low Self-Efficacy	Codes that related to low self-efficacy (i.e., feelings of confidence toward learning programming)	"Oh, it [self-efficacy] was definitely at a zero before." – Paula
Options in Job Seeking	Codes that denoted increased options while job seeking	"I think that it's like a unique skill set to have when you're like applying as a teacher anywhere… like, maybe be able to be thrown into that classroom to get your first job or whatever." – Jennifer
Prepares Students for Future Careers	Codes which noted learning programming would help preservice teachers' future students learn and be better prepared for their futures/jobs (student-centered)	"I think honestly like the stem program and like that's gonna be the more like… the jobs that everyone's gonna look forward to as like technology advances. So, I feel like children need to learn how to do it." – Mariah

Table 4.14. *Cycle 2 – Final Pattern Codes* Continued.

Pattern Codes	Pattern Code Definitions	Example Excerpt
Programming Embodies Abstract Concepts	Codes which illustrated participants' perceptions about taking abstract formulas or equations and seeing them embodied through programming	"I think it's interesting how it translates from like a math equation…" – Jennifer
Robotics to Visualize Abstract Concepts	Codes which illustrated participants' perceptions about taking abstract formulas or equations and seeing them visualized through robotics	"…it's like a physical way, it shows them like visual, like they'll be able to see like you do this you add this and the robot does something." – Paula
Single Subject Integration Strategies	Codes representing participants' single subject integration ideas for their future classrooms	"Maybe like how kids think through math… so like if you have like 1 movement block and two movement blocks is gonna move like 2 blocks." – Katy
Updates to Instruction	Codes that related to updates students suggested for the programming instruction	"I would make it longer… make it longer, maybe 6 weeks and that way you can go slow because like I know every not everybody in the class knew everything on how to do it in this pace and like I didn't know every single answer right off the bat but like I think like if we went like slower and it would just be more beneficial." – Simon

Identifying themes. With the pattern codes finalized, I sorted the pattern codes to illuminate categories and themes. I sorted these pattern codes in a fluid and dynamic process which allowed for flexibility (Corbin & Strauss, 2008). In a code mapping process described by Saldaña (2016), “categories of categories” in “superordinate and subordinate arrangement” (p. 278) were created by moving around the pieces of paper for each pattern code. Pattern codes were united into categories. The categories were analyzed, and themes were revealed, as shown in Figure 4.6.

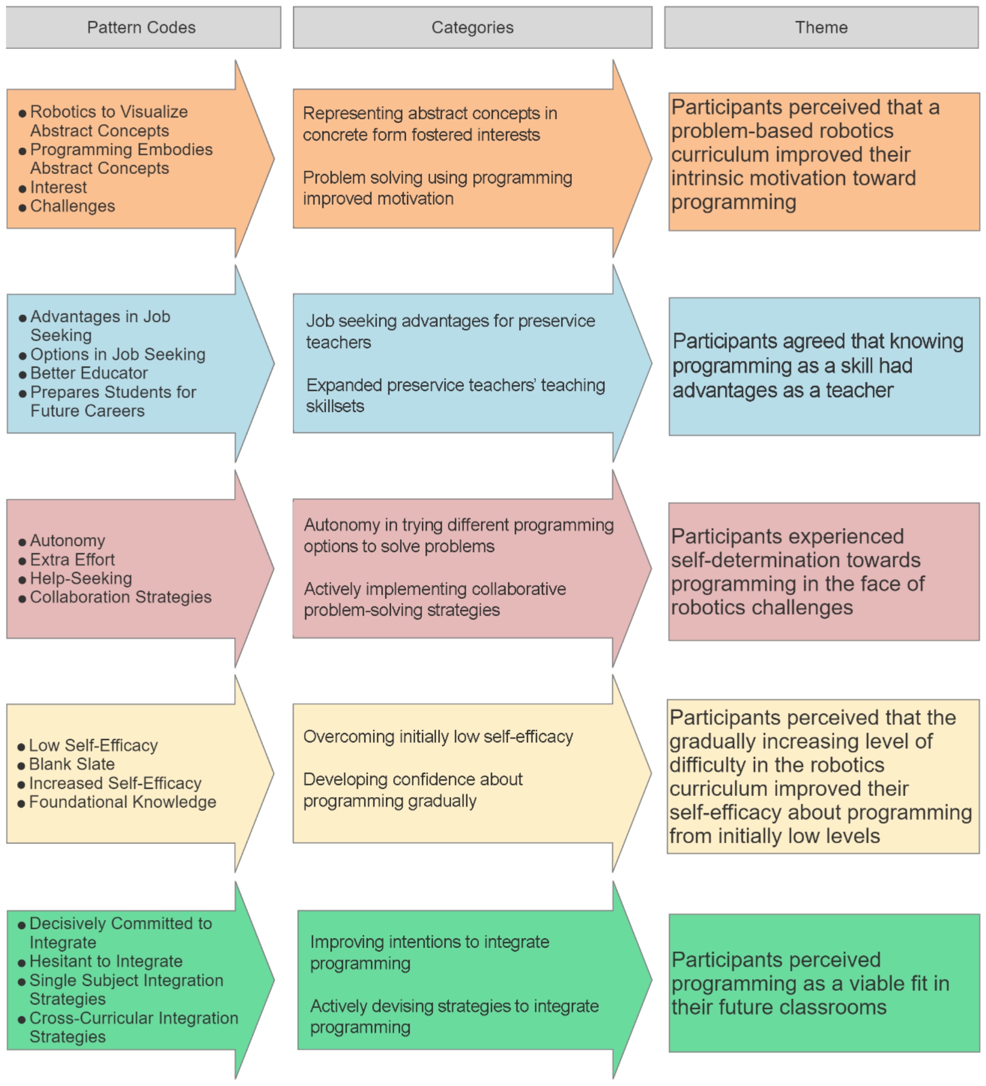

Pattern Codes
Categories
Theme
Robotics to Visualize Abstract Concepts
Programming Embodies Abstract Concepts
Interest
Challenges
Representing abstract concepts in concrete form fostered interests
Problem solving using programming improved motivation
Participants perceived that a problem-based robotics curriculum improved their intrinsic motivation toward programming
Advantages in Job Seeking
Options in Job Seeking
Better Educator
Prepares Students for Future Careers
Job seeking advantages for preservice teachers
Expanded preservice teachers' teaching skillsets
Participants agreed that knowing programming as a skill had advantages as a teacher
Autonomy
Extra Effort
Help-Seeking
Collaboration Strategies
Autonomy in trying different programming options to solve problems
Actively implementing collaborative problem-solving strategies
Participants experienced self-determination towards programming in the face of robotics challenges
Low Self-Efficacy
Blank Slate
Increased Self-Efficacy
Foundational Knowledge
Overcoming initially low self-efficacy
Developing confidence about programming gradually
Participants perceived that the gradually increasing level of difficulty in the robotics curriculum improved their self-efficacy about programming from initially low levels
Decisively Committed to Integrate
Hesitant to Integrate
Single Subject Integration Strategies
Cross-Curricular Integration Strategies
Improving intentions to integrate programming
Actively devising strategies to integrate programming
Participants perceived programming as a viable fit in their future classrooms

Figure 4.6. A concept map of the coding process.

The pattern codes of *Difficult* and *Updates to Instruction* were not relevant to motivation and were aligned to interview questions about participants' perceptions of the curriculum that were designed to guide future curriculum improvement. They have therefore been addressed in the Curriculum Design Implications section of this dissertation and were not used to support any categories or themes.

By reviewing participants' interview responses, a theme was uncovered which explained participants' intrinsic motivation. Participants explained that their interest and enjoyment increased, that the authentic problems in the intervention that they solved with educational robotics were fun, and that the ability of educational robotics to represent abstract concepts was interesting. The incorporation of four pattern codes (*Robotics to Visualize Abstract Concepts*, *Programming Embodies Abstract Concepts*, *Interest*, and *Challenges*) led to categories associated abstract concepts in concrete form being interesting, and problem-solving using programming being motivating. From those categories, the theme that participants perceived that a problem-based robotics curriculum improved their intrinsic motivation toward programming was uncovered.

In addition, participants' interview responses showed that participants perceived that learning programming through educational robotics would provide them with an attractive skillset in interviews, more options in the job market outside of their planned certification area, and the ability to better teach and prepare their future students. Incorporating four pattern codes (*Advantages in Job Seeking, Options in Job Seeking, Better Educator,* and *Prepares Students for Future Careers*) led to categories associated with participants' perceptions that they had increased their advantages and options in the job market and they had expanded their future teaching potential. In turn, the theme showing that participants agreed that knowing programming as a skill had advantages as a teacher was revealed.

Qualitative data showed that participants used collaborative problem-solving strategies, preferred autonomy in solving problems, and put forth extra effort while programming. Incorporating four pattern codes (*Autonomy, Extra Effort, Help-Seeking,* and *Collaboration Strategies*) led to categories associated with autonomy in trying different programming options to solve problems and actively implementing collaborative problem-solving strategies. The theme highlighting that participants experienced self-determination towards programming in the face of robotics challenges was revealed.

Participants' interview responses noted that at the beginning of the intervention, participants did not have confidence in their programming skills, but by the end, those views had changed. Participants noted that the foundational knowledge and skills that they learned could be relied upon as the difficulty of the units increased, which built their self-efficacy. Incorporating four pattern codes (*Low Self-Efficacy, Blank Slate, Increased Self-Efficacy,* and *Foundational Knowledge*) led to categories associated with how participants had overcome initial low levels of self-efficacy, and the gradually increased level of difficulty of the units developed participants' confidence. In turn, the theme reflecting that participants perceived that the gradually increasing level of difficulty in the robotics curriculum improved their self-efficacy about programming from initially low levels was uncovered.

Reviewing participants' interview responses uncovered decisively positive as well as more reserved commitments to integrate programming into their future classrooms.

Participants' interview responses also revealed that they had already brainstormed specific integration ideas for their subject area and grade level. Incorporating four pattern codes (*Decisively Committed to Integrate*, *Hesitant to Integrate*, *Single Subject Integration*, and *Cross-Curricular Integration Strategies*) led to the categories about participants' improving intention to integrate programming, and the ways in which they had devised instructional strategies for using programming in their future classrooms. Based on these categories, the theme illustrating that participants perceived programming as a viable fit in their future classrooms was generated from these categories.

Validating and finalizing the themes. As themes were identified, thick, rich description, an audit trail, peer-debriefing, and member checking were used to evaluate the themes' validity. Thick, rich descriptions (Bazeley, 2013; Creswell, 2017; Mertler, 2017) in the form of verbatim quotes from the participants were used to support the themes. A researcher journal was used to maintain an audit trail documenting the events and decisions which occurred during the study and subsequent analysis (Lincoln & Guba, 1985; Merriam, 1998). The researcher journal was used to justify codes as well as compare and supplement the thick, rich descriptions. Peer debriefing (Lincoln & Guba, 1985; Shenton, 2004) was performed with my dissertation chair, which aligned codes and focused the language of the themes. Member checking (Creswell, 2017; Merriam, 1998; Mertler, 2017) occurred via email because of the COVID-19 pandemic and was used to allow participants to verify the accuracy of their interview transcripts as well as the findings. Interviewees were first emailed the interview transcripts and were instructed to reply with critiques or questions. Three of the six interviewees emailed back to confirm the accuracy of their transcripts, but no additional insights were provided. The three other interviewees did not respond. Then, interviewees were asked to review the themes and categories and email me back with critiques or questions. Three of the six interviewees emailed back to confirm the accuracy of the themes and categories, but no additional insights were provided. The other three interviewees did not respond. The themes and categories were then finalized.

Themes

Themes were derived from the finalized categories. Categories arranged by common responses shared by multiple participants were composed into themes related to the second research question (Saldaña, 2016). In the following section, themes are presented accompanied by meaningful verbatim quotations from the individual interviews, attributed to participants via pseudonyms, and excerpts from the field notes are indicated in the text as field note entries that have been chosen to support the themes by presenting context (Creswell, 2017). Interview quotations are accompanied by a pseudonym to protect the participants' identities (i.e., Paula, Simon, Randy, etc.). Five overarching themes were revealed from the qualitative analysis. Through the evaluation of the field notes and individual interviews, it was revealed how and to what extent the educational robotics intervention influenced preservice teachers' motivation related to programming. Interview data indicated the following themes: (1) participants perceived that a problem-based robotics curriculum improved their intrinsic motivation toward programming, (2) participants agreed that knowing programming as a skill had advantages as a teacher, (3) participants experienced self-determination towards programming in the face of robotics challenges, (4) participants perceived that the gradually increasing level of difficulty in the robotics curriculum improved their self-efficacy about programming from initially low levels, and (5) participants perceived programming as a viable fit in their future classrooms. These themes, their associated categories, and example open codes which contributed toward them are outlined in Table 4.15.

Table 4.15. *Summary of Themes, Categories, and Example Open Codes*

Theme	Categories	Example Open Codes
Participants perceived that a problem-based robotics curriculum improved their intrinsic motivation toward programming	Representing abstract concepts in concrete form fostered interests	Translates from math Physical way to teach abstract Visualize equations
	Problem solving using programming improved motivation	Cool Interesting Authentic problem-solving
Participants agreed that knowing programming as a skill had advantages as a teacher	Job seeking advantages for preservice teachers	Career motivation Unique skillset High demand
	Expanded preservice teachers' teaching skillsets	Grow as a teacher Technology will be more relevant in the future Would come in handy as a teacher
Participants experienced self-determination towards programming in the face of robotics challenges	Autonomy in trying different programming options to solve problems	Self-Determination Reviewed class resources Googled formulas
	Actively implementing collaborative problem-solving strategies	Asked a partner Asked other groups Ask somebody who already completed it for help

Table 4.15. *Summary of Themes, Categories, and Example Open Codes* Continued.

Theme	Categories	Example Open Codes
Participants perceived that the gradually increasing level of difficulty in the robotics curriculum improved their self-efficacy about programming from initially low levels	Overcoming initially low self-efficacy	Beginning: Did not know what to expect Beginning: Blank slate Beginning: Didn't know much programming
	Developing confidence about programming gradually	Basics and foundational knowledge effective Lectures were effective End: Programming knowledge has grown
Participants perceived programming as a viable fit in their future classrooms	Improving intentions to integrate programming	Sees potential for use in classroom Definitely add programming to future teaching
	Actively devising strategies to integrate programming	Math Science Use as reward

Theme 1: Participants perceived that a problem-based robotics curriculum improved their intrinsic motivation toward programming

This theme describes how participants perceived that the problem-based educational robotics activities in the curriculum improved their intrinsic motivation toward programming. Kim et al. (2015, 2018) and Kucuk and Sisman (2018) emphasized that preservice teachers' intrinsic motivation should be kept at high levels throughout robotics activities. Participants experienced increased intrinsic motivation toward programming. Intrinsically motivated learners work toward attaining a goal because of their internal enjoyment in completing the task (Amabile et al., 1994; Law et al., 2010). Interviewees described their intrinsic motivation through characterizations of the educational robotics by often referring to them as being “fun,” “cool,” or “interesting,” and therefore intrinsically motivating. “I've taken technology classes before and if we did something like this it would have been like 10 times cooler,” Simon stated in his interview. Educational robotics were not mentioned by me in any of the individual interview questions or follow-up prompts; however, the educational robotics activities and challenges were the elements of the curriculum that seemed to motivate participants the most. For example, Mariah explained in her interview, “Honestly, I think the whole experience is really fun and just being able to move the… program things so you could move a robot. I think that’s a really cool thing to do.” Overall, the participants found programming educational robotics to be intrinsically motivating.

Theme 1 conveyed how participants perceived that a problem-based robotics curriculum improved their intrinsic motivation toward programming. The following sections will outline the categories subsumed in support of this theme: (1) representing abstract concepts in concrete form fostered interests, and (2) problem solving using programming improved motivation.

Representing abstract concepts in concrete form fostered interests. Half of the interviewees (*n* = 3) commented that an element they found interesting was the ability of the programming and educational robotics to take abstract concepts and make them concrete for learning. This category is related to Theme 1 because interest aligns with intrinsic motivation (Ryan & Deci, 2000). Constructivism includes the building of abstract knowledge structures in a learner’s mind through concrete experiences (Piaget, 1967, 1973). Educational robotics have been used to demonstrate physical representations of abstract concepts for learners (Bers, 2010; Bers et al., 2002; Han, 2013). The idea of using educational robotics to represent abstract concepts was noticed by participants. For example, Paula mentioned that she perceived robotics a tool for concrete representation in her interview: “…it’s like a physical way, it shows them like visual, like they'll be able to see like you do this you add this and the robot does something.” An excerpt from Jennifer’s interview summarized participants’ positive perspectives on the transition of abstract concepts into concrete actions:

> I liked when there was a maze and we had to make an equation to figure it out because I think it’s interesting how it translates from like a math equation to like actually like seeing it happen in front of your eyes.

The transition of math to something observable being interesting was not unique. Further validating this category and overall theme, Katy noted in her interview that she liked the computational thinking aspect of programming the robots and watching them perform those programs, as well:

> Maybe like how your thought processes are like related to like what the robots are doing. I never knew about robots really but learning how to program and how it kind of like went along with like people[’s] like thought processes I thought that was really interesting.

Participants’ recognition of the process of taking abstract ideas and making them more concrete took another form as well. Similarly, Randy enjoyed the pseudocode process. In his interview, he explained that he appreciated the process of writing the pseudocode and then translating it into the programming language, making it more concrete:

> I forget the word, what it was called…pseudocode. And actually doing that is exactly the same things [sic] like putting the code into the computer so I thought [it] was good visual representation of that, so I appreciate that.

The idea of translating abstract concepts into concrete processes was also reflected in my field notes. For example, I noted that "Many students [are] using math as opposed to guess and check," choosing the workflow of writing abstract math formulas and then transitioning their programs from math formulas to programs as opposed to tinkering and writing the programs based on the concrete actions of their robots. Additionally, I made a field note about how there was confusion over presenting a complicated variables algorithm without focusing on the math and pseudocode behind it. Altogether, these data indicated that while participants solved problems, they were interested in seeing abstract thinking translated into concrete representations either in the programming language or in the movements of the educational robotics. Participants' interest links to intrinsic motivation and supports Theme 1.

Problem solving using programming improved motivation. The problems participants solved improved their intrinsic motivation. This category aligns with Theme 1 because it describes a source of participants' intrinsic motivation. Authentic problems in this context are those which combine content from science and math areas to be solved with the aid of educational robotics (Kopcha et al., 2017). Learners are most likely to learn programming skills when educational robotics tasks are introduced in a context that necessitates problem-solving through authentic science and math skills (Pea, 1987). All interviewees (*n* = 6) articulated that the authentic activity and challenge elements of the curriculum were intrinsically motivating in their responses to question #2, "Tell me about your experiences with the programming activities in the course." In addition, programming the robots to resolve authentic problems or challenges (e.g., color testing, mazes) were the aspects of the intervention which participants most often characterized as interesting or fun.

For example, partners found it enjoyable to program the robot to say the name of the color the sensor detected using a switch and a loop to control the flow of the program depending on the color input. In reference to this activity, Katy stated the following in her interview:

> Probably when we learned to get them to talk. I think it was cool how… I think it added more of a sense of like depth to it, maybe? Not just in moving around like they were like moving and talking and it was like really interesting to see like a box do that really.

Overall, half of the interviewees ($n = 3$) mentioned that they were interested in not only seeing the robots move, but some authentic tasks such as programming them to identify different colors and speak were intrinsically motivating aspects as well. My field notes confirmed the interview data and noted that participants were energized and interested in checking the colors of different folders they had in their backpacks, as well as different objects throughout the room. However, it was noted that some participants quickly tired of hearing the colors repeatedly announced by the robots. Interestingly, some groups – outside of my classroom instruction – figured out how to record their own sounds and write programs that played their recordings for different colors, exceeding the requirement of the activities. This demonstrated students' interest in the activity.

The authentic problems participants solved in the challenges were the intrinsically motivational aspect cited by all interviewees (*n* = 6). Interviewees pointed to the authentic problems as being fun and interesting. "I guess the challenges were fun and figuring out ways to use the program," Randy stated. Two challenges that were most often mentioned by interviewees as being intrinsically motivating were the Maze Challenge in the Advanced Procedures unit and the Maze with Variables Challenge in the Variables unit.

Participants were motivated by the mazes. "I think the most enjoyable part was we had to do the maze," Randy noted in his interview about the Maze Challenge. Paula reinforced Randy's enjoyment of the Maze Challenge. Paula mentioned in her interview that she enjoyed working through the Maze Challenge early in the intervention because it gave her an opportunity to exercise her new, yet limited programming skills. The experiences shared by Randy and Paula further validate the importance of this category related to solving problems and Theme 1.

The Maze with Variables Challenge was also described as being motivating. This challenge took the original Maze Challenge and added color swatches to the floor of the maze. Participants had to program their robots to turn in a specific direction or stop depending on which color their robot's color sensor picked up. Simon explained in his interview what he liked from the intervention:

> Thinking back…like each time you use the robots to navigate a different course and like just like learning about like how to do every single course having to stop [at a] certain color and have it [to] make like sharp turns and just like being able to like fully understand how to use it in particular.

Mariah echoed Simon's statement in her interview: "The last one…Just seeing a robot move whenever it hits the color or like having it stop. I think it's a really interesting way [to learn]." In summary, these data indicated that problem solving using programming improved participants' intrinsic motivation.

Theme 2: Participants agreed that knowing programming as a skill had advantages as a teacher

This theme describes participants' agreement that knowing programming as a skill had advantages for them professionally as teachers. Learners who have career motivation related to a topic see that topic's relevance to their future careers (Arwood, 2004; Glynn et al., 2009). Preservice teachers who have learned programming with the aid of educational robotics have experienced meaningful increases in their STEM career motivation (Kim et al., 2015). Interviewees described their career motivation through references to the personal career and teaching advantages of learning programming. For example, Randy explained in his interview that learning programming as a teacher was advantageous: "Especially with how society is going with more technology, so it'll be a plus for you to have that special for employers that you have that [sic], so I think it's a plus." The following sections will outline the categories subsumed in support of this theme: (1) knowing programming had job seeking advantages for preservice teachers, and (2) knowing programming expanded preservice teachers' teaching skillsets.

Job seeking advantages for preservice teachers. Interviewees expressed their perceptions of the value of learning programming in terms of obtaining more advantages on the job market. This category aligns to Theme 2 because it explains a personal professional reason behind why the participants valued knowing programming. Career motivation is important to learners' long-term goals, with professional success being a primary reason why learners pursue college studies (Glynn et al., 2011). Educational robotics have been used in prior research to increase learners' career motivation (Goh & Ali, 2014). Advantages or options in job seeking are pertinent to participants' career motivation. The two main reasons interviewees reported that they were motivated to learn and use programming for their own benefit stemmed from (1) being more marketable in interviews, and (2) creating more opportunities for themselves for positions outside their licensure area. Overall, half the interviewees (*n* = 3) viewed learning programming as a skill that would be valuable in obtaining their future employment. Interviewees noted that the future of the economy being tied to the growth of technology was a factor that impacted their career motivation. In his interview, for example, Simon explained that knowing programming could make him more desirable in a job interview: "You walk into a job interview, and you tell them I don't even need training like I know how to do this I think it goes a long way." While some interviewees noted that programming was a skill that employers would be impressed by, others noted that learning programming might give them more options on the job market for positions different from their licensure area. For example, Jennifer explained in her interview:

> I think that it's like a unique skill set to have when you're like applying as a teacher anywhere because like I know at my high school that the tech ed. teachers were like in high demand, but then nobody wanted to teach it, so I think that it's like, you need to have that in like, maybe be able to be thrown into that classroom to get your first job or whatever.

Participants perceived the programming experiences gave them a more diverse skillset to get their first teaching jobs. From adding confidence in interviews to creating a greater number of opportunities to get a foot in the door in schools for positions outside of their licensure area, participants confirmed their career motivation through their recognition of the employability of programming-literate educators.

Expanded preservice teachers' teaching skillsets. Most interviewees (*n* = 4) expressed that learning programming through educational robotics would help them expand their future teaching skillsets to benefit their future students. This category aligns to Theme 2 because it explains an altruistic professional reason for why the participants valued knowing programming. Increased knowledge of teaching, such as teaching strategies, is a factor which can motivate teachers to stay in their career (Sinclair, 2008). Programming offered new teaching strategies, among other things, to participants. Interviewees became motivated to learn programming through educational robotics because it would allow them to provide better lessons for their students. Statements from interviewees identified the added teaching options which programming offered. For example, "I feel like it would come in handy a lot with me going into teaching," noted Katy. Learning with educational robotics also promoted personal growth as a teacher. In his interview, Randy stated, "I think anything you can learn – any tool or whatever – you can learn as a person, it's always good to grow." Jennifer echoed this perspective in her interview and reinforced how learning programming would further benefit the participants' future students:

> I think like because technology is – even since I was like in kindergarten keep coming into the classroom – more and more and more and it's going to be like a

> bigger thing and understanding it will help you like to better the education of your students.

In particular, the use of programming and educational robotics to create interesting and engaging lessons was a common idea. “I think it could be fun for them,” remarked Jennifer in her interview. Paula thought back to the integration example videos, which showed teachers using programming in various subjects. “Seeing all the different videos that we watched seeing teachers incorporate it even in like gym class, I thought it was a really good way to get students like interested in learning whatever topic it was,” she explained in her interview. Mariah added she was motivated to integrate programming because it could help with getting students’ attention within a lesson, “Just make lessons really interesting and just to keep them engaged.” Participants noted perspectives that programming activities offered a teaching tool to enhance their lesson plans to grab students’ attention and engage them, making their teaching better.

Recognition of the importance of participants preparing their students for the future technology-driven economy was common. In her interview, Katy stated that the aspects of “Math and learning technology” were important for students to learn. Katy explained:

> We are getting more in[to] the future [and] technology is getting more ingrained in our lives. Technology and like, learning how to program stuff because, like I said, like the more and more into the future stuff like [progresses], that's going to be more relevant.

Mariah echoed this perspective of the importance of preparing future students for a technology-driven job market in her interview: “I think honestly like the STEM program and like…the jobs that everyone’s going to look forward to as like technology advances. So, I feel like children need to learn how to do it.” Mariah’s perspective further validates the importance of this category and theme.

Using programming to help provide differentiated methods of instruction was another commonality in the interview responses. Jennifer stated in her interview that she would use programming “If there’s like value in it” such as using programming and educational robotics as a reward activity for students or having students learn through play. “I think it would be like beneficial just like the parallelogram blocks and stuff, like kids play with that they don't even realize that they're learning,” Jennifer added. Paula also explained in her interview the benefits of programming educational robotics as an added teaching strategy to help students learn without knowing it:

> I feel like because students like don't always like… I think it's a way to like get them to learn without realizing that they're learning something 'cause they're just like oh cool it’s robots like they're not really thinking about the fact that they are learning something through using them.

The idea shared by Paula further validates this category and overall theme. As outlined above, one reason participants valued learning programming through educational robotics was because it could help them become better teachers to improve their future teaching and benefit their future students.

Theme 3: Participants experienced self-determination towards programming in the face of robotics challenges

This theme describes interview responses that indicated participants’ experiences with self-determination. Learners with self-determination feel as though they have control over their learning (Black & Deci, 2000). Teachers must have self-determination to be successful when integrating technology (Cullen & Greene, 2011). Field note entries highlighted participants' preference for autonomy in their problem-solving solutions and participants cited using personalized problem-solving techniques and collaborative problem-solving strategies in order to solve problems. In his interview, Randy described how he would seek out peers and "compare notes with other people" as a way to identify what he was doing wrong in order to adjust his programming strategy. The following sections will outline the categories subsumed in support of this theme: (1) autonomy in trying different programming options to solve problems, and (2) actively implementing collaborative problem-solving strategies.

Autonomy in trying different programming options to solve problems. The educational robotics activities and challenges fostered the autonomy of participants to try their own unique options to solve problems. This category is related to Theme 3 because autonomy is a factor in self-determination (Black & Deci, 2000; Ryan & Deci, 1985, 2020). The educational robotics activities and challenges were designed to be able to be solved in multiple different ways, which allowed participants to experiment with different programming processes. Participants indicated that the open-ended nature of the activities and challenges fostered autonomy among participants. Randy explained in his interview the appeal of autonomy in the educational robotics programming activities: "I guess you could use the same but different program but like I guess use different ways to get to the same result." Paula explained in her interview that she was intrigued by the opportunity to exercise her new, yet limited programming skills to try out various solutions and find the one that solved the problem. "We didn't really know that much about programming yet and we had to kind of like figure out our own way to like get through the maze," Paula said. My field notes offered insights into autonomy. One field note stated that participants asked if they had to solve an activity in a particular way (using rotations, degrees, or seconds) or if they were allowed to choose their own programming method to solve the problem. The participants were excited when they were told that they could solve the challenge using their own preferred method. These data indicated that the educational robotics programming activities encouraged interviewees' autonomy in problem-solving

Actively implementing collaborative problem-solving strategies. All interviewees (*n* = 6) commented that actively implementing collaborative problem-solving (CPS) strategies contributed toward learning the programming concepts. This category is related to Theme 3 because CPS strategies can help learners' self-determination by combining their collective efforts and knowledge (Kopcha et al., 2017; Lanzonder, 2005; Witney & Smallbone, 2011). Both the aid of partners designed as part of the curriculum as well as the unplanned collaborative classroom environment were mentioned by participants in the interviews. This category is related to Theme 3 because participants' utilization of CPS strategies represented participants' additional effort toward solving a learning task, thus their self-determination. The grouping of participants into partners provided a strong aid for participants to collaborate and build upon their collective insights to solve problems. "I just worked with my partner and like used her insight and used my insight together," Paula explained in her interview. Randy also stated in his interview that his partner aided him in learning programming: "I guess I will lean on the people that [I] worked with and ask them questions." Partners who combined their insights to solve problems represented CPS strategies and self-determination.

Interviewees noted they also sought help from peers outside of their immediate partner when they did not understand something. Mariah explained in her interview that becoming stuck on a problem was the trigger for when she would ask a peer: "When I was really stuck on something that's when I was like OK, maybe I need [a person] to help me but I need someone else… it's not clicking right now." In his interview, Randy commented that "I kind of would compare notes with other people and see their thinking process and how they got their results and compare and see what I was doing and see if I can make any adjustments." In his interview, Simon explained his process for reaching outside his immediate partner for help.

> We all had somebody or some people either next to us doing it with us and like if you didn't know how to do something like maybe your partner did… but like there was always somebody in the class.

Participants described picking out peers in other pairs who had completed the activities and challenges successfully to help them. "If I wasn't sure about something, I would go ask somebody who got it, already got finished the course [sic]," Simon added in his interview. The language found in four field notes affirmed participants' interview descriptions. Three notes in particular focused on partner collaboration dynamics. My first note on partner dynamics chronologically was from the first Basic Procedures class. This entry mentioned, "Partners began working together, but [they are] still not working together as much as I would like." This note, which highlighted that partners were less collaborative during the Basic Procedures unit, is contrasted from one in the second

Advanced Procedures class: “Partners seem to be working together better across the board.” Finally, a separate observation noted, “Partners are planning their programs collaboratively.” This change in partner dynamics might be attributed to the more difficult problems given to the participants as the curriculum progressed, which required them to collaborate more. During the Control Structures unit, another note mentioned, “Some groups finished quickly while others struggled to keep their robot in a straight line. Groups [are] helping each other.” In total, these data indicated that participants used CPS strategies both between partners and between groups. These findings might indicate that as the difficulty of the problems increased, participants sought collaboration outside of their immediate partner to solve the problems. These interview excerpts highlighting CPS strategies firmly supported Theme 4 and educational robotics challenges contributing toward self-determination related to programming.

Theme 4: Participants perceived that the gradually increasing level of difficulty in the robotics curriculum improved their self-efficacy about programming from initially low levels

This theme describes how educational robotics affected the participants’ self-efficacy toward programming. Learners with self-efficacy have confidence in their ability to achieve a learning task (Bandura, 1997). Low self-efficacy can be attributed to educators using new teaching materials and their uncertainty with learning new technologies (Curzon et al., 2009; Meerbaum-Salant et al., 2013). Participants were able to overcome an uncertainty barrier to improve their programming self-efficacy. Participants described low initial levels of self-efficacy due to their perceived low comprehension of programming concepts. “So, I had like a blank slate,” Katy said about her beginning programming knowledge and skills in her interview. However, participants described how their self-efficacy related to programming increased as they developed an evolving confidence that they attributed to the gradually increasing level of difficulty of the robotics curriculum. For example, in her interview, Mariah attributed the gradually increasing level of difficulty of the robotics curriculum as being helpful: “starting with the basics, everything just leading up to the final thing, just everything adding together was the most effective thing for me personally.” The following sections will outline the categories subsumed in support of this theme: (1) overcoming low self-efficacy, and (2) developing confidence about programming gradually.

Overcoming initially low self-efficacy. All interviewees (*n* = 6) described low initial levels of self-efficacy related to programming. This category is related to Theme 4 because it explains the commonality of where participants’ self-efficacy related to programming began. Grover and Pea (2013) have found that self-efficacy related to computer science was low in educators teaching computer science concepts. Low self-efficacy may negatively impact teachers’ usage of a new technology in the classroom (Ertmer & Ottenbreit-Leftwich, 2010; Ertmer et al., 2012), which means that participants would not have been comfortable or competent enough to integrate programming before the intervention. The interviewees (*n* = 5) commonly mentioned their initial level of programming comprehension was nonexistent: “Oh, it was definitely at a zero before,” explained Paula. “I didn't have much background knowledge,” stated Mariah. “Like, I didn't know anything I didn't even know how to turn them on, so it's definitely improved,” insisted Jennifer. Simon explained the following experience in his interview:

> Well at the beginning like, I didn't really know what to expect. I don't really know but I remember we took the pretest and like I see all these codes and stuff and like I even sent the picture to my mom and I was like, ‘do you have any idea how to do this?’ And she's like, ‘what are you talking about?’ And I was like I wasn't really sure what to expect.

Participants noted that the composition of the programming curriculum contributed toward their improved self-efficacy. All interviewees (*n* = 6) stated that they felt that the educational robotics programming activities helped them a considerable amount in learning programming, removing their uncertainty in various ways. For example, Simon stated the following in his interview:

> Obviously, you know like each week something like the first week we learned how to turn it around and stop at colors, so like learning how to do all of that, like I didn't know how to do any of that.

Others agreed with this perspective in their interviews. “Oh, it’s definitely a lot better,” Jennifer stated about her self-efficacy. Katy noted, “now I kind of understand that program a little bit more…definitely, it’s grown.” “Yeah, a little bit more confidence. I think it [confidence] has definitely grown since we started with the programming,” remarked Randy. These data indicated that participants initially had low levels of self-efficacy related to programming, which they overcame throughout the intervention.

Developing confidence about programming gradually. Participants’ confidence about programming developed gradually. This development was aided by the gradual building of the difficulty of concepts in the curriculum. Learners’ self-efficacy can be increased by experiencing success completing similar learning tasks (Bandura, 1997). This category is related to Theme 4 because it shares participants' experiences and explains how their self-efficacy related to programming increased. A greater commitment to teaching is reported by teachers with higher levels of self-efficacy (Chen & Yeung, 2015; Gunning & Mensah, 2011). Almost all interviewees (*n* = 5) noted that the introductory knowledge and concepts – things they characterized with the terms "foundational," "basics," or "simple" – were the most helpful to them. Successes with these basic concepts developed participants' confidence gradually, and the basics that they learned helped them have success with more difficult problems. When asked what the most meaningful part of the curriculum was in his interview, Simon expressed a preference for the basic programming skills on which the other skills were built:

> For the most meaningful [part], I really liked the start on how to do it. It started like you could like figure it out. You can use the program on the computer to like navigate through it if you learn how to do it, and then you could just like try different things see what works [and] what doesn't, and so I think [the] foundational stuff.

This idea was common among the interviewees. Interviewees' responses explained that the basic knowledge they learned could be applied and help them be successful on the more difficult units. For example, Paula stated, "I think 'cause I'm kind of a visual person, I think just having like the slides that you provided ahead of time and then seeing that and being able to like apply it myself is probably the most valuable," in reference to the instructional presentations of basic concepts that she could apply later.

Further, the programming concepts gradually increased in difficulty level from the foundational knowledge and skills to more complex knowledge and skills, which participants explained helped with building their competence and self-efficacy. For example, Katy articulated the following experience in her interview:

> Probably the first couple [lessons] when we were learning how many centimeters is like in one rotation or like how many seconds it takes across this much distance. You're really helping conceptually building the foundations of like the other stuff that we learned.

Jennifer supported this perspective in her interview, as well. “Well, I feel like they all were valuable because they all like built onto each other, and then I feel like each time you did it like you could apply stuff from the last time.” Mariah also identified the gradual progression from the basics to more advanced concepts as being helpful in her interview: “It was most valuable starting with the basics, everything just leading up to the final thing, just everything adding together was the most effective thing for me personally.” These data show that the gradually increased difficulty of programming concepts helped build participants’ programming competence and self-efficacy gradually. Overall, participants recognized the gradual increase in the level of the units’ difficulty and how it impacted their competence, which supported improvements in their self-efficacy.

Theme 5: Participants perceived programming as a viable fit in their future classrooms

This theme describes interview responses that indicated the educational robotics programming activities affected the participants’ perceptions of programming and how it could be applied into their pedagogy. Preservice teachers who have experienced educational robotics interventions have been noted to develop increased motivation to integrate programming robots into their STEM teaching (Kim et al., 2015). Participants' perceptions about integrating programming appeared in two different areas in the interviews. The following sections will outline the categories subsumed in support of this theme: (1) improving intentions to integrate programming, and (2) actively devising strategies to integrate programming.

Improving intentions to integrate programming. Almost all the interviewees' (*n* = 5) intentions to integrate programming into teaching improved, as evidenced by each of their responses to interview question #9: "Where do you position yourself in the continuum of adding or not adding programming activities to your classes? Why?". This category is aligned to Theme 5 because it demonstrates how participants' perspectives changed on their intentions to integrate programming into teaching. Positive or negative beliefs and experiences influence teachers' intentions to integrate a technology into their instruction (Ajzen, 2005). For example, Paula summarized in her interview how her perception of programming's usefulness changed:

> Going into it when you first proposed the idea that we would be using programming and stuff in this class I didn't really think that it would be useful at all, like I didn't really understand how I can possibly even use it in teaching and how it had anything to do with teaching, but obviously going through it I realized like it is very useful so it's kind of done a complete 180 to be honest.

"I think it's more valuable now and I understand like why it helps students like learning like through math and stuff," Jennifer noted in her interview. Mariah remarked in her interview that she now felt programming should be incorporated into schools even more than it currently is: "So I originally thought like programming was like… it's already in a lot of [local school district redacted]. OK, they're doing it now. I feel like it should be incorporated just a little bit more." These interview responses demonstrate participants' improvement in their intentions to integrate programming into their teaching.

Participants' current intentions to integrate that were articulated in the interviews ranged from solid confirmations of intent to perceptions that participants needed to do more research before integrating. For example, Mariah starkly stated in her interview, "I want to add it." Others expressed their desire to integrate programming into their instruction but felt they needed further research into their future curriculum and applicable connections first. For example, Katy expressed a more reserved or hesitant intent to integrate programming, summarized in this interview statement:

> Um, I can see it being used a lot with like math and science, especially for younger kids. I feel like I haven't learned enough about it, but I can see the potential for like how programming could possibly work out in classrooms.

She further elaborated: "I could really see myself adding this to my lesson plans," and, "I don't know exactly how it would fit in, but I know I could definitely like find a way once I get their curriculum. Like I would love to find a way." These responses demonstrate participants' range of encouraging programming integration intentions.

In summation, participants' interview responses indicated that their intentions to integrate programming into their teaching improved correspondingly with their valuations of programming. Participants' intentions included more reserved responses in which participants affirmed they wanted to integrate programming but needed to learn more about their curriculum or programming more generally before doing so, to decisive intentions to integrate programming into their teaching. These improved, positive intentions among participants support this category's theme that participants perceived programming as a viable fit in their future classrooms.

Actively devising strategies to integrate programming. As outlined above, many interviewees stated strong confirmations of their intentions to incorporate programming in their future instruction. Positive attitudes about technology integration have been shown to be the strongest predictor of whether teachers integrate instructional strategies into their teaching (Palak & Walls, 2009). Ajzen (2005) suggested that a way to assess teachers' technology integration attitudes is through studying their behavior. One behavior that demonstrated attitudes and technology integration potential of most of the interviewees (*n* = 4) was that they had already brainstormed strategies for future programming integration. Interviewees' ideas for integration into their future curriculum are related to Theme 5 because they show exactly how participants envisioned fitting programming into their instruction.

Interviewees had multiple ideas for integrating programming into their future instruction, including singular subjects as well as cross-curricular connections. "I feel like there's a lot of different ways to incorporate it," posited Paula. Four interviewees shared ideas for integrating programming with math. The use of educational robotics to represent abstract math concepts was a commonality. Jennifer explained in her interview that she would use programming to teach students the different parts of math equations. Katy explained in her interview that she would use programming as an introduction to technology for her elementary students to illustrate math problems. "So, like if you have like one movement block and two movement blocks, it is going to move two blocks," she explained. Paula also noted in her interview that she would use programming to illustrate math in a more tangible way:

> I want to teach second or third grade probably, and I feel like there's a lot of different ways I could incorporate it. Probably with math even like using the algorithms and stuff...

As an example of a cross-curricular integration, Randy had an idea to integrate math into social studies through programming educational robotics. This detailed idea for a lesson plan that he shared in his interview was not based on any priming from anything similar that participants saw in the integration videos:

> You could do like longitude and latitude. But you could do that…voyages of different explorers. You could talk about the mileage, and you could actually kind of have like on a scale, and I didn't think about it that way, but it was pretty interesting. I guess I can go back to the example with um…about colonialization in America. We can talk about the different, um, probably the different British ships that came over and we could talk about how I guess like focusing for a little bit on how long they took to travel and as far as like mileage and then we can do like a fun activity with programming. A small activity that doesn't take too much time but also gives the children some programming knowledge.

These integration ideas showed that participants could imagine both single-subject and cross-curricular linkages in lesson plans they had already devised. These interview excerpts that highlighted integration ideas firmly supported Theme 5 – participants perceived programming as a viable fit in their future classrooms.

Integrating Quantitative and Qualitative Findings

The quantitative Programming Motivation Survey and qualitative individual interview findings were combined to present a better representation of RQ#2 and the intervention's effects on preservice teachers' motivation related to programming. To do this, first, I interpreted the quantitative Programming Motivation Survey results. Then, I compared these results with the qualitative individual interview themes. In this way, the qualitative data offered additional explanation to what the quantitative results implied (Creswell, 2014; Mertler, 2017). The quantitative and qualitative findings were grouped by subscale, as demonstrated in Table 4.16. Then, these combined findings were used to investigate research question #2: How and to what extent does educational robotics influence preservice teachers' motivation related to programming? Through this process, I found that the quantitative data that denoted an increase in motivation in each of the subscales was supported by the qualitative data. Further, the qualitative data offered insights into participants' statistical increases in motivation through statements describing their experiences.

Through this method, the qualitative data and findings were used to emphasize and detail the quantitative findings. The integrated quantitative and qualitative findings of this study indicate that preservice teachers' motivation related to programming can be improved significantly through educational robotics' influences on (1) intrinsic motivation, (2) career motivation, (3) self-determination, (4) self-efficacy, and (5) motivation to integrate programming into teaching.

Table 4.16. *Integrating Quantitative and Qualitative Findings – Motivation*

Finding	Quantitative Evidence	Qualitative Evidence
Intrinsic motivation improved in preservice teachers	Intrinsic motivation increased from the pre-survey (*M* = 2.23, *SD* = 0.93) to the post-survey (*M* = 3.11, *SD* = 0.96), *t*(17) = 4.26, *p* = .001, Cohen's *d* = 1.00.	Theme 1: Participants perceived that a problem-based robotics curriculum improved their intrinsic motivation toward programming.
Career motivation improved in preservice teachers	Career motivation medians increased between pre-survey career motivation (3) and post-survey career motivation (3.72), (*Z* = -3.58, *p* < .001, *r* = -.6).	Theme 2: Participants agreed that knowing programming as a skill had advantages as a teacher.
Self-determination improved in preservice teachers	Self-determination increased from the pre-survey (*M* = 1.99, *SD* = 0.98) to the post-survey (*M* = 3.39, *SD* = 0.72), *t*(17) = 7.07, *p* < .001, Cohen's *d* = 1.67.	Theme 3: Participants experienced self-determination towards programming in the face of robotics challenges.
Self-efficacy increased in preservice teachers	Self-efficacy increased from the pre-survey (*M* = 2.17, *SD* = 0.82) to post-survey (*M* = 3.47, *SD* = 0.84), *t*(17) = 5.75, *p* < .001, Cohen's *d* = 1.36.	Theme 4: Participants perceived that the gradually increasing level of difficulty in the robotics curriculum improved their self-efficacy about programming from initially low levels.
MTIPIT improved in preservice teachers	MTIPIT increased from the pre-survey (*M* = 2.59, *SD* = 1.04) to the post-survey (*M* = 3.72, *SD* = 0.75), *t*(17) = 6.10, *p* < .001, Cohen's *d* = 1.20.	Theme 5: Participants perceived programming as a viable fit in their future classrooms.

Intrinsic Motivation

Quantitative findings showed that intrinsic motivation increased from the pre-survey (*M* = 2.23, *SD* = 0.93) to the post-survey (*M* = 3.11, *SD* = 0.96), *t*(17) = 4.26, *p* = .001, Cohen's *d* = 1.00. Qualitative findings suggested that participants were intrinsically motivated to complete programming tasks as they solved problems and used concrete robots to represent abstract concepts. These combined findings indicated that educational robotics improve preservice teachers' motivation related to programming by affecting their intrinsic motivation.

Career Motivation

Quantitative findings showed that career motivation medians increased between pre-survey career motivation (3) and post-survey career motivation (3.72), ($Z = -3.58$, $p < .001$, $r = -.60$). Qualitative findings suggested that participants were motivated to complete programming tasks in order to give themselves more advantages or options in job seeking and allow them to improve future teaching. These combined findings indicated that educational robotics improve preservice teachers' motivation related to programming by affecting their career motivation.

Self-Determination

Quantitative findings showed that self-determination increased from the pre-survey ($M = 1.99$, $SD = 0.98$) to the post-survey ($M = 3.39$, $SD = 0.72$), $t(17) = 7.07$, $p < .001$, Cohen’s $d = 1.67$. Qualitative findings suggested that participants were motivated to complete programming tasks as they tried different options to solve problems and used CPS strategies. These combined findings indicated that educational robotics improve preservice teachers' motivation related to programming by affecting their self-determination.

Self-Efficacy

Quantitative findings showed that self-efficacy increased from the pre-survey ($M = 10.83$, $SD = 4.08$) to post-($M = 2.17$, $SD = 0.82$) to post-survey ($M = 3.47$, $SD = 0.84$), $t(17) = 5.75$, $p < .001$, Cohen’s $d = 1.36$. Qualitative findings suggested that participants were motivated to complete programming tasks and were able to improve their initially low programming self-efficacy as a result of the gradually increasing level of difficulty of the programming concepts in the instruction. These combined findings indicated that educational robotics improve preservice teachers' motivation related to programming by affecting their self-efficacy.

MTIPIT

Quantitative findings showed that MTIPIT increased from the pre-survey (M = 2.59, SD = 1.04) to the post-survey (M = 3.72, SD = 0.75), $t(17) = 6.10$, $p < .001$, Cohen's d = 1.20. Qualitative findings suggested that participants were motivated to integrate programming into their instruction to the level that they had devised practical strategies to do so. These combined findings indicated that educational robotics improve preservice teachers' motivation related to programming by affecting their motivation to integrate programming into their teaching.

Chapter Summary

This section reviewed the analysis and findings of this study. This study employed both quantitative and qualitative data. Quantitative data from the Programming Comprehension Assessment and the Programming Motivation Survey were analyzed through paired sample *t*-tests. Findings associated with RQ#1 showed that participants' overall comprehension of programming concepts significantly increased. Further, participants' comprehension of basic procedures, advanced procedures, control structures, and variables significantly increased. Quantitative findings associated with RQ#2 indicated that participants' overall motivation related to programming increased. Further, participants' intrinsic motivation, career motivation, self-determination, self-efficacy, and MTIPIT significantly increased. Qualitative data revealed five themes: (1)

participants perceived that a problem-based robotics curriculum improved their intrinsic motivation toward programming, (2) participants agreed that knowing programming as a skill had advantages as a teacher, (3) participants experienced self-determination towards programming in the face of robotics challenges, (4) participants perceived that the gradually increasing level of difficulty in the robotics curriculum improved their self-efficacy about programming from initially low levels, and (5) participants perceived programming as a viable fit in their future classrooms.

The findings of this study indicate that educational robotics can be used to significantly improve preservice teachers' comprehension of programming concepts related to (1) basic procedures, (2) advanced procedures, (3) control structures, and (4) variables. The integrated quantitative and qualitative findings of this study indicate that preservice teachers' motivation related to programming can be improved significantly through educational robotics' influences on (1) intrinsic motivation, (2) career motivation, (3) self-determination, (4) self-efficacy, and (5) motivation to integrate programming into teaching.

CHAPTER 5

DISCUSSION, IMPLICATIONS, AND LIMITATIONS

The purpose of this action research was to evaluate the effect educational robotics have on programming comprehension and motivation of preservice teachers at a medium-sized liberal arts university in the southeastern United States. Quantitative findings indicated an increase in participants' comprehension of programming concepts as well as an increase in motivation related to programming. Qualitative data revealed five themes: (1) participants perceived that a problem-based robotics curriculum improved their intrinsic motivation toward programming, (2) participants agreed that knowing programming as a skill had advantages as a teacher, (3) participants experienced self-determination towards programming in the face of robotics challenges, (4) participants perceived that the gradually increasing level of difficulty in the robotics curriculum improved their self-efficacy about programming from initially low levels, and (5) participants perceived programming as a viable fit in their future classrooms. Integrated findings of this study suggest that preservice teachers' comprehension of programming concepts and motivation related to programming can be improved through educational robotics. This chapter shares the (a) discussion, (b) implications, and (c) limitations of this action research.

Discussion

The quantitative and qualitative data were combined to directly address the research questions of this study: (1) What is the effect of educational robotics on preservice teachers' comprehension of programming concepts? and (2) How and to what extent does educational robotics influence preservice teachers' motivation related to programming? To look at the big picture and compare this study's results to previous findings in the field, existing literature on programming, educational robotics, preservice, and in-service teachers was used to guide these quantitative and qualitative findings. In this section, comprehension of programming concepts will first be discussed, followed by teachers' motivation related to programming.

Research Question #1: What is the effect of educational robotics on preservice teachers' comprehension of programming concepts?

The findings of this study indicate that educational robotics can be used to significantly improve preservice teachers' comprehension of programming concepts related to (1) basic procedures, (2) advanced procedures, (3) control structures, and (4) variables. Comprehension of programming concepts, synthesized as programming comprehension in this study, is described by Ala-Mutka (2004) as the "ability to track code to build a mental model of the program and predict its behavior" (p. 5). Educational literature has shown that comprehension can be demonstrated in multiple ways, either by comparing, interpreting, describing, or organizing, among others (Bloom et al., 1956). Ramalingam and Wiedenbeck (1997) have explained that programming comprehension includes reading a program with the purpose of doing some further task, which necessitates understanding.

Scores on the Programming Comprehension Assessment suggest that the educational robotics had a positive impact on participants' comprehension of programming concepts. The paired sample *t*-test revealed that participants' overall posttest scores (*M* = .58, *SD* = .24) were significantly higher than pretest scores (*M* = .21, *SD* = .07), *t*(17) = 6.48, *p* < .001, Cohen's *d* = 1.53. Participants entered the study with a low level of programming comprehension. The lowest score on the pretest was a 5%, and the highest was 40%. After the intervention, the participants' scores increased significantly. The lowest score on the posttest was 15%, and the highest was 90%. Not all participants' scores on the Programming Comprehension Assessment improved. Two participants' scores stayed the same, while one participant's score decreased from the pretest to the posttest. Although it is possible that these participants either did not learn anything over the four weeks of the intervention's instructional time or the educational robotics intervention led to a decrease in their comprehension of programming concepts, these low scores might also be attributed to other factors, like assessment apathy (Thompson, 2008). While no participants achieved a perfect score on the Programming Comprehension Assessment, there were five participants who scored 80% or higher on the posttest. Altogether, these findings suggest that preservice teachers' comprehension of programming concepts can be improved through educational robotics.

The nearly unanimous positive results in this study confirm previous studies' findings (Jaipal-Jamani & Angeli, 2017; Sullivan & Moriarty, 2009) on the comprehension of programming concepts. Jaipal-Jamani and Angeli (2017) found that their population of preservice elementary teachers had statistically significant differences in programming knowledge between pre and posttests as the result of an educational robotics intervention. This study's results also confirm research by Sullivan and Moriarty (2009), which indicated that in-service teachers' understanding of programming increased from the no proficiency and low proficiency levels to the moderate and strong proficiency levels after robotics workshops.

The findings of this study indicate that educational robotics can be used to significantly improve preservice teachers' comprehension of programming concepts related to (1) basic procedures, (2) advanced procedures, (3) control structures, and (4) variables. The next sections will present an analysis of the findings related to the comprehension of programming concepts delineated by each unit of the Programming Comprehension Assessment. These findings will then be discussed in relation to existing literature.

Basic procedures. Basic procedures in programming include syntactic programming concepts like the vocabulary, grammar, and format of a programming language (Mayer, 1979) as well as sequencing, which Strawhacker and Bers (2015) defined as "the idea that order matters when giving instructions" in programming (p. 297). The fact that participants' comprehension of basic procedures increased significantly from the pretest (*Mdn.* = .20, SD = *.15*) to posttest (*Mdn.* = .70, *SD* = .29) indicated that there was a statistically significant effect in participants' comprehension of basic procedures concepts ($Z = -3.30$, $p = .001$, $r = -.55$). On the Basic Procedures unit, participants improved from a 19% to a 59% on average. This section will discuss the findings of the Basic Procedures unit and relate them to the existing literature.

The increase in comprehension of basic procedures might be explained best by Ala-Mutka (2004) who suggested that "visualizing the basic programming structures" can be beneficial to for novices in building their comprehension of programming (p. 6). The educational robots' actions allowed participants to visualize basic programming concepts for the participants. According to Pennington’s (1986) framework of programming comprehension, mental representations based on experiences are layered on top of classic language comprehension. Through Pennington’s (1986) framework, novices visualize the programming functions in a more concrete form, adding operational mental models to the programming language through the visualizations. Visualizing programming in concrete form through educational robotics could account for participants’ improvements to their programming comprehension as functional knowledge could have been layered onto state knowledge and operations knowledge.

Despite research by Kim et al. (2018), which noted that “participants omitted commands that were necessary for the robot to perform as planned” (p. 772), the results of this study, particularly in question #3 (Gain = .72), were different. This difference might stem from Kim et al. (2018) using a different block-based programming language that was less intuitive for their participants than the EV3-G programming language used in this study to demonstrate comprehension of the syntactic aspects of programming. Another possibility is that the activities and challenges in this study improved the proficiency of participants in basic programming procedures beyond the level of comprehension of participants in the Kim et al. (2018) study. This study provides additional research to compliment Kim et al.’s (2018) findings and add to the limited literature on preservice teachers’ comprehension of basic programming procedures.

In addition, Kim et al. (2018) found that preservice teachers exhibited difficulty with debugging a block-based programming language while programming robots. Jayathirtha, Fields, and Kafai (2018) have explained that debugging “can reveal significant information about student learning” (p. 1). Kim et al. (2018) noted that debugging was “indeed difficult for preservice teachers” as an overarching finding of their study. In the Basic Procedures unit, the question participants scored the lowest on was a question that assessed participants’ abilities to spot an error in a program. Participants answered question #4 correctly on the posttest only 28% of the time (Gain = .06). While one question specifically addressing debugging in this section might not have extensively assessed participants’ debugging skills, it offered insight into participants’ comprehension scores on this unit and was informed by prior studies utilizing one specific debugging question (Jaipal-Jamani & Angeli, 2017; Lister et al., 2004). These findings parallel those by Kim et al. (2018), who found that preservice teachers struggled with debugging. Kim et al. (2018) theorized that it is difficult for even those who are advanced programmers to debug a program as “it requires mindful, persistent engagement” (p. 769). Similarly, Falloon (2016) noted that debugging was a complicated process because it necessitates perseverance and a systemic approach, which is often discounted by students who adopt random, unsystematic, hasty approaches. There is little research on debugging in block-based programming languages (Kim et al., 2017, 2018); therefore, it is my supposition that participants’ scores might not have improved as much as in other units because they did not adopt disciplined, systematic debugging approaches.

Overall, scores on the Basic Procedures unit indicated that educational robotics had a positive effect on preservice teachers’ comprehension of programming concepts. The Basic Procedures unit had the second-highest increase out of all the units, slightly behind the Advanced Procedures unit. While data show significant gains from the pretest to posttest, participants’ scores on this unit suggest an incomplete understanding of fundamental programming procedures related to debugging. Existing literature (Falloon, 2016; Kim et al., 2017, 2018), in combination with this study's results, suggests that while educational robotics can be used to increase preservice teachers' comprehension of basic procedures in programming, debugging remains a difficult skillset for this population.

Advanced procedures. Advanced procedures are defined by Pea and Kurland (1984) as "higher level executive and metaplanning decisions such as what strategic approach to take to the problem" (p. 160). Advanced procedures combine syntactic and semantic programming knowledge into strategic programming decisions (McGill & Volet, 1997). Participants' comprehension of advanced procedures increased significantly from the pretest (*Mdn.* = .20, *SD* = .18) to posttest (*Mdn.* = .70, *SD* = .30) and indicated that there was a statistically significant effect in participants' comprehension of advanced procedures concepts (*Z* = -3.43, *p* = .001, *r* = -.57). This section will discuss the findings of the Advanced Procedures unit and situate them within the existing literature.

Participants' average posttest scores were the highest on the Advanced Procedures unit. This unit also showed the greatest increase out of all the units from an average of 22% on the pretest to 66% on the posttest. The Advanced Procedures unit showing the greatest increase among all the units – even over Basic Procedures – may be explained by schema theory (Kalyuga, 2010; Sweller, 1994). Previously learned Basic Procedures unit concepts filed as long-term memory may have been updated with conceptually similar, yet new Advanced Procedures unit schema, adding to the participants' programming comprehension. Chunks associated with previous knowledge from the Basic Procedures unit were updated with new schemas as new material was learned, which contributed towards a deeper understanding of those concepts (Sweller, 1994). Because Advanced Procedures concepts built on Basic Procedures concepts, the participants could rely on previous knowledge, which contributed toward a deeper comprehension and a larger increase on the Programming Comprehension Assessment.

Scores on the Advanced Procedures unit indicated that educational robotics had a positive effect on preservice teachers' comprehension of programming concepts. The findings on the Advanced Procedures unit echo those by Kay et al. (2014). In their research, Kay et al. (2014) indicated that their mixed in-service and preservice participants' (*N* = 22) correct answers on the movement programming question of their content knowledge assessment that conceptually aligned to this study's Advanced Procedures unit increased dramatically. In Kay et al.'s (2014) study, participants' scores increased from 40% to 100% after three days of robotics workshops.

The question with the largest average improvement was question #9 (Gain = .61), which participants answered correctly over 66% of the time on the posttest. Question #9 assessed participants' syntactic and semantic comprehension of programming turns. This data might suggest that participants were comfortable with combining syntactic and semantic programming comprehension to solve problems. The mazes utilized in the study's Advanced Procedures unit exercised the skills participants needed to solve question #9. Thus, qualitative findings in Theme 1 – participants perceived that a problem-based robotics curriculum improved their intrinsic motivation toward programming – could provide an explanation of the motivational increase in the Advanced Procedures unit. In the individual interviews, the Maze Challenge from the Advanced Procedures unit of instruction was the most-noted fun and enjoyable curriculum element. My speculation is that because the highly enjoyed Maze Challenge was embedded within the instructional unit that had the highest comprehension improvement (Advanced Procedures) it may indicate that the Maze Challenge motivated participants to learn and contributed toward participants' leap in comprehension within that unit.

A question with the lowest average on the posttest was question #8. This question assessed participants' abilities to pick out the program which included the correct strategic programming to move a robot along a path that includes the hypotenuse of a triangle. This data indicated that participants had a shallow comprehension of strategic programming within the Advanced Procedures unit. One possible reason for the low scores on this question might be that the introduction of the Pythagorean Theorem (i.e., $a^2 + b^2 = c^2$) confused participants. However, deductive reasoning and code tracking could be used to eliminate incorrect answers to this question. Therefore, participants might simply have mis-tracked the program from start to finish. Further data on this question is needed to inform future teaching and assessment.

Participants' posttest scores were significantly higher than their pretest scores on the Advanced Procedures unit. Further, participants' average posttest scores were the highest out of all units. Overall, these collective findings suggest that educational robotics can be used to significantly increase preservice teachers' comprehension of advanced programming procedures.

Control structures. Control structures – also known as conditionals or flow control – include programming concepts such as loops and switches that guide the course of action within a program based on special instructions (Bers et al., 2014). Participants' comprehension of control structures increased from the pretest (*M* = .26, *SD* = .17) to the posttest (*M* = .58, *SD* = .26), *t*(17) = 4.68, *p* < .001, Cohen's *d* = 1.10. While participants' scores increased significantly on the Control Structures unit which indicated that educational robotics had a positive effect on preservice teachers' comprehension of programming concepts, this increase was less pronounced compared to other units. This section will discuss the findings of this study related to the Control Structures unit and connect these findings to existing literature.

Participants' average scores on this unit indicated a significant increase, but they reflected a limited comprehension of control structures in general. Conceptually, the Control Structures unit was designed as the second-most complex topic of the instruction, and the unit's posttest scores were fittingly the second lowest on average (*M* = .58, *SD* = .26). Similarly, studies that used text-based programming languages (Ahmadzadeh et al., 2007; Fitzgerald et al., 2008) as well as block-based programming languages (Chiu & Huang, 2015; Kim et al., 2018) have pointed to participants' most produced errors occurring in control structures concepts. One-third of the interviewees (*n* = 2) commented that the Control Structures concepts were difficult and needed more time dedicated to them in the instruction. This research corroborated Kim et al.'s (2018) findings which indicated that preservice teachers often struggled with "improperly defined conditionals" (p. 772). Therefore, while the increase in this unit was significant, participants exhibited a lower increase than in other units.

This unit's lower increase may be attributed to participants' struggles with multiple loops. Kim et al. (2018) explained that preservice teachers incorrectly designed their programs, "omitting loop or other commands that had to be included to complete the program" (p. 772). This study's findings indicated that preservice teachers had trouble with multiple loops in particular. To explain, the question with the largest improvement in the unit was #15 (Gain = .44), which assessed participants' abilities to modify a single loop in an algorithm to execute a specific route for the robot. My speculation is that participants scored highly on this question due to its relative simplicity in only utilizing one loop. In addition, question #12 had the lowest gain (.17) out of all the questions on the unit, possibly because it had the highest pretest average score out of all the questions on the assessment (.56). This question required participants to correctly simplify a program using a single loop. This data indicated that over half the participants had an initial comprehension of the concept of looping. However, when participants were given multiple loops, they struggled. For example, question #11 addressed multiple loops and had the lowest average score on the posttest in the unit (.39). This question evaluated participants' abilities to trace a program and determine its outcome using multiple loops. Therefore, participants demonstrated competency and comprehension of simplifying programs using one loop but had difficulty tracing the outcome of programs utilizing multiple loops.

Participants' scores increased significantly on the Control Structures unit which indicated that educational robotics had a positive effect on preservice teachers' comprehension of programming concepts. However, this increase was the second lowest of all units. Participants excelled with problems featuring a single loop but struggled with tracing multiple loops in an algorithm. In sum, these findings suggest that educational robotics can be used to significantly increase preservice teachers' comprehension of the control structures; however, this population struggles with depth of comprehension of looping.

Variables. Variables are values in a program that can change based on different instructions and inputs within the program. Participants' comprehension of variables increased from the pretest (*M* = .19, *SD* = .17) to the posttest (*M* = .51, *SD* = .32), *t*(17) = 3.69, *p* = .002, Cohen's *d* = .87. This section will discuss the findings related to the Variables unit and relate these findings to existing literature.

The increase in variable comprehension by participants in this study may be best explained by the visualization and concrete modeling of programming through the actions of the robots. According to Ala-Mutka (2004) recursion, or the use of loops with variables to complete smaller tasks that reiterate to complete a larger task, is a programming concept which can be taught through visualizations "on [a] high level" (p. 8). Mayer's (1981) programming comprehension model which borrowed concepts from the IPM (Newell & Simon, 1972) was used by Bayman and Mayer (1983) to evaluate programming comprehension. As a result of their study, Bayman and Mayer (1983) determined that novices learning programming required more concrete models of programs to understand abstract programming functions.

Variables are often considered difficult to comprehend by novices (Grover & Basu, 2017; Kuittinen & Sajaniemi, 2004), and the Variables unit of instruction was correspondingly the most advanced of the intervention. Therefore, it is fitting that this unit had the lowest pretest (*M* = .19, *SD* = .17) and posttest (*M* = .51, *SD* = .32) scores on average. While scores increased significantly, these data suggest that participants did not have as deep of a comprehension of variables as other programming concepts. This study confirms Kim et al.'s (2018) findings that preservice teachers commonly demonstrate errors in defining values of variables while programming robots and Govender and Grayson's (2008) findings that in-service and preservice teachers find the concept of variables confusing. In their study that utilized block-based programming, Grover and Basu (2017) noted that beginners had difficulty with using "mathematical and logical expressions, naming variables, and assigning suitable data types and structures" (p. 268). Further, variables can be difficult to define by teachers, as Meerbaum-Salant et al. (2013) found. In their study, Meerbaum-Salant et al. (2013) observed that mathematics teachers and computer science teachers had different conceptual understandings of variables. Meerbaum-Salant et al. (2013) attributed the inaccurate mathematics conceptual understanding of variables to the math students' struggles with the concept. Govender and Grayson (2008) found that their mixed group of in-service and preservice teachers learning to program in Java, a text-based programming language, felt that variables were confusing and complicated.

The question with the largest improvement was #17. This question assessed participants' comprehension of variables' syntactic and semantic elements within a switch in a program. While this question required participants to track a program through a switch and then use their syntactic and semantic programming comprehension, participants only had to explain the variable's influence on the program. Explaining falls under the middle analyzing tier of Bloom's Taxonomy, a continuum of ways in which students can demonstrate understanding arranged from simple to complex (Anderson, Krathwohl, & Bloom, 2001). Therefore, this question might have been less difficult to complete than the others in the unit.

The final question, #20, was answered correctly only 22% of the time on the posttest. This question also demonstrated the smallest gain from pretest to posttest (Gain = .17). This question was the most difficult of the assessment as it required participants to understand variables as well as apply all the other programming concepts of the intervention to fill in appropriate values to execute a program. This low increase might be attributed to this question requiring participants to create a program with different values in a fill-in-the-blank format. Creating is the highest tier, and the most complex way students can demonstrate understanding in Bloom's Taxonomy (Anderson et al., 2001). Therefore, this question was fundamentally complex, which might have led to its low increase.

While the average scores did increase significantly for the Variables unit, they did not increase to the extent of the other units. One-third of the interviewees (*n* = 2) mentioned that the concept of variables was difficult for them. Overall, these findings suggest that educational robotics can be used to increase preservice teachers' comprehension of variables but to a lesser extent than other programming concepts due to the difficulty in obtaining a high-level understanding of relevant concepts.

Research Question #2: How and to what extent does educational robotics influence preservice teachers' motivation related to programming?

The integrated quantitative and qualitative findings of this study indicate that preservice teachers' motivation related to programming can be improved significantly through educational robotics' influences on (1) intrinsic motivation, (2) career motivation, (3) self-determination, (4) self-efficacy, and (5) motivation to integrate programming into teaching. Both quantitative and qualitative data were gathered to investigate the research question addressing motivation. Motivation is described by Johns (1996) as the extent to which persistent effort is sustained toward a specific goal. Motivation is an abstract concept that is comprised of many different indicators (Ball, 1977; Jenkins & Davy, 2002; Law et al., 2010).

Participants' overall motivation increased from the pre-survey (M = 2.38, SD = 0.84) to the post-survey (M = 3.48, SD = 0.64), $t(17) = 6.10$, $p < .001$, Cohen's d = 1.44. Participants entered the study with low motivation related to programming. The lowest average Likert scale level of motivation on the pre-survey was 1.32/5, and the highest was 3.92/5 (M = 2.38, SD = 0.84). After the intervention, participants' average motivation levels increased significantly. The lowest average motivation conveyed on the post Programming Motivation Survey was 2.24/5, and the highest was 4.68/5 (M = 3.48, SD = 0.64). However, not all participants' motivation levels on the Programming Motivation Survey improved. While 17 of the 18 participants experienced gains in their motivation, one participant's motivation level decreased from the pre-survey to the post-survey. My speculation is that one participant did not find educational robotics to be motivational. None of the participants' motivation levels remained the same. These 17 of 18 increased levels of agreement on Likert scale statements in the Programming Motivation Survey suggest that the educational robotics positively impacted participants' motivation related to programming.

Quantitative findings explain that educational robotics positively influence preservice teachers' motivation related to programming to statistically significant extents. Qualitative themes explained and reinforced that educational robotics positively influence preservice teachers' motivation related to programming through (1) Intrinsic Motivation,

(2) Career Motivation, (3) Self-Determination, (4) Self-Efficacy, and (5) MTIPIT. These combined findings suggest that preservice teachers' motivation related to programming can be improved through educational robotics. The following paragraphs will discuss participants' motivation related to programming by comparing the qualitative themes with the quantitative survey findings.

Intrinsic motivation. Integrated findings of this study (see Table 4.16) indicated that intrinsic motivation improved in preservice teachers. Intrinsic motivation is one's internal drive to complete tasks (Deci & Ryan, 2000; Taylor, 1916). Enjoyment of and interest in a task link are linked to intrinsic motivation (Deci & Ryan, 2000; Law et al., 2010). Preservice teacher participants in studies by Kucuk and Sisman (2018) and Kim et al., (2015, 2018) emphasized the importance of maintaining their intrinsic motivation throughout the robotics activities. This section will discuss the findings of this study related to the quantitative Intrinsic Motivation subscale and Theme 1 in the qualitative findings – participants perceived that a problem-based robotics curriculum improved their intrinsic motivation toward programming – and relate them to the existing literature.

Participants' intrinsic motivation significantly increased from the pre-survey (*M* = 2.23, *SD* = 0.93) to the post-survey (*M* = 3.11, *SD* = 0.96), *t*(17) = 4.26, *p* = .001, Cohen's *d* = 1.00. Interview data affirmed and explained participants' growth of intrinsic motivation. Theme 1 from the qualitative data indicated that intrinsic motivation appeared to be substantially impacted by the intervention's use of problems in the form of robotics programming activities and challenges. All interviewees (*n* = 6) indicated that the activities and challenges were intrinsically motivational. In particular, the Maze Challenge and Maze Challenge with Variables were reported to be motivating to participants. It can be logically inferred that the activities and challenges using the educational robotics increased participants' total intrinsic motivation.

This study provides results that are consistent with previous research (Kim et al., 2015, 2018; Kucuk & Sisman, 2018) that found that preservice teachers perceived educational robotics to be intrinsically motivating while learning to program. This study's combined findings paralleled those of Kucuk and Sisman (2018), who found that their preservice teacher population considered educational robotics activities and learning by doing to be fun. This study's findings support those of Kim et al. (2015, 2018) and Kucuk and Sisman (2018) while also extending their findings by pinpointing high intrinsic motivation gains by participants in the areas of interest and enjoyment. On the Programming Motivation Survey, intrinsic motivation exhibited the largest average increases in two statements: #3 "Learning programming is interesting" and #19 "I enjoy learning programming" (Gain = 1.27). On the post-survey, participants also had the highest level of agreement with statement #3 within the Intrinsic Motivation subscale (3.78/5). Theme 1 explained that participants experienced increased interest and enjoyment due to the problems they solved. Kopcha et al. (2017) explained that authentic problems afford learners opportunities to solve the problem based on the lessons they learned through real-life scenarios. Different interviewees used words like "fun," "enjoyable," and "interesting" to describe the challenges.

Educational robotics can be used to demonstrate physical representations of abstract concepts, such as equations (Han, 2013). Theme 1 also explained that participants were interested in the representation of abstract concepts in concrete form through the educational robotics curriculum, which boosted their intrinsic motivation levels. This finding is supported by the research of Bayman and Mayer (1983) that investigated Mayer's (1981) model of programming comprehension and suggested that novice programmers should be given concrete models of programs in order to build their mental models. Piaget (1967, 1973) explained that constructivism is the building of abstract knowledge structures in one's mind through concrete experiences. Therefore, participants were intrinsically motivated by constructivist processes of representing abstract concepts in concrete form through educational robotics.

These quantitative and qualitative findings on intrinsic motivation reinforce those by Kim et al. (2015, 2018) and Kucuk and Sisman (2018), which stated that educational robotics are intrinsically motivating for preservice teachers. Further, this study adds to the literature on preservice teachers learning programming through educational robotics by explaining that preservice teachers' intrinsic motivation can be boosted by implementing authentic problem-solving challenges and representing abstract concepts in concrete form.

In summation, quantitative data indicated significant gains in participants' intrinsic motivation in the areas of interest and enjoyment. These results were confirmed and explained by the qualitative data, which indicated that authentic problem-solving through educational robotics activities and challenges, as well as representing abstract math in concrete form, boosted participants' interest and enjoyment. Existing literature (Kim et al., 2015, 2018; Kucuk & Sisman, 2018), in combination with this study's results, suggest that educational robotics can be used to increase preservice teachers' intrinsic motivation related to programming.

Career motivation. Integrated findings of this study (see Table 4.16) indicated that career motivation improved in preservice teachers. Career motivation includes one's beliefs of a topic's career relevance as well as one's effort to enhance their career possibilities (Arwood, 2004; Glynn et al., 2009). While careers are often associated with extrinsic factors such as money, Glynn et al. (2009, 2011) found a close relationship between intrinsic motivation and career motivation in science. This section will discuss the findings of this study related to the quantitative Career Motivation subscale and Theme 2 in the qualitative findings and relate them to the existing literature.

The medians of the pre-survey (*Mdn.* = 3) Career Motivation and post-survey (*Mdn.* = 3.72) Career Motivation increased significantly ($Z = -3.58$, $p < .001$, $r = -.6$). Career Motivation was tied for the highest average agreement level on the post-survey ($M = 3.72$, $SD = 0.59$) with MTIPIT. However, the Career Motivation subscale also exhibited the lowest subscale increase, which could be attributed to participants having high agreement with the statements in this subscale on the pre-survey. Participants' career motivation only increased on average from 2.94 to 3.72 (Gain = .78). Qualitative interview data in Theme 2 – Educational robotics affected participants' career motivation towards programming – supported the quantitative data by describing participants' high levels of career motivation. For example, participants noted that teachers who could teach programming were in "high demand," as Jennifer explained. These data indicated increased career motivation among participants.

Research by Kim et al. (2015) found that preservice teachers who learned programming through educational robotics had a small but meaningful increase in their interest in STEM careers. Although this increase was relatively low, Kim et al. (2015)

categorized this finding as “noteworthy considering that their career goals were already set to become an early childhood educator” (p. 27). The findings of this study run parallel to those of Kim et al. (2015) and are also noteworthy because participants’ highest combined pre and post motivation levels were in the Career Motivation subscale even though none of them were on a path to become computer science teachers. Career motivation increased most dramatically on statement #23, “My career will involve programming” (Gain = 1.22). In Theme 2, many participants voiced their perspectives that schools and the economy were moving toward more technology-rich futures. The large increase for this subscale could be attributed to the intervention’s use of different lectures about new state standards for K-8 computer science as well as videos showcasing how teachers are implementing computer science into their instruction. While high pre-survey career motivation indicated that participants were cognizant of the current and future outlook of the economy before they took part in the intervention, they may not have been informed about the relevance and imminence of computer science standards for the grade level they plan to teach. The statement with the lowest increase in career motivation between pre and post was statement #10 (Gain = .44), “Knowing programming will give me a career advantage.” This lower increase could be attributed to how high participants’ level of agreement was on this statement on both the pre-survey and post-survey. Participants’ pre-survey level of agreement (3.67/5) was the highest initial level of agreement of the subscale. Correspondingly, their post level of agreement (4.11/5) was also the highest level of agreement within the subscale on the post-survey. Again, this small increase is noteworthy, as described by Kim et al. (2015), because participants’ career motivation was already high, and the educational robotics intervention increased that high career motivation even more. These high levels are reflected in Theme 2. Participants stated that learning programming would give them career advantages in the interviews. For example, Simon explained that the ability to walk into a teaching interview with programming as a skill on a resume “goes a long way.” These findings demonstrate that participants exhibited increases in their already high career motivation related to programming.

The qualitative findings from Theme 2 can add to the literature about preservice teachers’ career motivation. Theme 2 offers insights into the reasons preservice teachers experience increased career motivation. Theme 2 presented two categories of preservice teachers’ career motivation: (1) to give themselves more advantages or options in job seeking, and (2) to expand their skillsets for teaching their future students. These categories can be used by preservice teacher educators as they design their curricula to boost career motivation.

The combined quantitative and qualitative findings of this study indicated significant gains in participants’ career motivation. However, because participants initially rated the Career Motivation subscale statements at such a high level, gains were not as large as in other subscales. It is my supposition that because all the participants in this study (*N* = 18) were between the ages of 18 and 23, it is likely that the increasing importance of technology that they have experienced in their own lifetimes has led them to share such high pre-survey career motivation related to programming. The instructional materials showcasing the new state computer science standards and data on jobs in the computer science field further increased this high career motivation related to programming. Existing literature, in combination with this study’s results, suggest that preservice teachers' career motivation related to programming can be increased through educational robotics.

Self-determination. Integrated findings of this study (see Table 4.16) indicated that self-determination improved in preservice teachers. Self-determination is the control learners have over their learning and includes autonomy, competence, and relatedness (Black & Deci, 2000; Cullen & Greene, 2011; Ryan & Deci, 2020). Research by McGill (2012) found that college students struggled to identify the relevance of learning programming using educational robotics to their daily lives. This section will discuss the findings of this study related to the quantitative Self-Determination subscale as well as Theme 3 in the qualitative findings – participants experienced self-determination towards programming in the face of robotics challenges – and relate them to the existing literature.

Participants demonstrated the largest increase to their motivation in the subscale of Self-Determination. Participants' self-determination increased significantly the pre-survey (M = 1.99, SD = 0.98) to the post-survey (M = 3.39, SD = 0.72), $t(17)$ = 7.07, p < .001, Cohen's d = 1.67. Cullen and Greene (2011) noted that "consistent with Self-Determination Theory in that in order to be motivated to achieve a goal" related to technology integration, preservice teachers "must feel competent and able to do the task at hand" (p. 42). Self-determination can be improved through confidence-building (Ryan & Deci, 2000, 2020). Participants' building of competence likely contributed to their large increase in self-determination. All but one participant (n = 17) demonstrated improved comprehension of programming concepts between the pre and post Programming Comprehension Assessment. These increases improved perceptions of competence among participants. The competence of participants may have been most directly impacted by the achievement of completing the different activities and challenges in the intervention. For example, Mariah and Randy stated that by accomplishing the different challenges, they increased their competence and confidence. These combined quantitative and qualitative findings are concordant with those of Cullen and Greene (2011).

Kim et al. (2015) found that preservice teachers put in more effort when they encountered difficulties while programming educational robotics. According to Kim et al. (2015), one of the methods the preservice teachers used to solve problems was “seeking help from peers” by “exchanging ideas, questioning, and answering questions in collaborative small groups” (p. 26). Self-determination increased most dramatically on statement #5, “I put enough effort into learning programming” (Gain = 1.89). This statement also had the highest agreement on the post-survey (4.17/5) among the Self-Determination subscale statements. Qualitative data from Theme 3 indicated that participants used multiple different CPS strategies (Roschelle & Teasley, 1994) when they encountered difficulty. For instance, participants noted in the interviews brainstorming with partners and approaching other groups for help were ways they put extra effort into learning programming. It can be inferred that the CPS strategies described by participants in the qualitative findings reflect participants’ quantitative increase in their satisfaction level with their effort while learning programming. This study’s combined quantitative and qualitative findings of effort and CPS strategies between groups confirm Kim et al.’s (2015) findings that preservice teachers using educational robotics put in more effort to solve problems through collaboration.

Overall, participants displayed the largest increases in this subscale. Participants' large increases in competence and confidence correlated with their large increases in self-determination (Cullen & Greene, 2011). The quantitative data was supported by the qualitative data from Theme 3. Qualitative evidence supported the findings of Kim et al. (2015) that preservice teachers' extra effort while learning programming through robotics occurred through CPS strategies. Existing literature paired with this study's findings indicated that preservice teachers' self-determination related to programming could be increased through educational robotics.

Self-efficacy. Integrated findings of this study (see Table 4.16) indicated that self-efficacy improved in preservice teachers. Self-efficacy is defined as learners' beliefs in their abilities to achieve a learning task (Bandura, 1997; Martin, 2007). Self-efficacy can be improved through learners experiencing success completing similar tasks (Bandura, 1997). Self-efficacy has been found to be low with educators teaching computer science concepts (Grover & Pea, 2013). Contributing factors to teachers' low self-efficacy include anxiousness with learning how to use new technologies in class (Meerbaum-Salant et al., 2013) and using new and unfamiliar teaching materials (Curzon et al., 2009). Self-efficacy can impact teachers' usage of technology in the classroom (Ertmer & Ottenbreit-Leftwich, 2010; Ertmer et al., 2012), and teachers with higher levels of self-efficacy are more committed to teaching (Chen & Yeung, 2015; Gunning & Mensah, 2011). This section will discuss the findings of this study related to the quantitative Self-Efficacy subscale and Theme 4 – participants perceived that the gradually increasing level of difficulty in the robotics curriculum improved their self-efficacy about programming from initially low levels – in the qualitative findings and relate them to the existing literature.

Participants' exhibited the second-largest increase in the subscale of Self-Efficacy. Participants' self-efficacy increased significantly from the pre-survey (M = 2.17, SD = 0.82) to post-survey (M = 3.47, SD = 0.84), $t(17) = 5.75$, $p < .001$, Cohen's d = 1.36. The factors of past experiences, observed experiences, coaching, visualization of future success, and experience of physical and emotional states contribute toward self-efficacy (Bandura, 1997; Martin, 2007). Evidence from Theme 4 – participants perceived that the gradually increasing level of difficulty in the robotics curriculum improved their self-efficacy about programming from initially low levels – supported the participants' increased quantitative self-efficacy.

This study's self-efficacy findings parallel the literature. For example, research by Jaipal-Jamani and Angeli (2017) indicated that educational robotics could improve preservice teachers' self-efficacy pertaining to programming. Further, Kay et al.'s (2014) findings centered on confidence and found that in-service teachers' self-efficacy related to learning and teaching programming improved through the use of educational robotics. Kay et al.'s (2014) findings indicated that 95% of participants were quite or extremely confident in learning to program while 100% were quite or extremely confident with teaching programming after three days of robotics workshop. In this study's Self-Efficacy subscale, participants' agreement levels increased most dramatically on statement #4, "I am confident in learning programming" (Gain = 1.83) on the Programming Motivation Survey. Pre-survey responses to the statements in this subscale were low, which could be attributed to this being all but one of the participants' first experiences with programming. Research by Rogerson and Scott (2010) explained that students often exhibit apprehension and fear related to programming, which in turn can cause negative perceptions of programming. Participants' initial lack of confidence in learning programming could be attributed to what Rogerson and Scott (2010) described as "the nature of programming that gives rise to [negative] feelings" (p. 147). Once participants experienced programming through the educational robotics, their fears were diminished, and their confidence improved. Most qualitative data that demonstrated participants' increased confidence came from their explanations of their improved programming comprehension. As described in Theme 4, participants used words such as "zero" or a "blank slate" to define their initial programming comprehension and self-efficacy. This study echoed findings by Bower et al. (2017) that reported that teacher participants had low levels of self-confidence in teaching computational thinking. However, most participants interviewed in this study stated that their perceptions of their programming comprehension improved. For example, Paula explained that on a scale of "one to 10, I am probably a seven" up from an initial level of zero. The quantitative increases in confidence on the Self-Efficacy subscale are supported by the participants' qualitative remarks about increased competence and confidence.

This study can offer additional insights into factors that foster preservice teachers' self-efficacy related to programming. While Jaipal-Jamani and Angeli (2017) and Kay et al. (2014) noted their participants' increases in self-efficacy, these increases were uncovered through quantitative analyses without attribution of the increases to specific factors. Qualitative evidence from Theme 4 attributed the participants' enhanced self-efficacy to the curriculum's design of gradually increasing the level of difficulty of the concepts in the units. For example, Katy pointed out, “You're really helping conceptually building the foundations of like the other stuff that we learned.” Kuittinen and Sajaniemi (2004) noted that within constructivist teaching, it is “necessary that new knowledge is actively built on the top of existing knowledge” (p. 58). When teaching programming, Kuittinen and Sajaniemi (2004) explained, “It is important that the introduction of a new role is built on the top of existing information and that the distinction between the roles is explained properly,” which builds on the conceptual foundations of previous learning, moving the learner toward more difficult concepts (p. 58). The findings of this study extend the findings of Jaipal-Jamani and Angeli (2017) and Kay et al. (2014) by revealing a factor that can increase preservice teachers’ self-efficacy related to programming. Utilizing a curriculum with a gradually increasing difficulty level when teaching programming has been recommended in a conceptual piece in the literature (Kuittinen & Sajaniemi, 2004), but without study data supporting this teaching strategy. The insight into self-efficacy provided by this study can be used to guide preservice teacher educators as they design curricula to improve their students’ self-efficacy related to programming by slowly and carefully increasing the difficulty of the concepts covered in the instruction.

In sum, quantitative and qualitative data from this study indicated significant gains in participants’ self-efficacy. Participants initially held low levels of self-efficacy related to programming. Participants’ qualitative data indicated that they overcame fear, which boosted their confidence related to programming. This study confirmed findings by Bower et al. (2017), who reported their teacher participants had low levels of self-confidence in teaching computational thinking. In addition, this study’s combined quantitative and qualitative findings reinforced those by Jaipal-Jamani and Angeli (2017) and Kay et al. (2014) and extended the available literature by providing qualitative data which noted that preservice teachers' self-efficacy related to programming could be improved through a curriculum that gradually increases in difficulty. Existing literature paired with this study's findings indicated that preservice teachers' self-efficacy related to programming could be increased through educational robotics.

Motivation to integrate programming into teaching (MTIPIT). Integrated findings of this study (see Table 4.16) indicated that MTIPIT improved in preservice teachers. The MTIPIT subscale analyzed the reasons an individual wanted or did not want to use and teach programming based on intrinsic, extrinsic, altruistic, and contextual factors. MTIPIT was based on teacher motivation, which Han and Yin (2016) explained as including the factors of teachers' inherent interest in teaching, their lifelong commitment to teaching, as well as discouraging factors such as bad experiences with teaching. This section will discuss the findings of this study related to the quantitative MTIPIT subscale and Theme 5 in the qualitative findings – participants perceived programming as a viable fit in their future classrooms – and relate them to the existing literature.

Participants' MTIPIT increased significantly from the pre-survey (M = 2.59, SD = 1.04) to the post-survey (M = 3.72, SD = 0.75), $t(17)$ = 5.09, p < .001, Cohen's d = 1.20. The post-survey MTIPIT subscale average (M = 3.72) was tied for the highest post-survey subscale average with Career Motivation. Sisman and Kucuk (2019) found that the idea that motivated their preservice teacher participants the most while they learned programming was that they could learn to teach their future students programming through educational robotics. Therefore, this study's findings were consistent with those of Sisman and Kucuk (2019) because the MTIPIT subscale was tied for the highest post-survey motivation level of all the subscales. Interview data presented in Theme 5 provided explanations for the high MTIPIT levels and why MTIPIT increased.

MTIPIT increased most dramatically on statements #21 "I enjoy teaching programming to others" and #22 "I can teach programming in my future courses" (Gain = 1.39). The high agreement with these statements could be attributed to participants' experiences with programming the robots. As outlined in Theme 5, most interviewees (*n* = 5) demonstrated an improved intention to integrate programming into teaching, and two-thirds of the interviewees (*n* = 4) had an idea for how they would integrate programming into their instruction. These quantitative and qualitative findings combined indicated gains in participants' MTIPIT. Parallel results have been attained in the literature. For example, Jaipal-Jamani and Angeli (2017) reported that of their preservice teacher participants (*N* = 21), over 85% were motivated to integrate block-based programming and educational robotics into their elementary science classes as a result of a science methods course intervention. Similarly, results from research by Kaya et al. (2015) showed that out of their preservice teacher participants (*N* = 11), 100% were motivated to integrate block-based programming and educational robotics into their instruction.

While this study's findings suggested that participants enjoyed the idea of teaching programming to students and mentioned confidence that they can teach the topic, quantitative and qualitative data indicated that their MTIPIT is tempered by the uncertainty of how they will integrate programming into their curricula. The statement with the lowest increase between pre and post was a tie among statement #18 “I plan to incorporate programming into my teaching” and statement #2 “Teaching programming would benefit my students” (Gain = .94). These results are counterintuitive, given the large increases in the other statements. Low increases in these statements might be attributed to participants feeling that they need more instruction in programming and participants being unsure of programming’s fit with their future subject area. Simon explained that he was planning on teaching English and social studies and was hesitant because he was unsure of the exact curriculum fit for programming. Katy noted her increased perception of the potential of programming in the classroom but felt as though she still needed to learn more about integrating it. Similar perspectives might explain why participants had lower increases in their motivation to incorporate programming into their teaching. These combined quantitative and qualitative findings support Bower et al.’s (2017) findings that according to teachers’ post-workshop survey responses, they characterized themselves as still somewhat hesitant to integrate computer science concepts into their instruction due to perceptions that they did not yet have an adequate level of knowledge, experience, and integration strategies. It should be noted that even though preservice teachers may have positive attitudes toward programming, this does not mean they will implement it in their future teaching. Participants’ perceptions of their future teaching context may impact these results. However, these results do indicate, as Cullen and Greene (2011) explained, that participants “are ready to consider new paradigms of classroom technology integration” (p. 43).

This study offers insights into the extent to which participants can be motivated to integrate programming into their future instruction. Nearly all participants interviewed explained that they wanted to integrate programming into their future teaching. Theme 5 showed that preservice teachers are open to having their perspectives changed from not valuing programming in the classroom to valuing programming in the classroom. For example, Paula explained, “when you first proposed the idea that we would be using programming and stuff in this class I didn't really think that it would be useful at all.” However, this perspective changed because “going through it I [Paula] realized like it is very useful so it's kind of done a complete 180.” As described by Paula, participants’ increased valuation of programming in education, combined with their experiences with educational robotics, improved their intention to integrate programming into their future instruction. Many of the interviewees (*n* = 4) had already devised specific integration strategies. Therefore, preservice teachers’ MTIPIT can be improved through educational robotics from a level of disinterest to where they are motivated and have devised strategies to integrate programming into future instruction.

Quantitative findings showed statistically significant increases in participants’ MTIPIT, and qualitative data affirmed and explained these findings. Existing literature paired with this study’s findings indicated that preservice teachers’ MTIPIT could be increased through educational robotics. This study adds to the literature by explaining the extent to which MTIPIT can be increased in preservice teachers.

Implications

Through action research I was able to gather data through mixed methods. This study has informed my teaching of programming by using the action research to deeply analyze the instructional methods and the design of the curriculum. I was able to review what aspects of the instruction worked with respect to improving the participants’ comprehension of programming concepts and motivation related to programming. The findings of this study are significant for future design and teaching practices to improve preservice teachers' comprehension of programming concepts and motivation toward programming. First, the findings of this study suggest that preservice teachers' comprehension of programming concepts and motivation related to programming can be improved through educational robotics. Second, this study has informed my classroom instruction, including updates to the current curriculum. Third, the findings of this study can be used to offer suggestions for other preservice teacher educators integrating programming concepts into their instruction. The next three sections will describe (1) personal implications, (2) design implications, and (3) recommendations for preservice teacher educators.

Personal Implications

Through the process of this study, I have learned many personal lessons that will help me both as a scholar and an educator in my future practice. While the gains I have made as a scholar and educator are numerous, I will focus on two in this space. These two personal implications include (1) lasting scholarly experiences and (2) unexpected findings.

Lasting scholarly experiences. My work on this dissertation has left me with lasting experiences and knowledge. This dissertation has improved my depth of skill and understanding of quantitative data analysis. Through the guidance of my dissertation chair and personal research, I now feel confident in my abilities to both analyze and interpret quantitative data. Before this dissertation, my comfort zone for interpreting quantitative data was in descriptive statistics. I now understand the differences between parametric and nonparametric results, as well as how these types of quantitative data are analyzed, presented, and interpreted. This improved depth of understanding has personal implications for my future research. With this new understanding, I look forward to adding analytical depth to my future quantitative data analyses.

This dissertation has taken me outside of my scholarly comfort zone with qualitative data analysis. As a teacher, I understood the concept of alignment relative to instruction. Lessons needed to be aligned to state standards and course objectives. Throughout the instrument creation process, I was often frustrated with the countless revisions to the wording of my instruments because each phrase in the instruments needed to be aligned to previous literature and fine-tuned to measure exactly what it was meant to with no overlap between related concepts. Similarly, through the qualitative coding process, I was often frustrated with how precise each code needed to be. I simply had not viewed the world through such a precise and scholarly lens before. I have come to appreciate making instruments and codes as accurate as possible. With my increased awareness of alignment, I now critically examine studies through a scholarly lens. This increased awareness has personal implications for my future research. I look forward to using what I have learned through this dissertation process to incorporate high levels of alignment within my future research. Through my dissertation chair, I feel my capabilities in qualitative analysis have improved. Previously, my qualitative data analysis focus was on “quantitizing the qualitative,” as Saldaña (2016, p. 25) described. My frame of reference for qualitative research was more defined by categories than by themes. I focused on what each participant said in regard to each question and focused on creating categories specific to each question instead of looking for commonalities outside of that immediate prompt. I have gained valuable experience with a qualitative coding tool. The coding tool used in this study, Delve, was an efficient way to assign open codes and look at the bigger picture. Delve helped to organize the open codes while keeping them tethered to their excerpts from the field notes and interview transcripts. This aspect of Delve proved helpful for reviewing excerpts while I moved through the qualitative coding steps. Through the qualitative coding process outlined in this dissertation, I have a deeper view of qualitative analysis. Now, I have the ability to take a deeper view of qualitative data and a broader view of qualitative codes in order to elicit comprehensive themes. I can connect different ideas through themes which span multiple categories. This deeper view of qualitative data analysis has personal implications for my future research. I look forward to using what I have learned to take a deeper look at the big picture within my future research.

Unexpected finding. Novelty effect refers to artificially positive results that are linked to the newness of a treatment and the curiosity of the participants (Hanus & Fox, 2015). The end of the novelty effect can be detected when a steep decline in engagement has occurred (Hamari et al., 2014). My personal observations of participants' behavioral engagement indicated that several participants had outwardly lost interest in the programming instruction. By the final week of the study, five participants seemed disengaged in programming and robotics. This indicated that the novelty of the intervention had worn off. The motivation survey and individual interview data were surprising because they demonstrated that while outwardly participants were ready to move on to new topics in the class, they had almost unanimously grown to value programming as a competency and were eager to integrate programming into their instruction. While I had expected a slight increase in motivation related to programming because of the educational robotics factor, the results were higher and deviated far less than I expected. My observations as the instructor indicated that the novelty effect had worn off, so I expected lower results, but the motivation data indicated that the instruction made a genuine and lasting impact on participants' value and perception of programming. This unexpected finding reinforces the importance of using mixed methods to overcome the biases of one type of data alone (Creswell & Plano Clark, 2018; Mertens, 2009).

Curriculum Design Implications

This research evaluated what effects educational robotics have on preservice teachers' comprehension of programming concepts as well as how and to what extent educational robotics influence preservice teachers' motivation related to programming. Results indicated that participants experienced increases in all programming concepts and motivation indicators evaluated. Select data, classified under the pattern codes of *Difficult* and *Updates to Instruction*, can be used to inform areas of emphasis and updates to the curriculum design (Mertler, 2017). Considering these data, areas of emphasis and updates include (1) duration and scope, (2) design of units, and (3) focus on wider curricular connections.

Duration and scope. The data from this study can help inform updates to the scope of the curriculum. While the curriculum's designed scope of instruction was largely effective, it was broad. Instruction could be updated to include more than the 10 hours of instructional time used in this study. While results indicated that 10 hours of instructional time, activities, and challenges are enough to significantly increase preservice teachers' comprehension of programming concepts, more instructional time has the potential to increase students' depth of comprehension of programming concepts even more. Other studies have employed a greater number of contact hours ranging from 12 (Kim et al., 2018; Sullivan & Moriarty, 2009) to 52 hours (Kucuk & Sisman, 2018). While 12 contact hours are possible in the context of the class in which this curriculum was taught, 52 hours are not. Therefore, 12 contact hours will be implemented in the updated curriculum. Data from the interview transcripts noted that participants wanted either more time to be dedicated to the more difficult concepts in the curriculum, or a longer overall instructional experience. For example, Simon summarized, "I would make it longer…maybe six weeks" as opposed to the four weeks of instructional time in the intervention, "that way, you can go slow." Katy noted, "maybe emphasize more like on the last part of the programming, like maybe have like an extra lesson or two about the looping." These suggestions could be incorporated in a few different ways. For example, when covering control structures, multiple class periods can provide more depth to the instruction on loops and switches. Participants' scores and interview responses noted that they had difficulty with control structures while previous research has indicated that students are likely to make errors in control structures when writing programs (Ahmadzadeh et al., 2007; Chiu & Huang, 2015; Fitzgerald et al., 2008; Kim et al., 2018). This update will allow for more time for practical experiences.

In addition, the concept of variables gave students difficulty. The concept of variables is noted in the literature to be difficult to comprehend by novices (Grover & Basu, 2017; Meerbaum-Salant et al., 2013). Variables are not concepts directly covered in South Carolina's K-8 computer science standards. The concept of variables was included to present a natural integration link for those who are preparing to teach middle school math and to provide participants with more depth of knowledge of programming. However, the historic student makeup in the course in which this instruction occurs is heavily skewed to elementary level preservice teachers. Therefore, this unit of instruction can be removed to limit the scope to more pertinent and applicable topics for all students. An update to the curriculum can reign in the scope to focus on basic and advanced procedures, as well as the control structures of switches and loops. The instructional time dedicated to the Variable unit can be used to provide further depth and meaningful learning experiences for the other units. These topics will provide students with a comprehensive programming background while not overwhelming them with the large scope of programming concepts outside of what they would likely be required to implement.

Design of units. The findings of this study can help inform updates to the design of educational robotics curricula. The design of this study included the units of Basic Procedures, Advanced Procedures, Control Structures, and Variables. It can be inferred that the design of these units largely contributed toward participants experiencing increased comprehension of programming concepts, as well as increased motivation related to programming, as demonstrated in the quantitative data, and verified by the qualitative data. However, these data also offered areas for improvement in the design of the curricula in the areas of comprehension and motivation. Areas of emphasis, as well as updates for the design of educational robotics curricula based on this study's quantitative and qualitative data, will be presented by the instructional unit below.

Basic procedures. Participants' scores on the Basic Procedures unit of the Programming Comprehension Assessment indicated substantial increases in the comprehension of basic syntactic and semantic programming concepts taught as part of the Basic Procedures unit. Participants' interview responses indicated that the concepts in the Basic Procedures unit were valuable and helped them understand more difficult programming concepts later in the curriculum. Therefore, an emphasis on meaningful lectures that explain the programming language and basic programming concepts is important for subsequent iterations of this instruction. Based on the findings of this study, the activities and challenges outlined in Appendix A were indicated to help participants learn basic programming concepts while being motivational. Therefore, these activities and challenges will remain unchanged. As found in this study, participants struggled with debugging in the Basic Procedures unit. These findings of preservice teachers' struggles with debugging are found in the literature as well (Kim et al., 2018). Therefore, debugging exercises should be prioritized within the instruction relative to syntactic and semantic concepts. The curriculum primarily taught debugging through examples in lectures. However, it did not include a practical application of debugging wherein participants needed to debug a program to perform a specific task. Based on participants' data, more practical debugging experiences will be incorporated into future educational robotics curricula. Carefully designed debugging activities and challenges to improve students' comprehension of this topic will be added. Updates to the curriculum will include an added emphasis on the foundational programming concepts as well as additional practical applications of debugging.

Advanced procedures. Participants had the highest increases as well as average posttest scores on the Advanced Procedures unit. Therefore, the curriculum design presented in Appendix A is well designed and necessitate few updates. In particular, the activities and challenges employed in this unit were characterized by participants in the interviews to be helpful for exercising their problem-solving skills as well as substantially motivational. While programming includes inherent math concepts (Barr & Stephenson, 2011; Garcia, Havey, & Barnes, 2015) that were taught as a part of this unit's design, participants struggled when applying math theorems within the problem-solving process in the Programming Comprehension Assessment. Therefore, the updated curriculum will include more direct practice related to math in programming problem-solving. This update will provide an increased depth of understanding to the unit already noted by participants to be both informative and motivational.

Control structures. Participants exhibited moderate increases within the Control Structures unit. Qualitative data revealed that participants enjoyed using the color sensor in combination with switches to write programs that announced the color that the color sensor was looking at. Therefore, future iterations of this instruction will emphasize the use of the color sensor in combination with switches. Participants excelled at modifying a loop within an algorithm to execute a more efficient program, but they struggled with tracing a program that utilized multiple loops. Similar results have been found by other researchers (Ahmadzadeh et al., 2007; Chiu & Huang, 2015; Fitzgerald et al., 2008; Kim et al., 2018), noting an area for emphasis. While an increased emphasis on control structures concepts and increased instructional time could improve this curriculum as outlined in the section addressing the scope above, there are two more additions that can be made for future teaching of this curriculum. First, an application activity will be added to this unit in which students follow the flow of a program that utilizes multiple loops. Second, the concept of looping can be taught mathematically first, and then demonstrated through educational robotics. Through this progression, constructivist teaching (Harel & Papert, 1991; Piaget, 1967) with educational robotics tools can be used to help students to take abstract math ideas and make them concrete through educational experiences. These two strategies will be added to the curriculum detailed in Appendix A for future instruction when teaching control structure concepts.

Variables. Participants exhibited moderate increases within the Variables unit. However, qualitative data indicated that participants felt the concepts in the unit were difficult to understand. Quantitative data showed that while participants were comfortable with basic identification and application of variables, they did not exhibit a deep comprehension of using variables in combination with the other programming concepts covered in the curriculum. Similar findings of novices struggling with variables are noted in the literature (Grover & Basu, 2017; Meerbaum-Salant et al., 2013; Wang et al., 2009). Therefore, in addition to the option of cutting the Variables unit in the section addressing the scope of the curriculum above, an alternative path could include more learning activities focused on applying variables in complex problem-solving scenarios that overlap with concepts learned in previous units.

Focus on wider curricular connections. Interview data revealed an imbalance of integration ideas between math and all other subjects. Preservice teachers will likely be expected to integrate programming into each of the core subject areas (Google Inc. & Gallup Inc., 2016). However, interviewees presented as many integration ideas for the subjects of English, social studies, and science combined (4) as they did math (4). The interviewees who were planning to teach English and social studies were unsure, describing how they would integrate programming into their future instruction. “So yeah, honestly in history I'm not sure like I said if I was teaching math, it would make perfect sense. In history, I don't know to be honest,” replied Jennifer. Randy explained, “I have to educate myself more about some cool ideas that you can put in history and also in English, too. I just have to dig into it more and figure out what would be the best for my students.” Mariah explained that she would use programming for digital storytelling without using educational robotics. She stated that she envisioned herself “incorporating it into a classroom with like story ideas or even the online like storyboard kind of things,” in reference to a video that participants watched on digital storytelling. Because participants’ integration ideas were largely skewed toward math integration, improvements can be made to the curriculum. The curriculum can be updated to showcase more integration videos and ideas for English, social studies, and science. For example, a study by Burke (2012) used programming as a new literacy with which middle school students could tell stories. Specific lesson plans for these subjects can also be presented. These updates can foster preservice teachers’ integration ideas for their future classrooms.

Implications for Preservice Teacher Educators

The general implication of this study is that educational robotics can be used to positively impact preservice teachers’ comprehension and motivation related to programming. Therefore, it is not only recommended that preservice teacher educators teaching programming use educational robotics to teach programming, but that they use the curriculum outlined in Appendix A in addition to the updates outlined in the Curriculum Design Implications section.

If preservice teachers elect to build their own educational robotics curriculum for teaching programming, select findings in this study can be used to inform their instruction and curriculum design while teaching programming concepts in the classroom. Suggestions for preservice teacher educators aiming to increase their students' comprehension of programming concepts and motivation related to programming will be presented in the sections below: (1) carefully sequence concepts, (2) use authentic problem-solving activities and challenges, and (3) offer collaborative problem-solving opportunities.

Carefully sequence concepts. The findings of this study can help inform subsequent preservice teacher educators' educational robotics curricula in terms of the sequence. This study's purposeful sequencing was largely effective. There were some aspects of the unit sequencing in this study that preservice teacher educators could follow in their original curricula. When designing programming curricula, preservice teacher educators should gradually increase the difficulty of programming concepts within their units but do the reverse when teaching each programming concept within the unit. To explain, curricula should begin with the basic concepts that participants in this study pointed to as being greatly valuable. The programming concepts at the start of curricula should focus on foundational syntactic and semantic concepts that can be utilized and built upon in later units (Bucks, 2010; Mayer, 1979; Soloway & Ehrlich, 1984). Preservice teachers should then be afforded time in curricula to apply these programming concepts through activities and challenges which test their problem-solving skills.

Strategic programming concepts should next be introduced to students (McGill & Volet, 1997). Participants in this study noted that the different strategic programming concepts in the Advanced Procedures unit also helped their understanding in later units. From this point, curricula can gradually present more difficult programming concepts that provide more depth for students' comprehension. The sequencing of these more difficult programming concepts would depend on the topics being taught, as well as the state standards and instructional goals of the course.

Use authentic problem-solving activities and challenges. Authentic problems have been proposed as a method with which to increase students' motivation (Parsons & Ward, 2011; Willems & Gonzalez-DeHass, 2012). The problem-solving skills students develop when solving authentic problems are aligned with the skills they will need in the professional world (Belland, 2013; Jonassen, 2011). Interview data from this study revealed that participants were motivated by the authentic problems posed to them in the activities and challenges. Preservice teacher educators designing their programming curricula can utilize educational robotics and authentic problems in much the same way as this study. Unique mazes can be used to scale the difficulty of the problems that students are given at each stage of the instruction, either up or down. By using authentic problems, like mazes, preservice teacher educators can increase the motivation of their students.

Offer collaborative problem-solving opportunities. This study used the learning support of partners. In the study, participants worked in pairs through different programming activities and challenges in order to provide immediate scaffolds for learning and frustration control. Participants' interview responses indicated that they often relied on their partner to help them through the programming process. Other researchers (Eguchi, 2007; Jaipal-Jamani & Angeli, 2017) noted similar positive results from paired groupings. Preservice teacher educators teaching programming through educational robotics can implement this same strategy. It is not, however, suggested to increase the groupings from pairs to any larger number. For example, research by Kucuk and Sisman (2018) and Sisman and Kucuk (2019) reported that preservice teachers working in groups of three or four experienced issues with communication and roles.

While the partner dynamic was indicated to aid participants in their comprehension, it did make ensuring equal programming time with the robot difficult. Preservice teacher educators dividing their students into groupings beyond pairs may further water down the hands-on programming experience time for students, negatively affecting comprehension.

Participants' immediate partner was the support that interview data indicated they most often turned to; however, this was not the only learning support that participants explained helped them. A collaborative classroom environment also was stated to have aided participants as they worked. This learning environment occurred naturally and was not by design within the curriculum. Collaborative classroom environments where separate groups collaborated have been noted to help students learn to program (Casler-Failing, 2017; Eguchi, 2013). If a participant had a question, and their partner could not help them, other groups in the classroom were noted to help the learner through the programming concept. Preservice teacher educators could build upon the phenomenon by encouraging group to group collaboration through social constructivist theory (Vygotsky, 1980), which emphasizes the collaborations between students. For example, preservice teacher educators may create special challenges for each instructional unit where multiple groups must work together to program their robots to interact to achieve a specific task. Then, groups working in collaboration could share ideas and help each other, further promoting group to group collaboration.

Implications for Future Research

The findings of this study offer implications for future research. This study can be used as the beginning of a progression of studies for researchers to evaluate the impact of educational robotics as a tool for teaching programming. These potential research topics can be divided into four categories (1) updated curriculum, (2) factor analysis, (3) programming unplugged, and (4) experimental studies.

Updated curriculum. In alignment with action research (Creswell, 2014; Mertler, 2017), this study's curriculum could be improved and tested. In the sections above, proposed updates to the curriculum in this study, as well as recommendations for preservice teacher educators, were detailed. In a follow-up to this study, future research could enact these updates and recommendations to evaluate the updated curriculum's impact on preservice teachers' comprehension and motivation related to programming. For example, cycle two of this action research could focus more on basic and advanced procedures in addition to control structures over 12 contact hours and analyze the result. From those results, further follow-up studies could be crafted in a cyclical process.

Factor analysis. The Programming Motivation Survey instrument utilized in this study indicated the potential for further refinement and validation. The Programming Motivation Survey was tested twice for reliability (N = 18), once on the pre-survey, and once on the post-survey. Very good reliability (DeVellis, 2003) was indicated on the

Cronbach's alpha for both this instrument's pre-survey (α = .96) and post-survey (α = .94). In addition, each of the instrument' subscales indicated very good reliability on both their pre-survey and post-survey Cronbach's alpha testing. The SMQ-II (Glynn et al., 2011), which I adapted and customized to create the Programming Motivation survey, was studied, revised, and validated with a factor analysis (Marsh, Balla, & McDonald, 1988) over the course of two studies (Glynn et al., 2009, 2011). Future research could validate the Programming Motivation Survey in much the same way by utilizing hundreds of participants through a multi-location sample of preservice teachers. This future research would gauge the construct validity of the Programming Motivation Survey, adjust its statements, and present a valid and reliable instrument for evaluating preservice teachers' motivation related to programming.

Schema and long-term memory. An investigation into the lasting effects of this study's intervention is another intriguing research topic. The findings of this study indicated that educational robotics could be used to increase preservice teachers' comprehension of programming concepts. Researchers (Atkinson & Shiffrin, 1968; Baddeley, 1992; Kalyuga, 2010) have explained theories of how learners store knowledge through schema and long-term memory. Further research could check to what extent the knowledge and skills developed by participants in this study return when called upon in long-term memory after an extended period. In this way, the interaction of the senses while learning to program (i.e., tangible educational robotics) could be evaluated through the lens of information processing models, such as the IPM (Newell & Simon, 1972) and Multi Store Model of Memory (Atkinson & Shiffrin, 1968). Such research could provide deeper insights into the processes through which programming is learned.

Programming unplugged. The results of this study may have important implications for unplugged programming activities. Unplugged activities, described by Bower et al. (2017), are programming activities that use "paper or other tactile modelling" such as blocks "to demonstrate the area of computational thinking" (p. 57). Some institutions may not have the resources necessary to teach programming through educational robotics. Unplugged activities are a low-cost way to teach programming. Furthermore, unplugged activities have been shown to increase the understanding of programming concepts among elementary students (Curzon et al., 2009; Lambert & Guiffre, 2009) middle school students (Meerbaum-Salant et al., 2013), high school students (Weintrop, 2016), and in-service teachers (Bower et al., 2017). Therefore, merging the insights about comprehension and motivation uncovered in this study – like the use of authentic problems and factors that increased career motivation – with unplugged activities represents a new area of investigation with a wide range of implications for education given the lack of required equipment.

Experimental studies. This study sets the stage for experimental research. Future research could evaluate educational robotics as a tool for teaching programming against non-tangible alternatives. Two future research ideas are outlined below.

Visual programming environments versus educational robotics. Future research could add to the literature available on the differences between learning programming through tangible and non-tangible modalities. For example, Weintrop (2016) and Weintrop and Wilensky (2017) examined the modality through which students learn programming between text-based, block-based, and hybrid text and block-based programming environments. Future research could continue this line of inquiry and examine the differences between students' learning experiences in visual programming environments, like Scratch, and students' learning experiences programming educational robotics. This research could investigate differences between control (visual programming environment) and experimental (educational robotics) groups' comprehension of programming concepts and motivation related to programming. This potential future research presents a logical next step in evaluating modalities of learning programming.

Educational robotics versus educational robotics simulators. Educational robotics simulators such as CoderZ, Robot Virtual Worlds, or Virtual Robotics Toolkit offer lower-cost alternatives to schools for teaching programming through robotics (Major et al., 2014; McNally et al., 2006). Future research could add to the inquiry into the differences between different modalities of programming started by Weintrop (2016) and Weintrop and Wilensky (2017). Future research could investigate the differences between students' learning experiences in educational robotics simulators versus using educational robotics in the real world. This research could investigate differences between control (educational robotics simulator) and experimental (educational robotics) groups' comprehension of programming concepts and motivation related to programming. This potential future research also presents a logical next step in evaluating modalities of learning programming.

Limitations

While this study suggests insights into the impact of educational robotics on preservice teachers' comprehension of programming concepts and motivation related to programming, there are several limitations of this study. These limitations present areas for further research. The following limitations will be outlined as they align to (1) methodology, (2) context, (3) participants, and (4) the researcher.

Methodology

One limitation of this study is its action research roots. Action research is a systematic process of inquiry that uses a cycle of planning, action, and reflection (Mertler, 2017). Because action research employs a highly contextualized problem, the solutions to that problem are highly contextualized, too. Therefore, the specificity of action research's results to a particular "wicked problem" (Kochhar-Bryant, 2017, p. 12) are limiting. Further, as Mertler (2017) explained, "action research is not conclusive; the results of action research are neither right nor wrong but rather tentative solutions that are based on observations and data collection" (p. 18). These inherent characteristics of action research limit this study's implications.

In addition, the lack of control and experimental groups in the design of this study does limit its generalizability. While action research and experimental design are not mutually exclusive (Mertler, 2017), the equitable nature of action research paired with the ethical notion that all participants must receive the same benefits (Creswell, 2014) does limit the research design in this context. This study did not test any predetermined hypotheses, nor did I exert the detailed control necessary to definitively generalize results based on the different variables. Further inquiry into this topic should utilize true experimental design to definitively analyze the relationship between the variables.

The muffled responses of one of the interviewees is also a limitation of this study. In two different sections of the interview, Simon provided muffled responses that could not be interpreted by the Microsoft Dictate live transcribing tool or by me when reviewing the backup audio recording. While Simon's response in one section was clarified by me in clear audio, the original wording of the interviewee was lost. The words and meanings of the interviewee in the second instance could not be interpreted and were not clarified in the recording.

Context

Equity of hands-on time with the technology in this study's intervention provides an additional limitation. In the intervention, participants worked in pairs. While this study utilized a constructivist framework that valued learners working collaboratively, the sharing of the laptops and the robots between partners could not be totally ensured. While participants were encouraged to share the programming responsibilities and were prompted with multiple reminders to switch program writing duties from one partner to another during the class periods, the onus was on the participants to manage this. Therefore, participants who had less self-efficacy or self-determination could relinquish responsibility to their partner and lose valuable programming experience through difficult problems. Future studies should employ constraints that ensure each partner is given equal programming time or utilize an individual participant design.

The novelty effect is a limitation for a short-term intervention, such as the one in this study (Hamari et al., 2014; Hanus & Fox, 2015; Tsay, Kofinas, Trivedi, & Yang, 2018). The novelty effect is especially relevant when new technologies are introduced to participants due to participants' propensity to engage more deeply with and view the technologies more favorably when they are new to them (Hamari et al., 2014; Hanus & Fox, 2015; Tsay et al., 2018). Future studies should implement longer-term interventions in order to analyze the novelty effect of educational robotics on preservice teachers with longitudinal data.

Limitations exist to the survey used in this study. This study followed a literature review and Glynn et al.'s (2011) valid and reliable SMQ-II survey. However, there were not enough participants in the class with which to complete a rigorous factor analysis to testify the Programming Motivation Survey's empirical validity.

Mixed methods involve qualitative interpretations (Creswell, 2014; Mertler, 2017). The act of interpretation by the researcher is an inherently subjective process (Aron, 1992). My interpretations of the data are the result of viewing the data through a personal lens. This lens is intrinsically linked to my background, experiences, knowledge, and beliefs. Therefore, it is possible that different researchers with different lived experiences may come to different conclusions based on their personal lenses when analyzing the data. While checks on my subjectivities – like member checking, triangulation, and peer debriefing – did occur throughout the course of this study, such limitations do still apply**.**

Participants

Another limitation of this study's highly specific context is the population. Mertler (2017) explained that action research is done by educators to better understand their own teaching practice, focusing "specifically on the unique characteristics of the population with whom a practice is employed" (p. 4). Due to the action research nature of this study, the sample was limited in size by the course cap of the class section taught by me. This population is small in sample size and largely homogenous. Of the final participants, 15 of the 18 were female, and half were elementary education majors. It is possible that if this intervention were implemented in a different class with a different makeup of education majors or a different number of participants that the data would be different. Therefore, the results of this study cannot accurately be generalized to the larger population. Further research into the impact of educational robotics on preservice teachers should include a much larger sample size with a more diverse population of education majors. Multiple research sites and random sampling may be used in order to improve both the sample size and diversity of the participants.

Researcher

The design of the instruction in this study was developed by me. Although this instruction was evaluated by experts, there is still room for improvement. Through this action research, I aim to make data-driven decisions to augment the current instruction for the future.

A final limitation involves the reflexivity of the researcher. As I acted as both the researcher and the instructor in this study, this may have unintentionally influenced its results. Participants were instructed to answer the survey and interview questions honestly and not solely in a way they thought their instructor would want. However, there is no way to know the inner psyche and motivations of participants during those data collection periods. Furthermore, as I acted as both the instructor and researcher, I may have missed important interactions and phenomena that occurred in the classroom while I was teaching or helping other participants. Such limitations can be removed from future studies by employing independent instructors and researchers.

REFERENCES

Abelson, H., & DiSessa. A. (1986). *Turtle geometry*. Cambridge, MA: The MIT Press.

Adams, A. E., Miller, B. G., Saul, M., & Pegg, J. (2014). Supporting elementary pre-service teachers to teach STEM through place-based teaching and learning experiences. *Electronic Journal of Science Education, 18*(5), 1-22.

Adams, D. B. (2010). Explore-create-present: A project series for CS. *Proceedings of the ASEE North Central Sectional Conference (ASEE'10)*.

Ahmadzadeh, M., Elliman, D., & Higgins, C. (2007). The impact of improving debugging skill on programming ability. *Innovation in Teaching and Learning in Information and Computer Sciences, 6*(4), 72–87. https ://doi.org/10.11120 /ital.2007.06040 072.

Ajzen, I. (2005). *Attitudes, personality, and behavior*. New York: Open University Press.

Ala-Mutka, K. (2004). Problems in learning and teaching programming. *Codewitz Needs Analysis*, 1–13. Retrieved from http://www.cs.tut.fi/~edge/literature_study.pdf

Alimisis, D. (2013). Educational robotics: Open questions and new challenges. *Themes in Science & Technology Education*, *6*(1), 63–71. https://doi.org/10.1109/FIE.2014.7044055

Alimisis, D., Moro, M., Arlegui, J., Pina, A., Stassini, F., & Papanikolaou, K. (2007). Robotics & constructivism in education: The TERECoP project. *EuroLogo*, 1–11. Retrieved from http://users.sch.gr/adamopou/docs/syn_eurologo2007_alimisis.pdf

Alkaria, A., & Alhassan, R. (2017). The effect of in-service training of computer science teachers on Scratch programming language skills using an electronic learning platform on programming skills and the attitudes towards teaching programming. *Journal of Education and Training Studies*, *5*(11). https://doi.org/10.11114/jets.v5i11.2608

Almalki, S. (2016). Integrating quantitative and qualitative data in mixed methods research—challenges and benefits. *Journal of Education and Learning*, *5*(3), 288. https://doi.org/10.5539/jel.v5n3p288

Altin, H., & Pedaste, M. (2013). Learning approaches to applying robotics in science education. *Journal of Baltic Science Education, 12*(3), 365-377.

Amabile, T. M., Hill, K. G., Hennessey, B. A., & Tighe, E. M. (1994). The work preference inventory: Assessing intrinsic and extrinsic motivational orientations. *Journal of Personality and Social Psychology*, *66*(5).

Anderson, L. W., Krathwohl, D. R., & Bloom, B. S. (2001). *A taxonomy for learning, teaching, and assessing: A revision of Bloom's Taxonomy of Educational Objectives*. New York: Longman.

Anderson, E. F., & McLoughlin, L. (2007). Critters in the classroom: A 3D computer-game-like tool for teaching programming to computer animation students. *International Conference on Computer Graphics and Interactive Techniques: ACM SIGGRAPH 2007 Educators Program*, 7.

Antle, A. 2007. Designing tangibles for children: What designers need to know. in *CHI EA '07 extended abstracts on human factors in computing systems*. San Jose, CA, USA: ACM.

Apiola, M., Lattu, M., & Pasanen, T. A. (2010). Creativity and intrinsic motivation in computer science education. In C. Laxer (Ed.), *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education* (pp. 199–203). https://doi.org/10.1145/1822090.1822147

Appleton, K. (2003). How do beginning primary school teachers cope with science? Toward an understanding of science teaching practice. *Research Science Education, 33*(1), 1–25.

Arlegui, J., Pina, A., & Moro, M. (2013). A PBL approach using virtual and real robots (with BYOB and LEGO NXT) to teaching learning key competences and standard curricula in primary level. *Proceedings of the First International Conference on Technological Ecosystem for Enhancing Multiculturality - TEEM '13*, 323–328. http://doi.org/10.1145/2536536.2536585

Aron, L. (1992). Interpretation as expression of the analyst's subjectivity. Psychoanalytic Dialogues, 2(4), 475-507. https://doi.org/10.1080/10481889209538947

Arwood, L. (2004). Teaching cell biology to nonscience majors through forensics, or how to design a killer course. *Cell Biology Education, 3*, 131–138.

Atkinson, R. C. & Shiffrin, R. M. (1968). Human memory. A proposed system and its control processes. In K. Spence & J. Spence (Eds.), *The psychology of learning and motivation* (Vol 2). New York, NY: Academic Press.

Babaei, M., & Abednia, A. (2016). Reflective teaching and self-efficacy beliefs: Exploring relationships in the context of teaching EFL in Iran. *Australian Journal of Teacher Education, 41*(9), 1–27. https://doi.org/10.14221/ajte.2016v41n9.1

Baddeley, A. (1992). Working memory. *Science, 255*, 556–559.

Bakke, C. (2013). *Perceptions of professional and educational skills learning opportunities made available through k-12 robotics programming* (Doctoral dissertation). Retrieved from ProQuest Dissertation & Theses. (AAT 3556716)

Ball, S. (1977). *Motivation in education*. Cambridge, MA: Academic Press.

Bandura, A. (1997). Self-efficacy. *Harvard Mental Health Letter, 13*(9), 4–5.

Barker, B. S., Nugent, G., & Grandgenett, N. (2014). Examining fidelity of program implementation in a STEM-oriented out-of-school setting. *International Journal of Technology & Design Education, 24*(1), 39-52. http://doi.org/10.1007/s10798-013-9245-9.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads 2*(1), 48-54.

Bayman, P. & Mayer, R. E. (1983). A diagnosis of beginning programmers' misconceptions of BASIC programming statements. *Communications of the ACM 26*(9), 677-679.

Bazeley, P. (2013). *Qualitative data analysis*. Thousand Oaks: Sage

Belland, B. R. (2013). Mindtools for argumentation, and their role in promoting ill-structured problem solving. In J. M. Spector, B. B. Lockee, S. E. Smaldino, & M. Herring (Eds.), *Learning, problem solving, and mind tools: Essays in honor of David H. Jonassen* (pp. 229–246). New York, NY: Routledge.

Belland, B. R., Kim, C. M., & Hannafin, M. J. (2013). A framework for designing scaffolds that improve motivation and cognition. *Educational Psychologist, 48*(4), 243–270. https://doi.org/10.1080/00461520.2013.838920

Bender, E., Schaper, N., Caspersen, M. E., Margaritis, M., & Hubwieser, P. (2016). Identifying and formulating teachers' beliefs and motivational orientations for computer science teacher education. *Studies in Higher Education*, *41*(11), 1958–1973. https://doi.org/10.1080/03075079.2015.1004233

Bers, M. U. (2008). *Blocks to robots: Learning with technology in the early childhood classroom*. New York, NY: Teachers College Press.

Bers, M. U. (2010). The TangibleK robotics program: Applied computational thinking for young children. *Early Childhood Research and Practice*, *12*(2). Retrieved from http://ecrp.uiuc.edu/v12n2/bers.html

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers and Education*, *72*, 145–157. https://doi.org/10.1016/j.compedu.2013.10.020

Bers, M. U., Ponte, I., Juelich, C., Viera, A., & Schenker, J. (2002). Teachers as designers: Integrating robotics in early childhood education. *Information Technology in Childhood Education Annual*, *14*, 123-145.

Bers, M. U., & Portsmore, M. (2005). Teaching partnerships: Early childhood and engineering students teaching math and science through robotics. *Journal of Science Education and Technology*, *14*(1), 59–73. https://doi.org/10.1007/s10956-005-2734-1

Black, A. E., & Deci, E. L. (2000). The effects of instructors' autonomy support and students' autonomous motivation on learning organic chemistry: a self-determination theory perspective. *Science Education, 84*(6), 740-756.

Bland, J. M., & Altman, D. G. (1995). Multiple significance tests: The Bonferroni method. *BMJ, 310*(170).

Bloom, B., Engelhart, M., Furst, E., Hill, W., & Krathwohl, D. (1956). *Taxonomy of educational objectives: The classification of educational goals*. New York: David McKay.

Bloomberg, L. D., & Volpe, M. (2016). *Completing your qualitative dissertation: A road map from beginning to end* (3rd ed.). Thousand Oaks: Sage.

Böhm, C., & Jacopini, G. (1966). Flow diagrams, turing machines and languages with only two formation rules. *Communications of the ACM, 9*(5), 366–371. https://doi.org/10.1145/355592.365646

Bonar, J., & Soloway, E. (1983). Uncovering principles of novice programming. *Proceedings of the 10th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, Austin, Texas.

Bower, M., Wood, L. N., Lai, J. W., Howe, C., Lister, R., Mason, R., Highfield, K., & Veal, J. (2017). Improving the computational thinking pedagogical capabilities of school teachers. *Australian Journal of Teacher Education, 42*(3). http://dx.doi.org/10.14221/ajte.2017v42n3.4

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology, 3*(2), 77-101. Retrieved from http://dx.doi.org/10.1191/1478088706qp063oa

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *AERA, 727*, 135–160. https://doi.org/10.1007/978-3-319-64051-8_9

Brookhart, S. M., & Freeman, D. J. (1992). Characteristics of entering teacher candidates. *Review of Educational Research*, *62*, 37–60. http://dx.doi.org/10.3102/00346543062001037

Brosterman, N. (1997). *Inventing kindergarten*. New York, NY: Harry N. Adams Inc.

Bruciati, A.P. (2004). Robotics technologies for K-8 educators: A semiotic approach for instructional design. *Education Faculty Publications*. Paper 56. http://digitalcommons.sacredheart.edu/ced_fac/56

Bruder, S., & Wedeward, K. (2003). Robotics in the classroom. *IEEE Robotics & Automation Magazine*, *10*(3), 25–29

Bryan, L. A. (2003). Nestedness of beliefs: Examining a prospective elementary teacher's belief system about science teaching and learning. *Journal of Research in Science Teaching*, *40*(9), 835–868

Bucks, G. W. (2010). *A phenomenographic study of the ways of understanding conditional and repetition structures in computer programming languages*. Retrieved from http://ezproxy.library.nyu.edu:2148/pqdtft/docview/858607918/abstract/138B68F802A21839DB5/39?accountid=12768%5Cnhttp://ezproxy.library.nyu.edu:2283/media/pq/classic/doc/2302029981/fmt/ai/rep/NPDF?hl=scratches,scratch,programming,programming&cit:auth=Bucks

Burke, Q. (2012). The markings of a new pencil: Introducing programming-as-writing in the middle school classroom. *Journal of Media Literacy Education*, *4*(2), 121–135.

Burke, Q. & Kafai, Y. B. (2014). Decade of game making for learning: From tools to communities. In M.C. Angelides & H. Agius (Eds.), *Handbook of digital games* 689-709. New York: John Wiley & Sons.

Burke, Q., Schep, M, & Dalton, T. (2016). CS for SC: A landmark report on K-12 computer science in South Carolina. *National Science Foundation*. 1-19.

Buss, R.R., & Zambo, D. (2014). *A practical guide for students and faculty in CPED-influenced programs working on an action research dissertation in practice*. Retrieved from http://www.cpedinitiative.org/resource/resmgr/Literature/ARbuss_zambo_cped_product.pdf

Casler-Failing, S. L. (2017). *The effects of integrating Lego robotics into a mathematics curriculum to promote the development of proportional reasoning* (Doctoral dissertation). Retrieved from ProQuest Dissertation & Theses. (AAT 10204060)

Castledine, A. R., & Chalmers, C. (2011). LEGO robotics: An authentic problem solving tool? *Design and Technology Education*, *16*, 19–27.

Catlin, D. (2012). Maximising the effectiveness of educational robotics through the use of assessment for learning methodologies. *Proceedings of 3rd International workshop teaching Robotics, Teaching with Robotics, Integrating Robotics in School Curriculum*. Trento, Italy. http://www.terecop.eu/TRTWR2012/trtwr2012_submission_01.pdf

Cejka, E., Rogers, C., & Portsmore, M. (2006). Kindergarten robotics: Using robotics to motivate math, science, and engineering literacy in elementary school. *International Journal of Engineering Education, 22*(4), 711–722.

Ceruzzi, P. E. (1998). *A history of modern computing*. Cambridge, MA: MIT Press.

Chambers, J. M., & Carbonaro, M. (2003). Designing, developing, and implementing a course on LEGO robotics for technology teacher education. *Journal of Technology and Teacher Education, 11*(2), 209–242.

Chan, T. S., & Ahern, T. C. (1999). Targeting motivation – adapting flow theory to instructional design. *Journal of Educational Computing Research 21*(2), 152–163

Chen, Z., & Yeung, A. S. (2015). Self-efficacy in teaching Chinese as a foreign language in Australian schools. *Australian Journal of Teacher Education, 40*(8). https://doi.org/10.14221/ajte.2015v40n8.2

Cheng, P. L. (2017). *Evaluating intention to use remote robotics experimentation in programming courses* (Doctoral dissertation). Retrieved from ProQuest Dissertation & Theses. (AAT 10273171)

Chiu, C.-F., & Huang, H.-Y. (2015). Guided debugging practices of game based programming for novice programmers. *International Journal of Information and Education Technology, 5*(5), 343–347. https ://doi.org/10.7763/IJIET .2015.V5.527.

Cliburn, D. (2006). A CS0 course for the liberal arts. *Proceedings of the 37th ACM Technical Symposium on Computer Science Education (SIGCSE'06).* 77–81.

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences*. Nature (2nd ed., Vol. 506). Lawrence Erlbaum Associates. https://doi.org/10.1038/506274a

Cohen, J. (1992). A power primer. *Psychological Bulletin, 112*(1), 155-159. http://doi.org/10.1037/0033-2909.112.1.155.

Cooper, S., Dann, W., & Pausch, R. (2000). Alice: A 3-D tool for introductory programming concepts. *Journal of Computer Sciences in Colleges 15*(5), 107–116.

Corbin, J., & Strauss, A. (2008). *Basics of qualitative research: Techniques and procedures for developing grounded theory* (3rd ed.). Sage Publications, Inc. https://doi.org/10.4135/9781452230153

Creswell, J. W. (2014). *Research design: Qualitative, quantitative, and mixed method approaches* (4th ed.). Thousand Oaks, CA: Sage.

Creswell, J.W. (2017). *Qualitative inquiry and research design: Choosing among the five traditions*. Thousand Oaks, CA: Sage Publications.

Creswell, J. W., & Plano Clark, V. L. (2018). *Designing and conducting mixed methods research* (3rd ed.). Thousand Oaks, CA: Sage.

Creswell, J. W., & Poth, C. N. (2018). *Qualitative inquiry and research design: Choosing among five approaches* (4th ed.). Los Angeles: Sage.

Csikszentmihalyi, M. (1975). *Beyond boredom and anxiety.* San Francisco, CA: Jossey-Bass.

Csikszentmihalyi, M. (1990). *Flow: the psychology of optimal experience*. New York, NY: Harper & Row.

Csikszentmihalyi, M. (2000). *Beyond boredom and anxiety: experiencing flow in work and play*. San Francisco, CA: Jossey-Bass.

Cullen, T. A., & Greene, B. A. (2011). Preservice teachers' beliefs, attitudes, and motivation about technology integration. *Journal of Educational Computing Research*, *45*(1), 29–47. https://doi.org/10.2190/EC.45.1.b

Curzon, P., Cutts, Q. I., & Bell, T. (2009). Enthusing and inspiring with reusable kinaesthetic activities. *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education, Paris, France, 3-7 Jul 2009*, (pp. 94-98). doi:10.1145/1595496.1562911

Dagdilelis, V., Sartatzemi, M., & Kagani, K. (2005). Teaching (with) robots in secondary schools: Some new and not-so-new pedagogical problems. *Proceedings - 5th IEEE International Conference on Advanced Learning Technologies, ICALT 2005*, *2005*(January), 757–761. https://doi.org/10.1109/ICALT.2005.255

Davis E. A., Petish, D., & Smithey, J. (2006). Challenges new science teachers face. *Review of Educational Research 76*(4), 607–651

Deci, E. L. (1992). The relation of interest to motivation of behavior: A self-determination theory perspective. In K.A. Renninger, S. Hidi & A. Krapp (Eds.), *The role of interest in learning and development* (pp. 3-25). Hill*SD*ale, NJ: Lawrence Erlbaum Associates.

Deci, E. L., & Ryan R. M. (2000). The ‘what’ and ‘why’ of goal pursuits: Human needs and the self-determination of behavior. *Psychological Inquiry, 11*, 227–68. doi:10.1207/ S15327965PLI1104_01

DeClue, T. H. (2003). Pair programming and pair trading: Effects on learning and motivation in a cs2 course. *Journal of Computing Sciences in Colleges, 18*(5), 49-56.

Denis, B., & Hubert, S. (2001). Collaborative learning in an educational robotics environment. *Computers in Human Behavior, 17*, 465–480.

DeVellis, R. F. (2003). *Scale development: Theory and applications*. Thousand Oaks, CA: Sage.

Devlin, A. S. (2017). *The research experience: Planning, conducting, and reporting research*. Thousand Oaks: Sage.

Dewey, J. (1913). Interest and effort in education. In J.A. Boydston (Ed.), *The middle works, 1899-1924: Vol.7 1912-1914* (pp. 153-197), Carbondale, IL: Southern Illinois University Press.

Dijkstra, E. W. (1976). *A discipline of programming*. Englewood Cliffs, N.J.: Prentice Hall.

Dodds, Z., Greenwald, L., Howard, A., Tejada, S., & Weinberg, J. (2006). Components, curriculum, and community: Robots and robotics in undergraduate AI education. *AI Magazine*, *27*(1), 11–22.

Donzeau-Gouge, V., Huet, G., Lang, B., & Kahn, G. (1984). Programming environments based on structured editors: The MENTOR experience. In D. Barstow, H. E. Shrobe, & E. Sandewall (Eds.), *Interactive Programming Environments*. McGraw Hill.

Dörnyei, Z., & Ushioda, E. (2011). *Teaching and researching motivation* (2nd ed.). New York, NY: Longman.

Driscoll, M. (2005). *Psychology of learning for instruction* (3rd ed.). Boston, MA: Allyn and Bacon.

Dwyer, S.C., & Buckle, J.L. (2009). The space between: On being an insider-outsider in qualitative research. *International Journal of Qualitative Methods*, *8*(1), 54-63.

Eccles, J. S., Simpkins, S. D., & Davis-Kean, P. E. (2006). Math and science motivation: A longitudinal examination of the links between choices and beliefs. *Developmental Psychology, 42*, 70–83.

Egbert, J. (2003). A study of flow theory in the foreign language classroom. *Modern Language Journal 87*(4), 499–518.

Eguchi, A. (2007). Educational robotics for undergraduate freshmen. *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2007*, 1792–1797. Retrieved from http://www.editlib.org/INDEX.CFM?fuseaction=Reader.ViewAbstract&amp;paper_id=25614

Eguchi, A. (2012). Educational robotics theories and practice: Tips for how to do it right. In B. S. Barker, G. Nugent, N. Grandgenett & V. I. I. Adamchuk (Eds.), *Robots in K-12 education: A new technology for learning* (pp. 1–30). Hershey, PA: Information Science Reference.

Eguchi, A. (2013). Educational robotics for promoting 21st century skills. *Journal of Automation, Mobile Robotics & Intelligent Systems*, *8*(1), 1–42. https://doi.org/10.14313/JAMRIS

Ertmer, P. A., & Ottenbreit-Leftwich, A. T. (2010). Teacher technology change: How knowledge, confidence, beliefs, and culture intersect. *Journal of Research on Technology in Education*, *42*(3), 255-284. https://doi.org/10.1080/15391523.2010.10782551

Ertmer, P. A., Ottenbreit-Leftwich, A. T., Sadik, O., Sendurur, E., & Sendurur, P. (2012). Teacher beliefs and technology integration practices: A critical relationship.

Computers & Education, 59(2), 423-435. https://doi.org/10.1016/j.compedu.2012.02.001

Erwin, B., Cyr, M., & Rogers, C. (2000). LEGO engineer and RoboLab: Teaching engineering with LabVIEW from kindergarten to graduate school. *International Journal of Engineering Education, 16*(3), 181-192.

Falloon, G. (2016). An analysis of young students' thinking when completing basic coding tasks using Scratch Jnr. on the iPad. *Journal of Computer Assisted Learning*, *32*(6), 576–593. https://doi.org/10.1111/jcal.12155

Feldgen, M. & Clua, O. (2004), Games as a motivation for freshman students learn programming, in *'Frontiers in Education, 2004. FIE 2004. 34th Annual'*, pp. S1H/11–S1H/16 Vol. 3.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5-6 years old kindergarten children in a computer programming environment: A case study. *Computers and Education*, *63*, 87–97. https://doi.org/10.1016/j.compedu.2012.11.016

Feurzeig, W., Papert, S., Bloom, M., Grant, R., & Soloman, C. (1969). Programming-languages as a conceptual framework for teaching mathematics. *Programming-languages as a conceptual framework for teaching mathematics. Final report on the first fifteen months of the Logo Project* (Technical Report 1889). Cambridge, MA: BBN.

Field, A. P. (2009). *Discovering statistics using SPSS*. Los Angeles; London: SAGE.

Fishbein, M., & Ajzen, I. (1972). Beliefs, attitudes, intentions and behavior: An introduction to theory and research. Reading, MA: Foster.

Fitzgerald, S., Lewandowski, G., McCauley, R., Murphy, L., Simon, B., Thomas, L., et al. (2008). Debugging: Finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education, 18*(2), 93–116. https ://doi.org/10.1080/08993 40080 21145 08.

Fredricks, J. A., Blumenfeld, P. C., & Paris, A. H. (2004). School engagement: Potential of the concept, state of the evidence. *Review of Educational Research, 74*(1), 59–109.

Frels R. K., & Onwuegbuzie, A. J. (2013). Administering quantitative instruments with qualitative interviews: A mixed research approach. Journal of Counseling & Development, 91(1), 184-194.

Garcia, D., Harvey, B., & Barnes, T. (2015). *The beauty and joy of computing*. ACM Inroads, 6(4), 71–79. https://doi.org/10.1145/2835184

Gibbons, J. D., & Chakraborti, S. (2011). *Nonparametric statistical inference* (5th ed.). Boca Raton, FL: Taylor & Francis Group, LLC.

Glynn, S. M., Brickman, P., Armstrong, N., & Taasoobshirazi, G. (2011). Science motivation questionnaire II: Validation with science majors and nonscience majors. *Journal of Research in Science Teaching, 48*(10), 1159–1176. https://doi.org/10.1002/tea.20442

Glynn, S. M., Taasoobshirazi, G., & Brickman, P. (2009). Science motivation questionnaire: Construct validation with nonscience majors. *Journal of Research in Science Teaching, 46*(2), 127–146. https://doi.org/10.1002/tea.20267

Goh, H., & Ali, B. (2014). Robotics as a tool to stem learning. *International Journal for Innovation Education and Research, 2*(10), 66–78.

Good, J. (2011). Learners at the wheel: Novice programming environments come of age. *International Journal of People-Oriented Programming, 1*(1), 1-24. http://dx.doi.org/ 10.4018/ijpop.2011010101.

Google Inc., & Gallup Inc. (2016). *Trends in the state of computer science in U.S. K-12 schools.* Retrieved from http://goo.gl/j291E0

Govender, I., & Grayson, D. J. (2008). Pre-service and in-service teachers' experiences of learning to program in an object-oriented language. *Computers and Education, 51*(2), 874–885. https://doi.org/10.1016/j.compedu.2007.09.004

Greenley, W., & Tidwell, C. (2002). Legos in the classroom?: Teaching computer programming to advanced high school students. In *Hawaiian International Business Conference* (pp. 1–16).

Greenwood, D. J., & Levin, M. (2007). *Introduction to action research* (Vol. 2). Thousand Oaks, CA: Sage.

Grover, S., & Basu, S. (2017). Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and Boolean logic. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education - SIGCSE '17*, 267–272. https://doi.org/10.1016/j.pupt.2006.04.005

Grover, S., & Pea, R. (2013). Computational thinking in k-12: A review of the state of the field. *Educational Researcher*, *42*(1), 38–43. https://doi.org/10.3102/0013189X12463051

Gunbatar, M. S., & Karalar, H. (2018). Gender differences in middle school students' attitudes and self-efficacy perceptions towards mBlock programming. *European*

Journal of Educational Research, 7(4), 925–933. https://doi.org/10.12973/eu-jer.7.4.923

Gunning, A. M., & Mensah, F. M. (2011). Preservice elementary teachers' development of self-efficacy and confidence to teach science: A case study. *Journal of Science Teacher Education*, *22*(2), 171-185. https://doi.org/10.1007/s10972-010-9198-8

Guzdial, M., & Soloway, E. (2002). Teaching the Nintendo generation to program. *Communications of the ACM, 45*(4),17–21.

Hadjiachilleos, S., Avraamidou, L., & Papastavrou, S. (2013). *The use of Lego technologies in elementary teacher preparation. Journal of Science Education and Technology, 22*(5), 614–629.

Hamari, J., Koivisto, J., & Sarsa, H. (2014). Does gamification work? A literature review of empirical studies on gamification. Paper presented at the *2014 47th Hawaii International Conference on System Sciences*.

Han, I. (2013). Embodiment: A new perspective for evaluating physicality in learning. *Journal of Educational Computing Research, 49*(1), 41– 59. doi:10.2190/EC.49.1.b.

Han, J., & Yin, H. (2016). Teacher motivation: Definition, research development and implications for teachers. *Cogent Education*, *3*(1), 1–18. https://doi.org/10.1080/2331186X.2016.1217819

Hanus, M. D., & Fox, J. (2015). Assessing the effects of gamification in the classroom: A longitudinal study on intrinsic motivation, social comparison, satisfaction, effort, and academic performance. *Computers & Education*, 80, 152-161.

Harasim, L. (2012). *Learning theory and online technologies*. New York: Routledge.

Harel, I., & Papert, S. (1991). *Constructionism*. Norwood NJ: Ablex Publishing.

Hathcoat, J. D., & Meixner, C. (2017). Pragmatism, factor analysis, and the conditional incompatibility thesis in mixed methods research. *Journal of Mixed Methods Research, 11*(4), 433–449. https://doi.org/10.1177/1558689815622114

Herr, K., & Anderson, G.L. (2005). *The action research dissertation*. Thousand Oaks, CA: Sage.

Hopkins, C. D. & Antes, R. L. (1990) *Classroom management and evaluation* (3rd ed.). F.E. Itasca, IL: Pencock Publishing, Inc.

Howe, J. A. M. (1981). Learning mathematics through Logo programming (Research Paper No 153). *Department of Artificial Intelligence*. Edinburgh: University of Edinburgh.

Huang, K. H., Yang, T. M., & Cheng, C. C. (2013). Engineering to see and move: Teaching computer programming with flowcharts vs. LEGO robots. *International Journal of Emerging Technologies in Learning*, *8*(4), 23–26.

Huang, Y., Backman, S. J., & Backman, K. F. (2010). Student attitude toward virtual learning in second life: A flow theory approach. *Journal of Teaching in Travel and Tourism*, *10*(4), 312–334.

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers and Education, 82*, 263–279. https://doi.org/10.1016/j.compedu.2014.11.022

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. https://doi.org/10.1007/s10956-016-9663-z

Jayathirtha, G., Fields, D. A., & Kafai, Y. B. (2018). Computational concepts, practices, and collaboration in high school students' debugging electronic textile projects. In the conference proceedings of the *International conference on computational thinking education (CTE'18)*, The Education University of Hong Kong, Hong Kong, China. https://par.nsf.gov/servlets/purl/10101544

Jenkins, T. (2001). The motivation of students of programming. *ACM SIGCSE Bulletin, 33*(3), 53-56.

Jenkins, T. & Davy, J. (2002). Diversity and motivation in introductory programming. *Innovation in Teaching and Learning in Information and Computer Sciences (1)*1. 1-9.

Johns, G. (1996). *Organizational behavior: Understanding and managing life at work* (4th ed.). New York: HarperCollins.

Johnson-Laird, P. N. (1983). *Mental models: towards a cognitive science of language, inference, and consciousness*. Cambridge, MA: Harvard University Press.

Jonassen, D. H. (2000). *Computers as mindtools for schools: Engaging critical thinking* (2nd ed). Upper Saddle River, NJ: Prentice Hall.

Jonassen, D. H. (2011). *Learning to solve problems: A handbook for designing problem solving learning environments*. New York, NY: Routledge.

Kabatova, M., & Pekarova, J. (2010). Learning how to teach robotics. *Constructionism 2010 Conference*, 1–8. https://doi.org/10.18848/1447-9494/cgp/v15i06/45812

Kafai, Y. B., & Resnick, M. (1996). Constructionism in practice: Designing, learning and thinking in a digital world. Mahwah, NJ: Lawrence Erlbaum Associates.

Kalyuga, S. (2010). Schema acquisition and sources of cognitive load. *Cognitive Load Theory*, 48–64. https://doi.org/10.1017/CBO9780511844744.005

Karahoca, D., Karahoca, A., & Uzunboylu, H. (2011). Robotic teaching in primary school education by project based learning for supporting science and technology courses. *Procedia Computer Science*, *3*, 1425–1431.

Katz, I. R., & Anderson, J. R. (1987). Debugging: An analysis of bug-location strategies. *Human–Computer Interaction*, *3*(4), 351.

Kay, J. S., Moss, J. G., Engelman, S., & McKlin, T. (2014). Sneaking in through the back door: Introducing K-12 teachers to robot programming. *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, 499–504. https://doi.org/10.1145/2538862.2538972

Kaya, E., Newley, A., Deniz, H., Yesilyurt, E., & Newley, P. (2015). Introducing engineering design to a science teaching methods course through educational robotics and exploring changes in views of preservice elementary teachers. *Journal of College Science Teaching*, *47*(2), 66–75.

Kazakoff, E., Sullivan, A., & Bers, M. (2013). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Childhood Education Journal*, *41*(4), 245–255.

Kelleher, C., Pausch, R., & Kiesler, S. (2007). Storytelling Alice motivates middle school girls to learn computer programming. *Proceedings of the SIGCHI Conference on*

Human Factors in Computing Systems, CHI '07, ACM. New York, NY, USA, (pp. 1455–1464).

Keller, J. M. (1983). Motivational design of instruction. In C. M. Reigeluth (Ed.), *Instructional Design Theories and Models: An Overview of Their Current Status* (pp. 386–434). Hill*SD*ale, NJ: Lawrence Erlbaum Associates.

Keller, J. M. (1987). *IMMS: Instructional materials motivation survey*. Tallahassee, FL: Florida State University.

Kim, C., Kim, D., Yuan, J., Hill, R. B., Doshi, P., & Thai, C. N. (2015). Robotics to promote elementary education pre-service teachers' STEM engagement, learning, and teaching. *Computers and Education*, *91*, 14–31. https://doi.org/10.1016/j.compedu.2015.08.005

Kim, C., Yuan, J., Kim, D., Doshi, P., Thai, C. N., Hill, R. B., & Melias, E. (2017). Studying the usability of an intervention to promote teachers' use of robotics in STEM education. *Journal of Educational Computing Research, 56*(8), 1179–1212. https://doi.org/10.1177/0735633117738537

Kim, C., Yuan, J., Vasconcelos, L., Shin, M., & Hill, R. B. (2018). Debugging during block-based programming. *Instructional Science*, *46*(5), 767–787. https://doi.org/10.1007/s11251-018-9453-5

Klatzky, R.L. (1980). Human memory: Structures and processes. San Francisco, CA: W.H. Freeman & Co.

Kochhar-Bryant, C. A. (2017). Symbiotic space: exploring the nexus of rigor, problems of practice and implementation. *Impacting Education: Journal on Transforming Professional Practice, 2*(1), 6–14. https://doi.org/10.5195/IE.2017.25.

Koller, A., & Kruijff, G.-J. M. (2004). Talking robots with LEGO MindStorms. *Proceedings of the 20th International Conference on Computational Linguistics - COLING '04*. https://doi.org/10.3115/1220355.1220404

Kolling, M. & Rosenberg, J. (2001). Guidelines for teaching object orientation with Java. *SIGCSE Bullitin 33*(3), 33–36.

Kopcha, T. J., McGregor, J., Shin, S., Qian, Y., Choi, J., Hill, R., … Choi, I. (2017). Developing an integrative STEM curriculum for robotics education through educational design research. *Journal of Formative Design in Learning, 1*(1), 31–44. https://doi.org/10.1007/s41686-017-0005-1

Krapp, A., Hidi, S., & Renninger, K. A. (1992). Interest, learning, and development. In K. A. Renninger, S. Hidi, & A. Krapp (Eds.), *The role of interest in learning and development* (pp. 3- 25). Hill*SD*ale, NJ: Lawrence Erlbaum Associates, Inc.

Kristensen, B. B., & Osterbye, K. (1994). Conceptual modeling and programming languages. *ACM SIGPLAN Notices 29*(9), 81-90.

Kucuk, S., & Sisman, B. (2018). Pre-service teachers' experiences in learning robotics design and programming. *Informatics in Education*, *17*(2), 301–320. https://doi.org/10.15388/infedu.2018.16

Kuittinen, M., & Sajaniemi, J. (2004). Teaching roles of variables in elementary programming courses. *SIGCSE Bulletin 36*(3) 57-61.

Lambert, L., & Guiffre, H. (2009). Computer science outreach in an elementary school. *Journal of Computing Sciences in Colleges, 24*(3), 118–124.

Landry, C. L. (2003). *Self-efficacy, motivation, and outcome expectation correlates of college students' intention certainty* (Doctoral dissertation). Retrieved from ProQuest Dissertation & Theses. (AAT 3085864)

Lanzonder, A. (2005). Do two heads search better than one? Effects of student collaboration on web search behaviour and search outcomes. *British Journal of Educational Technology, 36*(3), 465–475.

Lauwers, T., Nourbakhsh, I., & Hamner, E. (2009). CSbots: Design and deployment of a robot designed for the CS1 classroom. *Proceedings of the 40th Technical Symposium on Computer Science Education (SIGCSE'09)*. 428–432.

Law, K., Lee, V., & Yu, Y. (2010). Learning motivation in e-learning facilitated computer programming courses. *Computers and Education (55)*1. 218-228. https://doi.org/10.1016/j.compedu.2010.01.007

Lawson, A. E., Banks, D. L., & Logvin, M. (2007). Self-efficacy, reasoning ability, and achievement in college biology. *Journal of Research in Science Teaching, 44,* 706–724.

Levin, J. A., & Kereev, Y. (1980). *Personal computers and education: The challenge to schools. Center for Human Information Processing*. La Jolla, CA: University of California – San Diego.

Levin, T., & Long, R. (1981). *Effective instruction.* Alexandria, VA: Association for Supervision and Curriculum Development.

Lister, R., Seppälä, O., Simon, B., Thomas, L., Adams, E. S., Fitzgerald, S., … Sanders, K. (2004). A multi-national study of reading and tracing skills in novice programmers. *Working group reports from ITiCSE on Innovation and technology* *in computer science education - ITiCSE-WGR '04*. https://doi.org/10.1145/1044550.1041673

Lincoln, Y. S., & Guba, E. G. (1985). *Naturalistic inquiry*. Thousand Oaks, CA: Sage.

Lindh, J., & Holgersson, T. (2007). Does Lego training stimulate pupils ability to solve logical problems? *Computers & Education*, *49*(4), 1097-1111.

Lord, F. M. (1952). The relationship of the reliability of multiple-choice test to the distribution of item difficulties. *Psychometrika*, *17*, 181-194.

Lu, J. J., & Fletcher, G. (2009). Thinking about computational thinking. *Proceedings of the 40th ACM Technical Symposium on Computer Science Education (SIGCSE'09).*

Lye, S., & Koh, J. (2014). Review on teaching and learning of computational thinking through programming: What is next for K–12? *Computers in Human Behaviour*, *41*, 51–61.

Majherova, J., & Kralik, V. (2017). Innovative methods in teaching programming for future informatics teachers. *European Journal of Contemporary Education*, *6*(3), 390–401. https://doi.org/10.13187/ejced.2017.3.390

Major, L., Kyriacou, T., & Brereton, P. (2014). The effectiveness of simulated robots for supporting the learning of introductory programming: a multi-case case study. *Computer Science Education*, *24*(2–3), 193–228. https://doi.org/10.1080/08993408.2014.963362

Malan, D. J., & Leitner, H. H. (2007). Scratch for budding computer scientists. *ACM SIGCSE Bulletin*, *39*(1), 223. https://doi.org/10.1145/1227504.1227388

Manches, A., & Price, S. (2011). Designing learning representations around physical manipulation, 81–89. *Proceedings of the 10th international conference on interaction design and children*. https://doi.org/10.1145/1999030.1999040

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014). Computational thinking in K-9 education. *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference - ITiCSE-WGR '14*, 1–29. https://doi.org/10.1145/2713609.2713610

Martin, A. J. (2003). The Student Motivation Scale: Further testing of an instrument that measures school students' motivation. *Australian Journal of Education, 47*, 88-106.

Martin, A. J. (2007). Examining a multidimensional model of student motivation and engagement using a construct validation approach. *British Journal of Educational Psychology*, *77*(2), 413–440. https://doi.org/10.1348/000709906X118036

Martin, A. (2012). Part II commentary: Motivation and engagement: Conceptual, operational, and empirical clarity. In S. L. Christenson, A. L. Reschly, & C. Wylie (Eds.), *Handbook of Research on Student Engagement* (pp. 303-311). New York, NY: Springer US.

Martin, F. G., Mikhak, B., Resnick, M., Silverman, B. & Berg, R. (2000). To Mindstorms and beyond: Evolution of a construction kit for magical machines. In A. Druin & J. Hendler (Eds.), *Robots for kids: Exploring new technologies for learning* (pp. 9–33). San Mateo, CA: Morgan Kaufmann.

Martin, F. G, Scribner-MacLean, M., Christy, S., Rudnicki, I., Londhe, R., Manning, C., & Goodman, I. F. (2011). Reflections on iCODE: Using web technology and hands-on projects to engage urban youth in computer science and engineering. *Autonomous Robots*, *30*(3), 265–280. https://doi.org/10.1007/s10514-011-9218-3

Martin, F. G., & Resnick, M. (1993). LEGO/Logo and electronic bricks: creating a scienceland for children. In D. Ferguson (Ed.), *Advanced educational technologies for mathematics and science* (pp. 61–90). Berlin: Springer

Marsh, H. W., Balla, J. R., & McDonald, R. P. (1988). Goodness-of-fit indices in confirmatory factor analysis: The effect of sample size. *Psychological Bulletin, 102*, 391-410.

Marzano, R. J. (2007). *The art and science of teaching*. Alexandria, VA: ASCD.

Maslow, A. H. (1943). A theory of human motivation. *Psychology Review 50*, 370–396.

Maxwell, J.A. (2010). Using numbers in qualitative research. *Qualitative Inquiry, 16*(6), 475–482. http://doi.org/10.1177/1077800410364740

Mayer, R. E. (1979). A psychology of learning BASIC. *Communications of the ACM, 22*(11), 589-593.

Mayer, R. E. (1981). The psychology of how novices learn computer programming. *ACM Computing Surveys*, *13*(1): 121-141.

McClelland, J. L. (2011). Memory as a constructive process: The parallel-distributed processing approach. In S. Nalbantian, P. Matthews, and J. L. McClelland (Eds.), *The memory process: Neuroscientific and humanistic perspectives* (pp. 129-151). Cambridge, MA: MIT Press.

McGill, M. M. (2012). Learning to program with personal robots. *ACM Transactions on Computing Education*, *12*(1), 1–32. https://doi.org/10.1145/2133797.2133801

McGill, T. J., & Volet, S. E. (1997). A conceptual framework for analyzing students' knowledge. *Journal of Research on Computing in Education 29*(3), 276-298.

McMillan, J. H. (2016). *Fundamentals of educational research* (7th ed.). Boston: Pearson.

McNally, M., Goldweber, M., Fagin, B., & Klassner, F. (2006). Do Lego Mindstorms robots have a future in CS education? *ACM SIGCSE Bulletin*, *38*(1), 61. https://doi.org/10.1145/1124706.1121362

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science with scratch. *Computer Science Education*, *23*(3), 239–264.

Merriam, S.B. (1998). *Qualitative research and case study applications in education*. San Francisco, CA: Jossey-Bass.

Merriam, S.B., Johnson-Bailey, J., Lee, M.-Y., Kee, Y., Ntseane, G., & Muhamad, M. (2001). Power and positionality: Negotiating insider/outsider status within and across cultures. *International Journal of Lifelong Education*, *20*(5), 405–416. http://doi.org/10.1080/02601370120490

Merriam, S. B. & Tisdell, E. J. (2016). *Qualitative research: A guide to design and implementation*, (4th ed.). Hoboken, N.J.: Wiley.

Mertler, C. A. (2017). *Action research: Improving schools and empowering educators* (5th ed.). Thousand Oaks, CA: Sage.

Mertens, D.M. (2009). *Research and evaluation in education and psychology: Integrating diversity with quantitative, qualitative, and mixed methods*. Thousand Oaks, CA: Sage.

Mikropoulos, T., & Bellou, I. (2013). Educational robotics as mindtools. *Themes in Science & Technology Education., 6*(1), 5–14.

Miller, G. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review, 101*(2), 343–352. https://doi.org/10.1037/h0043158

Mills, G. E. (2018). *Action research: A guide for the teacher researcher* (6th ed.). New York: Pearson.

Milton, M., Rohl, M., & House, H. (2007). Secondary beginning teacher's preparedness to teach literacy and numeracy: A survey. *Australian Journal of Teacher Education, 32*(2). https://doi.org/10.14221/ajte.2007v32n2.4

Moreno-Leon, J., & Robles, G. (2015). Developing mathematical thinking with Scratch: An experiment with 6th grade students. *Proceedings of design for teaching and learning in a networked world: 10th European conference on technology enhanced learning, EC-TEL* (Vol. 9307). Toledo, Spain. https://doi.org/10.1007/978-3-319-24258-3

Morgan, D. (2014). *Integrating qualitative and quantitative methods: A pragmatic approach*. Thousand Oaks, CA: Sage. https://doi.org/10.1016/B978-0-444-53802-4.00055-5

Morgan, D. (2018). *Basic and advanced focus groups*. Thousand Oaks, CA: Sage.

Myers, B. A. (1990). Taxonomies of visual programming and program visualization. *Journal of Visual Languages & Computing*, *1*(1), 97–123.

National Association of Colleges and Employers. (2018). *Salary survey: Winter 2018*. Retrieved from https://careers.kennesaw.edu/employers/docs/2018-nace-salary-survey-winter.pdf

Navarro-Prieto, R. & Canas, J. J. (2001). Are visual programming languages better? The role of imagery in program comprehension. *International Journal of Human-Computer Studies, 54*, 799-829.

Newell, A. & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Norman, D. A., & Rumelhart, D. E. (1975). *Explorations in cognition*. San Francisco, CA: Freeman.

Nugent, G., Barker, B., Grandgenett, N., & Adamchuk, V. (2010). Impact of robotics and geospatial technology interventions on youth STEM learning and attitudes. *Journal of Research on Technology in Education, 42*(4), 391-408. doi: 2056317171.

O'Keefe, P. A. & Harackiewicz, J. M. (2017). *The multifaceted role of interest in motivation and engagement*. In P.A. O'Keefe & J. M. Harackiewicz (Eds.), *The Science of Interest*. Montreal, Canada: Springer International Publishing. https://doi.org/10.1007/978-3-319-55509-6

Ormerod, T. (1990). Human cognition and programming. In J. M. Hoc, T. R. G. Green, R. Samurcay, and D. J. Gilmore (Eds.), *Psychology of Programming* (63-82). San Diego, CA: Academic Press.

Ortiz, A., Bos, B., & Smith, S. (2015). The power of educational robotics as an integrated STEM learning experience in teacher preparation programs. *Journal of College Science Teaching*, *44*(5). https://doi.org/10.2505/4/jcst15_044_05_42

Osborne, R. B., Thomas, A. J., & Forbes, J. (2010). Teaching with robots: A service-learning approach to mentor training. In *ACM Technical Symposium on Computer Science Education (SIGCSE 2010)*, Milwaukee, WI. https://doi.org/10.1145/1734263.1734321

Pajares, F. (1996). Self-efficacy beliefs in achievement settings. *Review of Educational Research 66*, 543-578.

Palak, D., & Walls, R. T. (2009). Teachers' beliefs and technology practices: A mixed methods approach. *Journal of Research on Technology in Education*, *47*(4), 417-441.

Pallant, J. (2007). *SPSS survival manual: A step by step guide to data analysis using SPSS for Windows*, 3rd Edition. McGraw Hill Open University Press, New York.

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

Papert, S. (1990). A critique of technocentrism in thinking about the school of the future. *MIT Epistemology and Learning Memo No. 2*. Cambridge, MA: MIT Media Laboratory.

Papert, S. (1993). *The children's machine.* New York, NY: Basic Books

Papert, S. (1999). *Constructionism: Research reports and essays.* Norwood, NJ: Ablex.

Papert, S., Watt, D., diSessa, A., & Weir, S. (1979). An assessment and documentation of a children's computer laboratory. *Final Report of the Brookline Logo Project*. Brookline, MA.

Pappas, P. A., & DePuy, V. (2004). An overview of non-parametric tests in SAS: When, Why, and How. *Duke Clinical Research Institute*. Durham: North Carolina. Retrieved from http://analytics.ncsu.edu/sesug/2004/TU04-Pappas.pdf.

Paraskeva, F., Bouta, H., & Papagianni, A. (2008). Individual characteristics and computer self-efficacy in secondary education teachers to integrate technology in educational practice. *Computers and Education, 50*(3), 1084–1091. https://doi.org/10.1016/j.compedu.2006.10.006

Parsons, S. A., & Ward, A. E. (2011). The case for authentic tasks in content literacy. *Reading Teacher*, *64,* 462–465. doi:10.1598/RT.64.6.12

Patton, M. Q. (2002). *Qualitative research and evaluation methods*. Thousand Oaks, CA: Sage.

Pea, R. (1983). Logo programming and problem solving. *ERIC Technical Report No. 12*.

Pea, R., & Kurland, M. (1984). On the cognitive effects of learning computer language. *New Ideas in Psychology, 2*(2), 137–168. https://doi.org/10.1109/BLOCKS.2015.7368989

Pennington, N. (1986). Stimulus structures and mental representations in expert comprehension of computer programs. *Cognitive Psychology, 19*, 295-341.

Pennington, N. (1987). Comprehension strategies in programming. In G. M. Olson, S. Sheppard, and E. Soloway (Eds.), *Empirical Studies of Programmers: Second Workshop* (pp. 100-113). Norwood, NJ: Ablex.

Perlman, R. (1974). TORTIS – Toddler’s Own Recursive Turtle Interpreter System. In *MIT AI Memo 311, Logo Memo 9*. Cambridge, MA: Massachusetts Institute of Technology, Retrieved from ftp://publications.ai.mit.edu/ai-publications/pdf/AIM-311.pdf

Perritt, D. C. (2010). Including professional practice in professional development while improving middle school teaching in math. *National Teacher Education Journal, 3*(3), 73–76. Retrieved from http://ezp.lib.ttu.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=eue&AN=57457833

Peshkin, A. (1988). In search of subjectivity – One's own. *Educational Researcher, 17*(7), 17-21. Retrieved from http://www.jstor.org.pallas2.tcl.sc.edu/stable/1174381

Petre, M., & Price, B. (2004). Using robotics to motivate ‘back door’ learning. *Education and Information Technologies*, *9*(2), 147–158. https://doi.org/10.1023/B:EAIT.0000027927.78380.60

Phillippi, J. & Lauderdale, J. (2018). A guide to field notes for qualitative research: Context and conversation. *Qualitative Health Research, 28*(3). Retrieved from https://journals.sagepub.com/doi/pdf/10.1177/1049732317697102

Piaget, J. (1967). *The child’s conception of the world*. New York, NY: Routledge & Kegan Paul.

Piaget, J. (1973). *To understand is to invent*. New York, NY: Basic Books.

Pintrich, P. R. (1999). The role of motivation in promoting and sustaining self-regulated learning. *International Journal of Educational Research, 31*, 459-470.

Pintrich, P. R., & DeGroot, E. V. (1990). Motivational and self-regulated learning components of classroom academic performance. *Journal of Educational Psychology, 82*(1), 33-40.

Pintrich, P. R., & Schunk, D. H. (1996). *Motivation in Education: Theory, Research, and Applications*. Englewood Cliffs, NJ: Merrill/Prentice-Hall.

Qualls, J. A., & Sherrell, L. B. (2010). Why computational thinking should be integrated into the curriculum. *Journal of Computing Sciences in Colleges, 25*, 66-71.

Ramalingam, V., & Wiedenbeck, S. (1997). An empirical study of novice program comprehension in the imperative and object-oriented styles. In *ESP '97 Papers presented at the seventh workshop on Empirical studies of programmer* (pp. 124–139). https://doi.org/10.1145/266399.266411

Renninger, K.A., & Hidi, S. (2011). Revisiting the conceptualization, measurement, and generation of interest. *Educational Psychology 46*(3),168–184.

Resnick, M. (2007). Sowing the seeds for a more creative society. *Learning & Leading with Technology, 35*(4), 18-22.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silverman, B., & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM, 52*(11), 60–67.

Resnick, M., Ocko, S., & Papert, S. (1988). LEGO, Logo, and design. *Children's Environments Quarterly, 5*(4), 14-18.

Resnick, M., & Silverman, B. (2005). Some reflections on designing construction kits for kids. *Proceedings of Interaction Design and Children Conference*, Boulder, CO.

Rheinberg, F., Vollymeyer, R., & Burns, B. (2001). FAM: A questionnaire on motivation in learning and performance situations. *Diagnostica, 47*(2), 57–66. http://dx.doi.org/10.1026//0012-1924.47.2.57

Rich, L., Perry, H., & Guzdial, M. (2004). A CS1 course designed to address interests of women. *Proceedings of the 34th ACM Technical Symposium on Computer Science Education (SIGCSE'04)*. 190–195.

Roschelle, J., & Teasley, S. D. (1994). The construction of shared knowledge in collaborative problem solving. *NATO ASI Series F Computer and Systems Sciences, 128*, 69–69.

Rogers, C. B., Wendell, K, & Foster, J. (2010). The academic bookshelf: A review of the NAE Report, "Engineering in K-12 education.". *Journal of Engineering Education, 99*(2), 179-181. Retrieved from http://findarticles.com/p/articles/mi_qa3886/is_201004/ai_n53931016/

Rogerson, C., & Scott, E. (2010). The fear factor: How it affects students learning to program in a tertiary environment. *Journal of Information Technology Education: Research, 9*, 147–171. https://doi.org/10.28945/1183

Rudestam, K. E., & Newton, R. R. (2007). *Surviving your dissertation: A comprehensive guide to content and process*. Thousand Oaks, CA: Sage.

Ryan, R. M., & Deci, E. L. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist, 55*, 68–78.

Ryan, R. M., & Deci, E. L. (2020). Intrinsic and extrinsic motivation from a self-determination theory perspective: Definitions, theory, practices, and future directions. *Contemporary Educational Psychology*, (in press), 101860. https://doi.org/10.1016/j.cedpsych.2020.101860

Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. *Computers and Education*, *97*, 129–141. https://doi.org/10.1016/j.compedu.2016.03.003

Saldaña, J. (2016). *The coding manual for qualitative researchers* (3rd ed.). Thousand Oaks, CA: Sage Publications.

Salkind, N. J. (2010). *Encyclopedia of research design* (Vols. 1-0). Thousand Oaks, CA: SAGE Publications, Inc. doi: 10.4135/9781412961288

Sandholtz, J.H., & Ringstaff, C. (2014). Inspiring instructional change in elementary school science: The relationship between enhanced self-efficacy and teacher practices. *Journal of Science Teacher Education, 25*(6), 729-751. https://doi.org/10.1007/s10972-014-9393-0

Scaife, M., & Rogers, Y. (2005). External cognition, innovative technologies, and effective learning. In P. Gardenfors & P. Johansson (Eds.), *Cognition, Education and Communication Technology* (pp. 181-202). Mahwah, NJ: Lawrence Erlbaum Associates.

Schanzer, E. (2015). *Algebraic functions, computer programming, and the challenge of transfer* (Doctoral dissertation). https://doi.org/10.1017/CBO9781107415324.004

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies, 18*(2), 351-380. https://doi.org/10.1007/s10639-012-9240-x

Shafer, D. S. & Zhang, Z. (2012). Introductory statistics. Washington, DC: Saylor Foundation.

Shenton, A.K. (2004). Strategies for ensuring trustworthiness in qualitative research projects. *Education for Information, 22*(2), 63-75.

Sinclair, C. (2008). Initial and changing student teacher motivation and commitment to teaching. *Asia-Pacific Journal of Teacher Education, 36*(2), 79–104. https://doi.org/10.1080/13598660801971658

Sinclair, C., Dowson, M., & McInerney, D. (2006). Motivations to teach: Psychometric and longitudinal perspectives. *Teachers College Record*, *108*(6), 1132–1154.

Singh, K., Granville, M., & Ditka, S. (2002). Mathematics and science achievement: Effects of motivation, interest, and academic engagement. *The Journal of Educational Research*, *95*(6). https://doi.org/10.1080/00220670209596607

Sisman, B., & Kucuk, S. (2019). An educational robotics course: Examination of educational potentials and preservice teachers’ experiences. *International Journal of Research in Education and Science*, *5*(1), 510–531. Retrieved from https://www.ijres.net/index.php/ijres/article/view/505

Skinner, B. F. (1954). The science of learning and the art of teaching. *Harvard Education Review 24*(2) 86–97.

Skinner, E. A., Kindermann, T. A., & Furrer, C. J. (2009). A motivational perspective on engagement and disaffection. *Educational and Psychological Measurement, 69*(3), 493–525. https://doi.org/10.1177/0013164408323233

Smith, M. L. (2013). A case study: Motivational attributes of 4-H participants engaged in robotics. (Doctoral dissertation). Retrieved from ProQuest Dissertation & Theses. (AAT 3558978)

Smith, E. E, Shoben, E. J., & Rips, L. J. (1974). Structure and process in semantic memory: A featural model for semantic decisions. *Psychological Review, 81*, 214-241.

Soloway, E., & Ehrlich, K. (1984). Empirical studies of programming knowledge. *IEEE Transactions on Software Engineering, 10*(5), 595-609.

South Carolina Commission on Higher Education. (2016). *Consideration of new federal Improving Teacher Quality competitive grants awards, FY 2015-16*. Columbia, SC. Retrieved from https://www.che.sc.gov/CHE_Docs/commission%20calendar&materials/2016/February/Agenda_Item_902A.pdf

South Carolina Department of Education. (2017). *South Carolina computer science and digital literacy standards*. Columbia, SC. Retrieved from https://ed.sc.gov/scdoe/assets/File/instruction/standards/Computer%20Science/FINAL_South_Carolina_Computer_Science_and_Digital_Literacy_Standards_(SBEApproved050917)063017.pdf

Staszowski, K., & Bers, M. U. (2005). The effects of peer interactions on the development of technological fluency in an early-childhood, robotic learning environment. In *Proceedings of the American Society for Education Annual Conference & Exposition*. Retrieved from http://labview8.ni.com/pub/devzone/tut/theeffectsofpeer....pdf%5Cnpapers2://publication/uuid/E66F72F2-4596-4B00-81E1-B82EC0FA65F0

Strauss, A., & Corbin, J. (1990). *Basics of qualitative research: Grounded theory procedures and techniques.* Newbury Park, CA: Sage.

Strawhacker, A., & Bers, M. U. (2015). “I want my robot to look for food”: Comparing kindergartner’s programming comprehension using tangible, graphic, and hybrid user interfaces. *International Journal of Technology and Design Education, 25*(3), 293–319. https://doi.org/10.1007/s10798-014-9287-7

Sullivan, F., & Moriarty, M. (2009). Robotics and discover learning: Pedagogical beliefs, teacher practice, and technology integration. *Journal of Technology and Teacher Education, 17*, 109–142. Retrieved from http://people.umass.edu/florence/jtate.pdf%5Cnpapers2://publication/uuid/284416E1-4D1B-48FA-8F07-583B7FCCFA47

Svinicki, M. A. (2010). *Guidebook on conceptual frameworks for research in engineering education*. www.ce.umn.edu/~Smith/docs/RREE-Research-Frameworks Svinicki.pdf.

Sweller, J. (1994). Cognitive load theory, learning difficulty, and instructional design. *Learning and Instruction, 4*, 295–312. https://doi.org/10.1016/0959-4752(94)90003-5

Taylor, F. W. (1916). The principles of scientific management. *Bulletin of the Taylor Society*.

Thompson, G. (2008). Beneath the Apathy. *Educational Leadership, 65*(6), 50–54.

Trees, F. P. (2010). *A meta-analysis of pedagogical tools used in introductory programming courses* (Doctoral dissertation). Retrieved from http://ezproxy.library.nyu.edu:2148/pqdtft/docview/305239837/abstract/138B68F802A21839DB5/12?accountid=12768%5Cn

Tsay, C. H., Kofinas, A. K., Trivedi, S. K., & Yang, Y. (2019). Overcoming the novelty effect in online gamified learning systems: An empirical evaluation of student engagement and performance. *Journal of Computer Assisted Learning, 36*(2), 128-146.

Ucgul, M. (2013). History and educational potential of Lego Mindstorms NXT. *Mersin University Journal of the Faculty of Education, 9*(2), 127–137.

United States Department of Labor (2018). *Employment projections*. Washington DC: Government Printing Office. Retrieved from https://www.bls.gov/emp/tables/fastest-growing-occupations.htm

Vessey, I. (1985). Expertise in debugging computer systems: A process analysis. *International Journal of Man–Machine Studies, 23*(5), 459–494. https ://doi.org/10.1016/S0020 -7373(85)80054 -7.

Vollmeyer, R. R., & Rheinberg, F. (2006). Motivational effects on self-regulated learning with different tasks. *Educational Psychology Review, 18*, 239–253.

Vygotsky, L. S. (1980). *Mind in society: The development of higher psychological processes*. Cambridge, MA: Harvard University Press.

Wang, E. (2001). Teaching freshmen design, creativity and programming with LEGOs. *Proceedings of the 31st ASEE/IEEE Frontiers in Education Conference*. Reno, NV, October 10-13, 2001. Retrieved from http://fie.engrng.pitt.edu/fie2001/papers/1291.pdf

Wang, T. C., Mei, W. H., Lin, S. L., Chiu, S. K., & Lin, J. M. C. (2009). Teaching programming concepts to high school students with Alice. *Proceedings - Frontiers in Education Conference, FIE*. https://doi.org/10.1109/FIE.2009.5350486

Wang, X. C., & Ching, C. C. (2003). Social construction of computer experience in a first-grade classroom: Social processes and mediating artifacts. *Early Education and Development*, *14*(3), 335-361.

Weintrop, D. (2016). *Modality matters: Understanding the effects of programming language representation in high school computer science classrooms* (Doctoral dissertation). Retrieved from ProQuest Dissertation & Theses. (AAT 10160575)

Weintrop, D., & Wilensky, U. (2015). To block or not to block, that is the question: Students' perceptions of blocks-based programming. *Proceedings for the 14th International Conference on Interaction Design and Children*, (2), 199–208. https://doi.org/http://dx.doi.org/10.1145/2771839.2771860

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

Werner, L., Campe, S., & Denner, J. (2012). Children learning computer science concepts via Alice game-programming. *SIGSCE '12 Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, 427–432. https://doi.org/10.1145/2157136.2157263

Wiedenbeck, S. (2005). Factors affecting the success of non-majors in learning to program. In *Proceedings of the Conference on International Computing Education Research (ICER'05)*. 13-24.

Wigfield, A., & Eccles, J. S. (2000). Expectancy–value theory of achievement motivation. *Contemporary Educational Psychology, 25,* 68–81. doi:10.1006/ceps.1999.1015

Willems, P., & Gonzalez-DeHass, A. R. (2012). School–community partner- ships: Using authentic contexts to academically motivate students. *School Community Journal, 22*(2), 9–30.

Wilson, A., & Moffat, D. C. (2010). Evaluating Scratch to introduce younger schoolchildren to programming. *Proceedings of the 22nd Annual Workshop of the Psychology of Programming Interest Group*, 64–75.

Wilson, C., Sudol, L., Stephenson, C., & Stehlik, M. (2010). Running on empty: The failure to teach K-12 computer science in the digital age. *The Association for Computing Machinery*. https://doi.org/10.1353/hpu.2010.0941

Witney, D., & Smallbone, T. (2011). Wild work: Can using wilds enhance student collaboration for group assignment tasks? *Innovations in Education and Teaching International, 48*(1), 101–110. doi:10. 1080/14703297.2010.54376.

Yadav, A., Good, J., Voogt, J., & Fisser, P. (2017). Computational thinking as an engineering competence domain. In M. Mulder (Ed.), *Competence-based vocational and professional education* (pp. 1051-1067). https://doi.org/10.1007/978-3-319-41713-4_51

Yadav, A., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education*, *14*(1), 1–16. https://doi.org/10.1145/2576872

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2011). Introducing computational thinking in education courses. *Proceedings of the 42nd ACM technical symposium on computer science education*. https://doi.org/10.1145/1953163.1953297

Yamazaki, S., Sakamoto, K., Honda, K., Washizaki, H., & Fukazawa, Y. (2015). Comparative study on programmable robots as programming education tools. *ACE 2015*.

Yonghiu, C. (2010). Study of flow theory and experiential learning. *Proceedings of the 2nd International Conference on Multimedia and Information Technology (MMIT'10)*. 2, 334–337.

Zeni, J. (1998). A guide to ethical issues and action research. *Educational Action Research, 6*(1), 9–19. https://doi.org/10.1080/09650799800200053

APPENDIX A

ROBOTICS LESSON PLANS, SCHEMATICS, AND EXAMPLES

Lesson Plan: Basic Procedures Class 1	
SC State Computer Science Standards	• Standard 1: Recognize that many daily tasks can be described as step-by-step instructions (i.e., algorithms). • Standard 4: Develop a program to express an idea or address a problem
EDUC 204 Student Learning Outcome	• 1) Demonstrate understanding of technology concepts, tools, systems and operations to enhance teaching practice, professional productivity, and student performance.
Objectives	• Students will be able to test and debug a program • Students will be able to create a functioning program
Materials	Lego EV3 robot; computer; Lego EV3 programming software
Procedures	The class will begin with a demonstration of how to use the Basic Procedures programming blocks. Special attention will be paid to demonstrating how to update each of the programming blocks for number of rotations, degrees, or running for a specific number of seconds. How to program the robots to turn will also be demonstrated. The debugging process will be demonstrated to help participants for when they encounter errors. Participants will be paired and given a pre-built Lego robot and a laptop with the programming software. Pairs of participants will experiment with programming the robots. Participants will be instructed to rotate the robot and programming hands-on time between each member of the pair so that all pairs receive hands-on time programming the robot. The instructor will provide scaffolding as needed and will assist with debugging. As an exit ticket to finish the class, participants will be asked to share one discovery they have made as a result of their free time.
Exit Ticket	• Share one discovery groups have made while programming their robots

Figure A.1. The lesson plan for Basic Procedures class one.

Lesson Plan: Basic Procedures Class 2	
SC State Computer Science Standards	• Standard 1: Recognize that many daily tasks can be described as step-by-step instructions (i.e., algorithms). • Standard 4: Develop a program to express an idea or address a problem.
EDUC 204 Student Learning Outcome	• 1) Demonstrate understanding of technology concepts, tools, systems and operations to enhance teaching practice, professional productivity, and student performance.
Lesson Objectives	• Students will be able to calculate values for a program • Students will be able to use different methods of programming to solve a problem
Materials	Lego EV3 robot; computer; Lego EV3 programming software; meter stick or pre-measured one meter of electrical tape
Procedures	To begin, pairs will be instructed on odometry and how teachers can use odometry in the classroom. Pairs will be given a pre-built Lego robot and a laptop with the programming software. The instructor will explain to the pairs that the robots can record how far the robots have travelled. The robots can record how many degrees the wheels have rotated. Using this data pairs will calculate how far each wheel rotation moves the robot. Pairs will then calculate how far each rotation moves their robots. Then, the One Meter Challenge will be introduced. Pairs will be challenged to program their robots to travel one meter in three different ways. The first way pairs can program their robots to move one meter is by using the move steering program block and customizing the number of rotations to their calculated odometer length. The second way pairs can program their robots is by a total number of degrees based on their calculations. The third way is that pairs can program their robots to move at a certain power for a certain number of seconds to reach one meter. The instructor will roam the room and provide scaffolding as needed.

Figure A.2. The lesson plan for Basic Procedures class two.

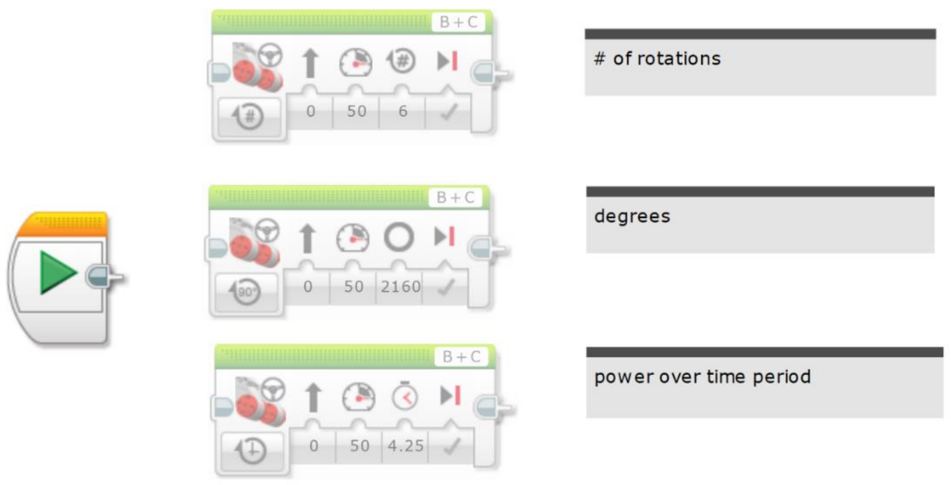


Figure A.3. A possible solution for the One Meter Challenge.

Lesson Plan: Advanced Procedures Class 1	
SC State Computer Science Standards	• Standard 1: Design, evaluate, and modify simple algorithms (e.g., steps to make a sandwich; steps to a popular dance; steps for sending an email). • Standard 3: Decompose problems into subproblems and write code to solve the subproblems (i.e., break down a problem into smaller parts).
EDUC 204 Student Learning Outcome	• 1) Demonstrate understanding of technology concepts, tools, systems and operations to enhance teaching practice, professional productivity, and student performance.
Lesson Objectives	• Students will be able to predict the outcome of a program • Students will be able to modify a simple program
Materials	Paper; pencil; Lego EV3 robot; computer; Lego EV3 programming software; Lego EV3 box
Description	The instructor will start by introducing turning to participants. After demonstrating how to program turns, the instructor will demonstrate how to write pseudocode and how teachers can use pseudocode in the classroom. Then, the instructor will introduce the lap activity. Participants will be divided into pairs and will be given a pre-built Lego robot and a laptop with the programming software. For the lap activity, pairs will be challenged to modify a given program so that their robots move around the box their robot came in. The robots must complete one full lap around the box without touching the box or straying outside of one foot from the box. Pairs will note that not all turns will be accurate due to friction and grip. The instructor will provide scaffolding as needed. Pairs will be instructed to make sure they save their Lap Activity programs, because they will be used again later.

Figure A.4. The lesson plan for Advanced Procedures class one.

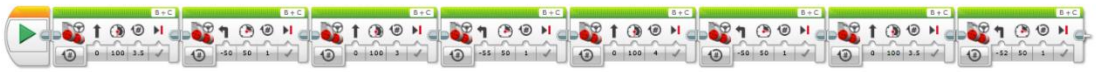


Figure A.5. A possible solution for the Lap Activity.

Lesson Plan: Advanced Procedures Class 2	
SC State Computer Science Standards	• Standard 1: Design, evaluate, and modify simple algorithms (e.g., steps to make a sandwich; steps to a popular dance; steps for sending an email). • Standard 3: Decompose problems into subproblems and write code to solve the subproblems (i.e., break down a problem into smaller parts).
EDUC 204 Student Learning Outcome	• 1) Demonstrate understanding of technology concepts, tools, systems and operations to enhance teaching practice, professional productivity, and student performance.
Lesson Objectives	• Students will be able to predict the outcome of a program • Students will be able to create a program to solve a problem
Materials	Paper; pencil; Lego EV3 robot; computer; Lego EV3 programming software; maze made of electrical tape
Procedures	The class will begin with another pseudocode demonstration and activity. Pseudocode will be reviewed. Participants will divide into pairs of four students and will be given a pre-built Lego robot and a laptop with the programming software. Then, pairs will write their pseudocode for navigating a maze. After pairs have created their pseudocode for navigating the maze, the pairs will translate their pseudocode into programming to solve the Maze Challenge. Six identical mazes will be marked off with black electrical tape on the floor throughout the classroom for efficiency in order to provide ample opportunity for pairs to test their programming. The robots should not touch the lines as they navigate the maze. If pairs complete the maze successfully in before the class period is over, they will be invited to try to solve the maze from the unmarked corner to the other unmarked corner in a much more difficult programming challenge. The instructor will provide scaffolding as needed.

Figure A.6. The lesson plan for Advanced Procedures class two.

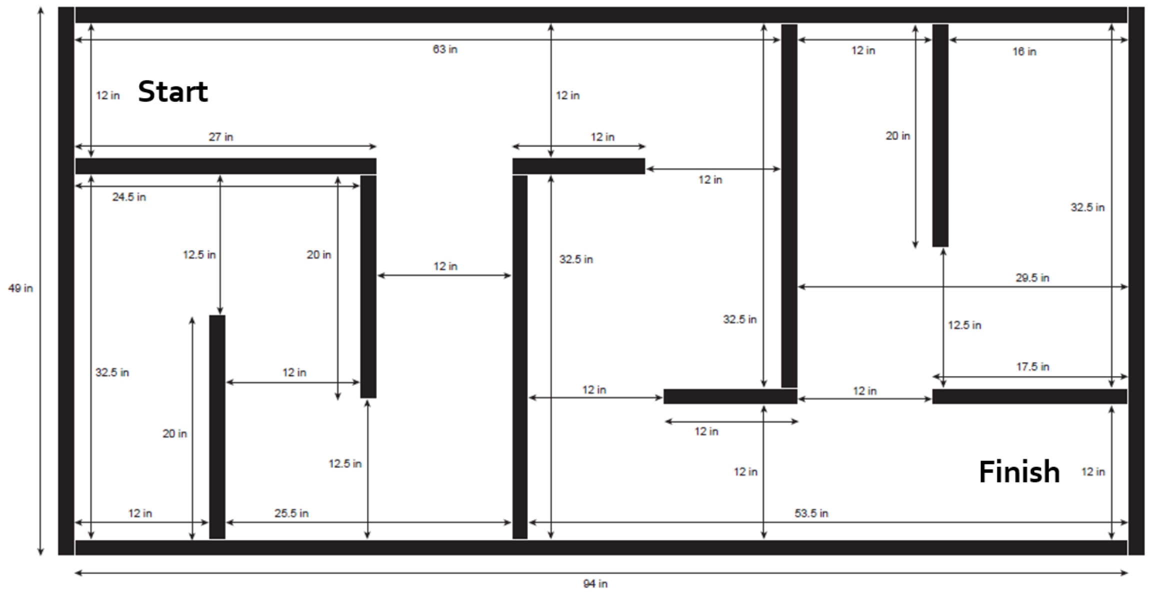


Figure A.7. The schematic for the Maze Challenge. Plans for this maze are derived from the Coastal Robotics curriculum.

Lesson Plan: Control Structures Class 1	
SC State Computer Science Standards	• Standard 2: Use and compare simple coding control structures (e.g., if-then, loops).
EDUC 204 Student Learning Outcome	• 1) Demonstrate understanding of technology concepts, tools, systems and operations to enhance teaching practice, professional productivity, and student performance.
Lesson Objectives	• Students will be able to predict the outcome of a program that uses control structures • Students will be able to create a program using control structures
Materials	Lego EV3 robot; computer; Lego EV3 programming software; meter stick or pre-measured one meter of electrical tape
Procedures	To begin, pairs will be instructed on control structures and how teachers can use wait, switch, and looping concepts to teach basic computer programming concepts in the classroom. Participants will predict the outcomes of the demonstrated programs. The instructor will then demonstrate how to write a program using each of the control structures. Participants will divide into pairs and will be given a pre-built Lego robot and a laptop with the programming software. After that, the instructor will introduce the Slithering One Meter Activity. Pairs will then begin programming their robots to complete the activity. Pairs will test their programs against either a meter stick or a pre-cut line of tape measuring one meter. The instructor will provide scaffolding as needed.

Figure A.8. The lesson plan for Control Structures class one.

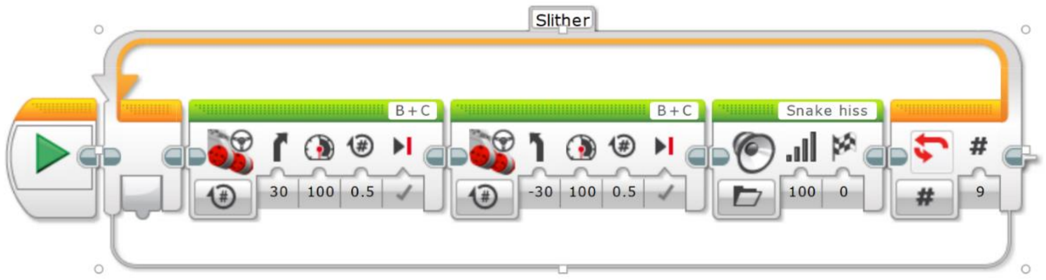


Figure A.9. A possible solution for the Slithering One Meter Challenge.

Lesson Plan: Control Structures Class 2	
SC State Computer Science Standards	• Standard 2: Use and compare simple coding control structures (e.g., if-then, loops).
EDUC 204 Student Learning Outcome	• 1) Demonstrate understanding of technology concepts, tools, systems and operations to enhance teaching practice, professional productivity, and student performance.
Lesson Objectives	• Students will be able to modify a simple program using control structures • Students will be able to create a program using control structures
Materials	Lego EV3 robot; computer; Lego EV3 programming software; Lego EV3 box
Description	To begin, pairs will be instructed on looping and how teachers can use looping to teach basic computer programming concepts in the classroom. The instructor will then demonstrate how to write a loop in the programming editor. After that, the instructor will introduce the Lap Loop Challenge. For this activity, pairs must modify their Lap Activity programs utilizing loops. Participants will divide into pairs and will be given a pre-built Lego robot and a laptop with the programming software. Pairs will then begin by modifying their programs from the Lap Activity. Pairs will test their programs around their robots' boxes. The instructor will provide scaffolding as needed and remind the students that the straight and turn commands need to be looped to complete one lap before playing a sound.

Figure A.10. The lesson plan for Control Structures class two.

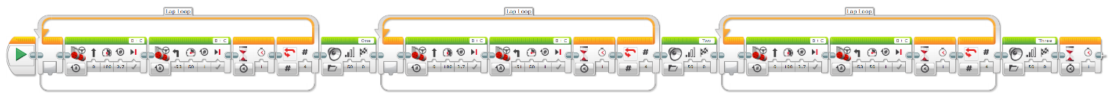


Figure A.11. A possible solution to the Lap Loop Challenge.

Lesson Plan: Variables Class 1	
SC State Computer Science Standards	• Standard 5: Identify variables and compare the types of data stored as variables.
EDUC 204 Student Learning Outcome	• 1) Demonstrate understanding of technology concepts, tools, systems and operations to enhance teaching practice, professional productivity, and student performance.
Lesson Objectives	• Students will be able to predict the outcome of a program based on the given variables. • Students will be able to create a program using variables.
Materials	Lego EV3 robot; computer; Lego EV3 programming software; blue tape; red tape
Procedures	The class will begin with the instructor demonstrating variables and explaining how teachers can use variables in their curricula. Participants will predict the outcomes of programs based on example variables. The instructor will demonstrate how to write a program with variables in the programming editor. The instructor will also demonstrate to participants how the color sensor works. Participants will divide into pairs and will be given a pre-built Lego robot and a laptop with the programming software. In the Red Light learning activity, pairs will have to program their robots to speed up when the color sensor detects blue and stop when the color sensor detects red. This programming will involve the switch and a speed variable. The instructor will walk around the room and provide scaffolding as needed.

Figure A.12. The lesson plan for Variables class one.

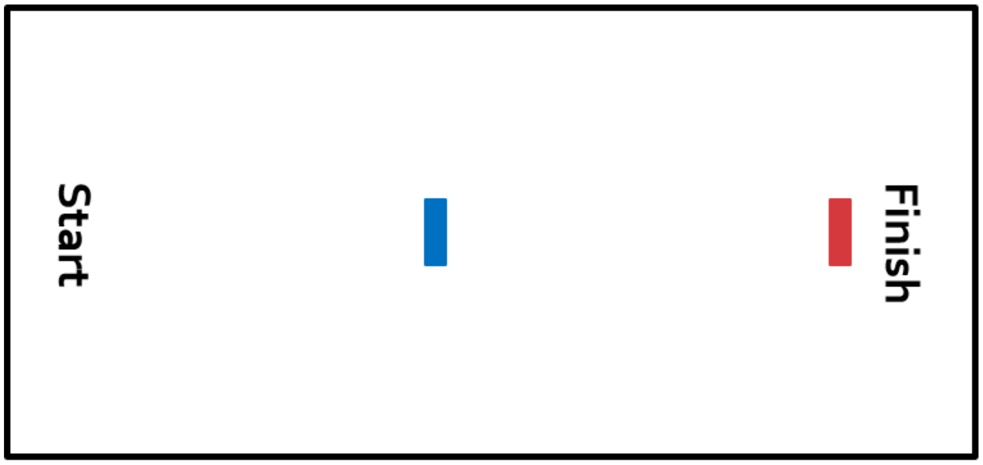


Figure A.13. A schematic of the Red Light Activity.

Lesson Plan: Variables Class 2	
SC State Computer Science Standards	• Standard 4: Develop a program to express an idea or address a problem. • Standard 5: Identify variables and compare the types of data stored as variables.
EDUC 204 Student Learning Outcome	• 1) Demonstrate understanding of technology concepts, tools, systems and operations to enhance teaching practice, professional productivity, and student performance.
Lesson Objectives	• Students will be able to create a program using variables. • Students will be able to modify a program using variables.
Materials	Lego EV3 robot; computer; Lego EV3 programming software; maze made of black, blue, and red tape
Procedures	Participants will divide into pairs and will be given a pre-built Lego robot and a laptop with the programming software. The Maze with Variables challenge will then be introduced to students. For the challenge, pairs will be instructed to utilize the wait block and the if/then statement block under the Flow Control heading as well as the variable block and the math block under the Data Operations heading. These functions will be reviewed. For this challenge, the mazes utilized in the Maze challenge will be modified. Green pieces of tape will be added to the mazes at points where the robots would need to turn left. Red pieces of tape will be added for spots where the robots should turn right. Every time the robots encounter a green line, they will turn left and execute the math sequence of x + 1 to count the turn on the EV3's screen. The robots should stop when they detect the black tape to keep the robots from leaving the maze and stopping at the finish. Pairs will complete this challenge when they successfully navigate their robots to the end of the maze using programming which utilizes movement, control structures, and variables.

Figure A.14. The lesson plan for Variables class two.

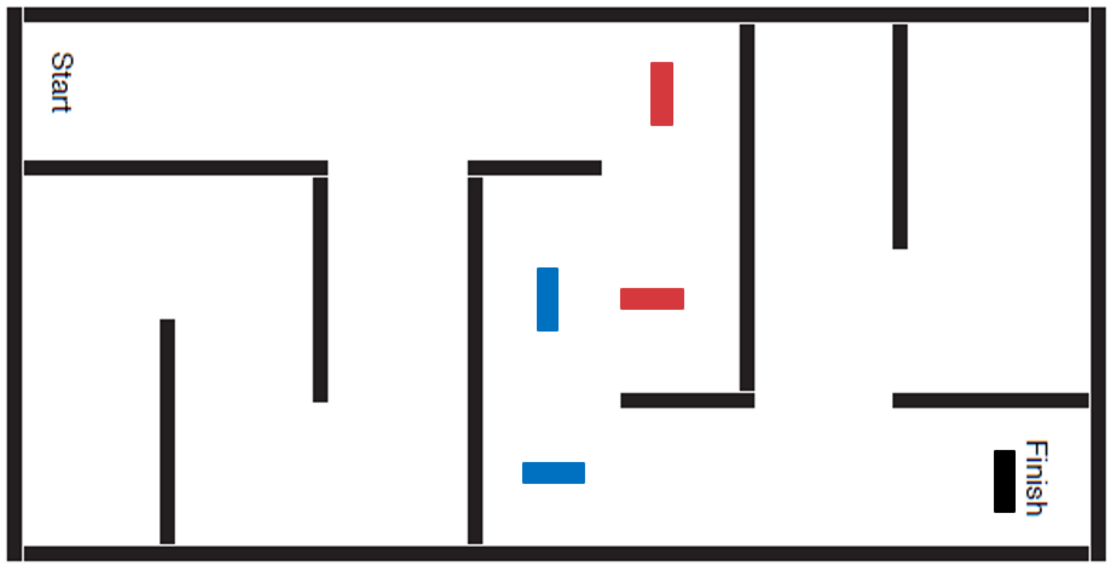


Figure A.15. A schematic for the Maze with Variables Challenge. This maze is derived from the Coastal Robotics curriculum.

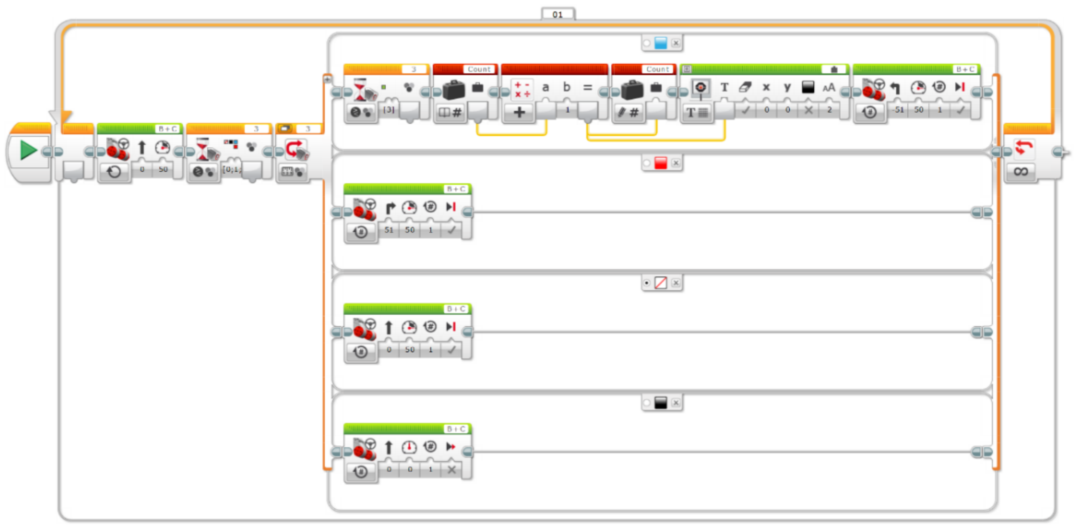


Figure A.16. A possible solution for the Maze with Variables Challenge.

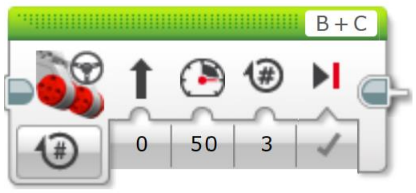
APPENDIX B

PROGRAMMING COMPREHENSION ASSESSMENT

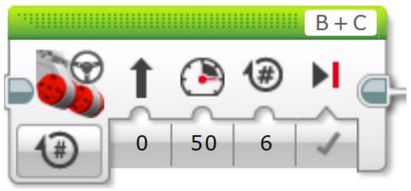
Basic Procedures

1. If one meter is equal to 2160° of turning on a wheel, which block set to number of rotations will move the robot half a meter?

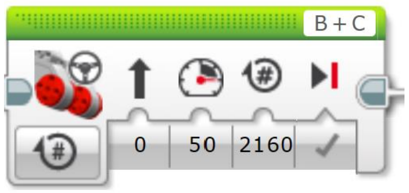
a.



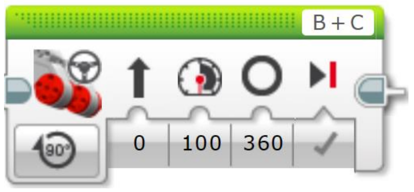
b.



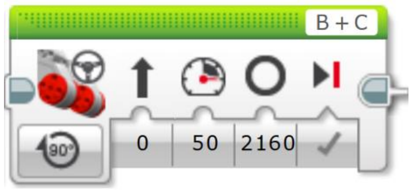
c.



d.

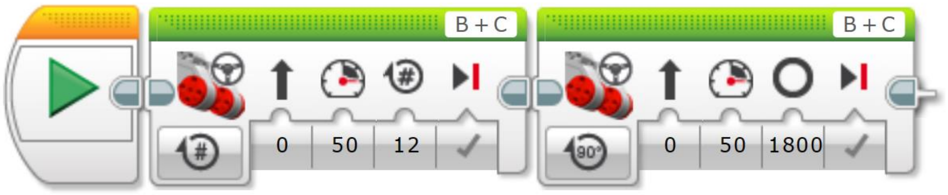


e.

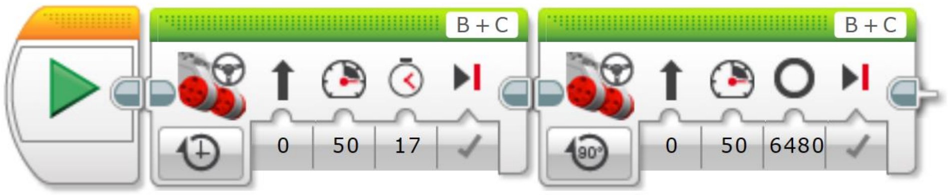


2. If (1 meter = 6 rotations = 2160º = 4.25 seconds) at 50% power, which of these programs will move the robot exactly 7 meters?

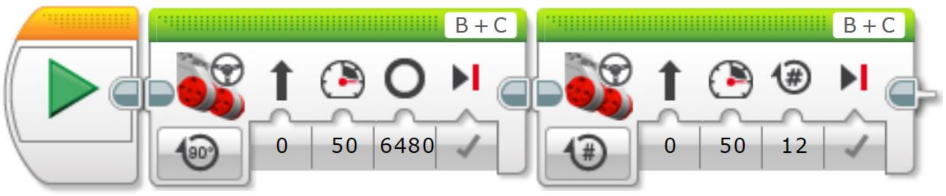
a.



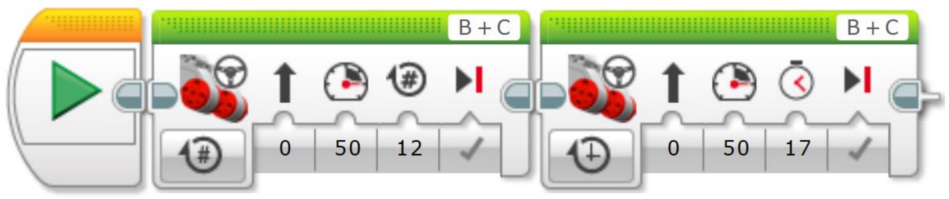
b.



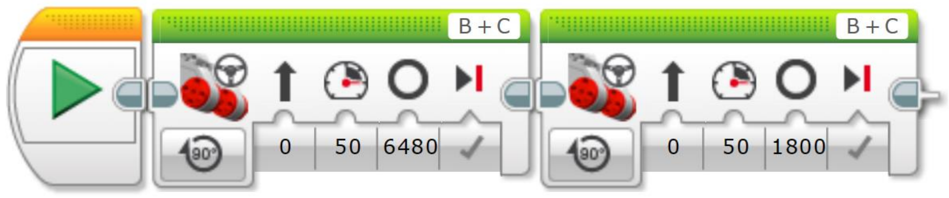
c.



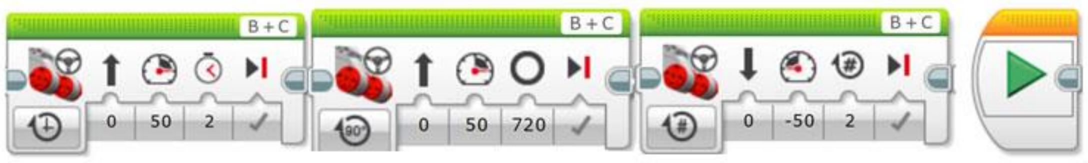
d.



e.



3. Arrange these pieces so the resulting program is executable and moves the robot forward for two seconds, backward for two rotations, and then forward for 720 degrees.



i ii iii iv

a. i, ii, iii, iv

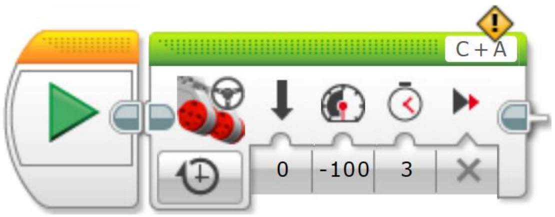
b. ii, i, iv, iii

c. iii, ii, i, iv

d. iv, iii, i, ii

e. iv, i, iii, ii

4. How would you debug the block of programming below so that the robot moves backward for three seconds at 100% power and then coasts?



a. Update the power to 100%.

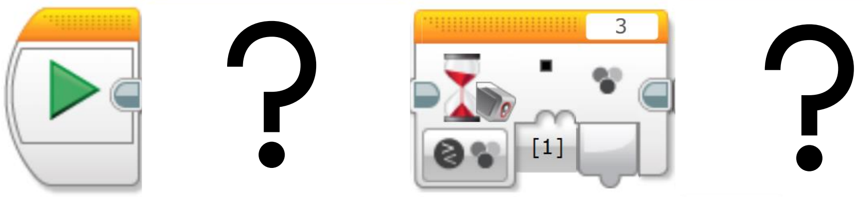
b. Update the time to 0.03.

c. Update brake at end to true.

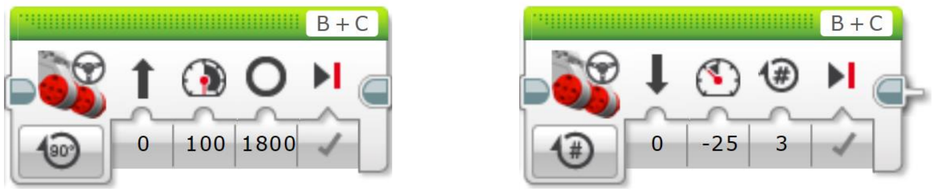
d. Update the ports for the proper move steering motor.

e. All of these.

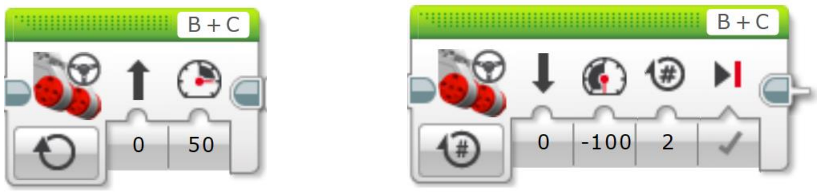
5. Which of these movement blocks would you add to build a program which moves the robot forward until it encounters a black line, then it backs up?



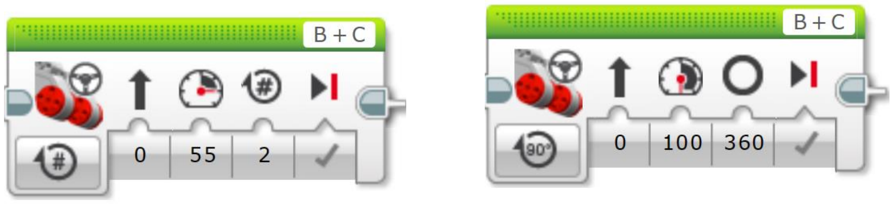
a.



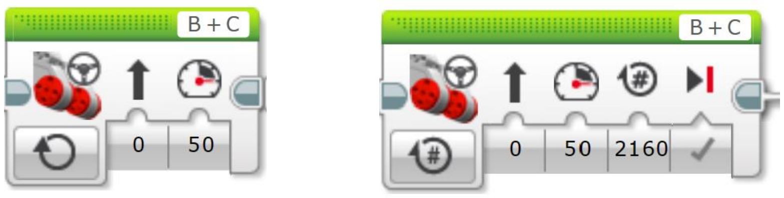
b.



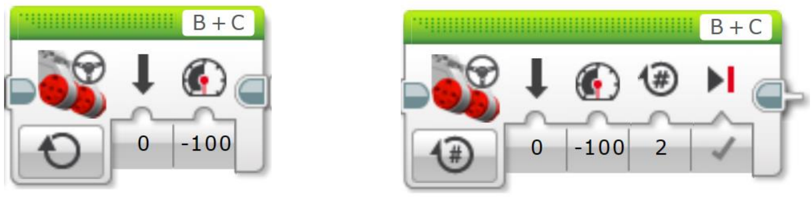
c.



d.

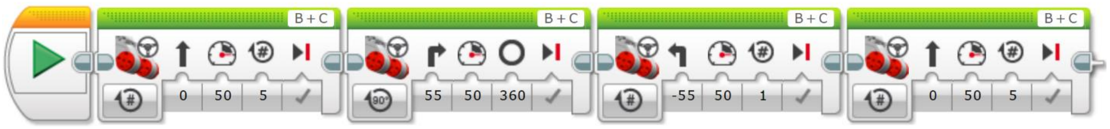


e.



Advanced Procedures

6. Where would a robot running this programming finish at the end of the program?



a. To the left of the starting position.

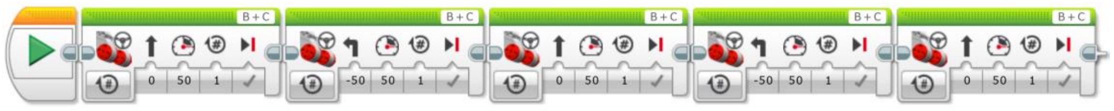
b. To the right of the starting position.

c. Directly in front of the starting position.

d. Directly behind the starting position.

e. At the exact same point as the starting position.

7. Where would a robot running this programming finish at the end of the program?



a. To the left of the starting position.

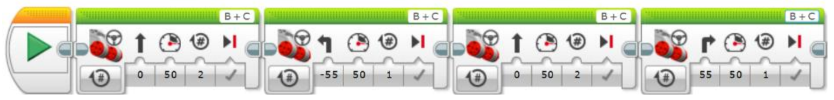
b. To the right of the starting position.

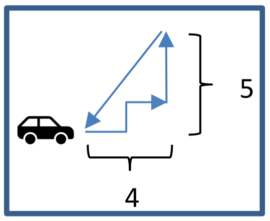
c. Directly in front of the starting position.

d. Directly behind the starting position.

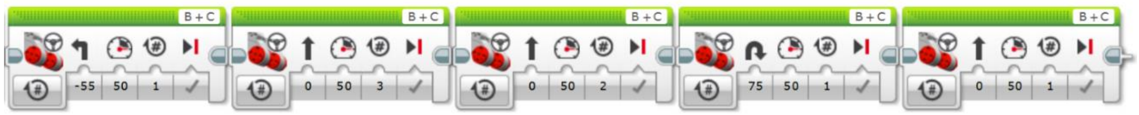
e. At the exact same point as the starting position.

8. Finish the program with the arranged segments to perform the action on the diagram.

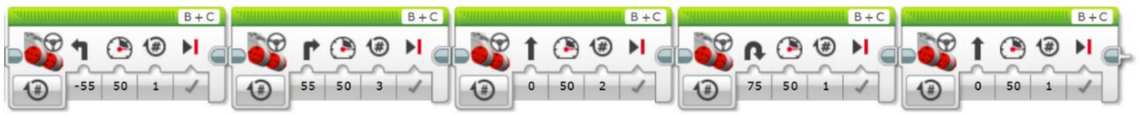




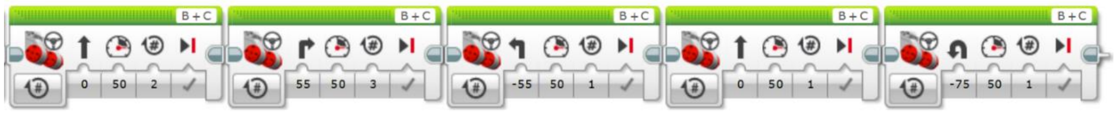
a.



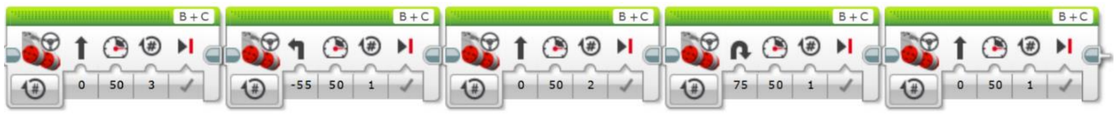
b.



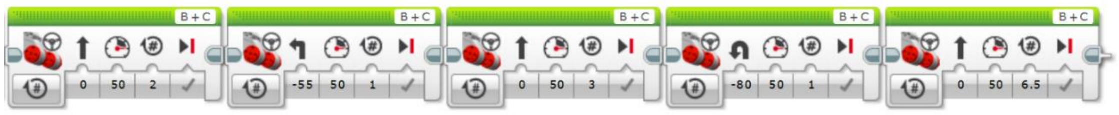
c.



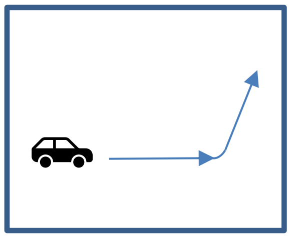
d.



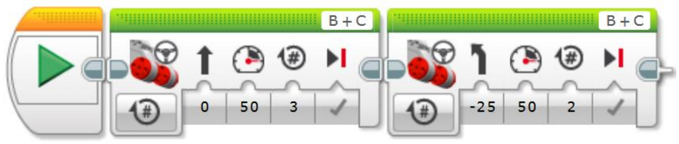
e.



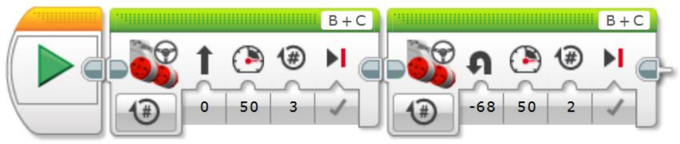
9. Identify the program designed to perform the action in the diagram.



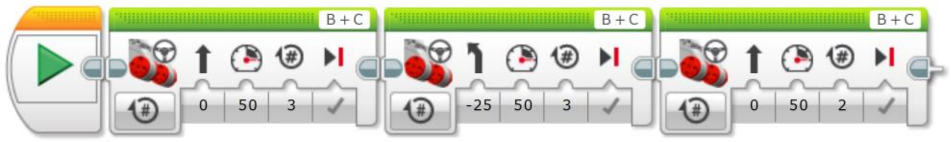
a.



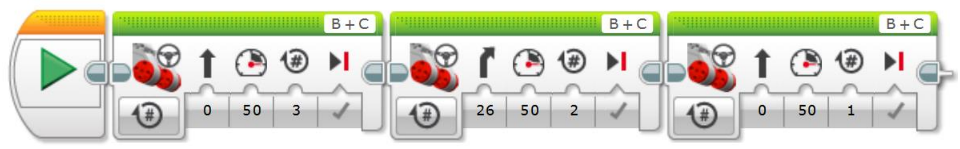
b.



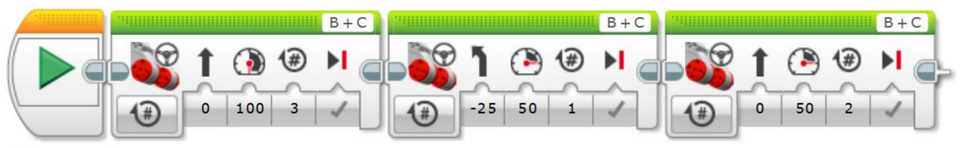
c.



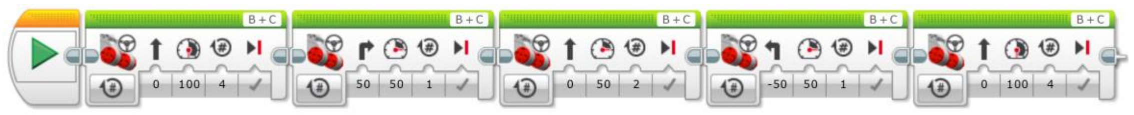
d.

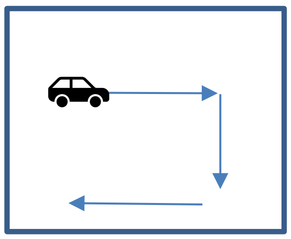


e.



10. Your friend writes a program to move a car in a backward C shape, but the program keeps moving in an S shape. Identify which segment is incorrectly programmed.





a. The first movement block.

b. The second movement block.

c. The third movement block.

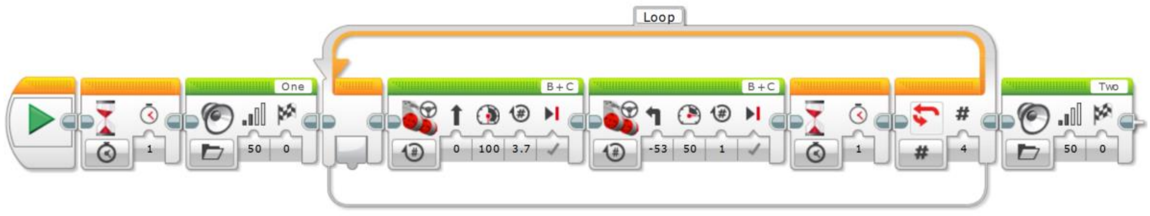
d. The fourth movement block.

e. The fifth movement block.

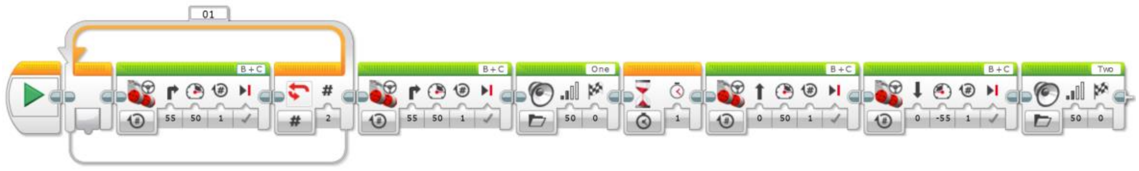
Control Structures

11. Which of the following loop sequences will say a different word after ever four turns?

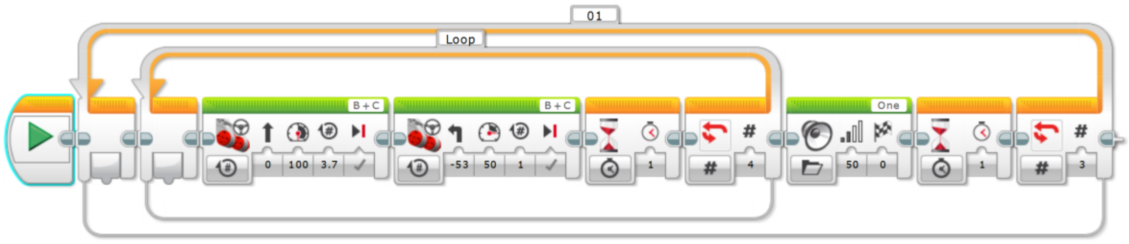
a.



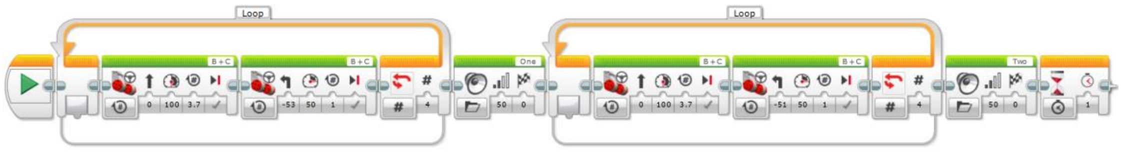
b.



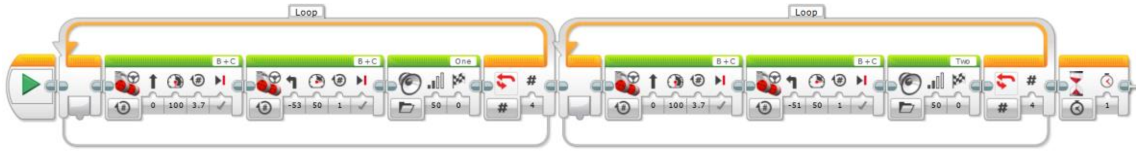
c.



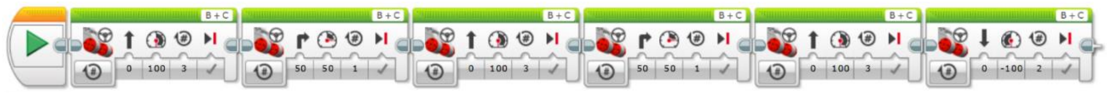
d.



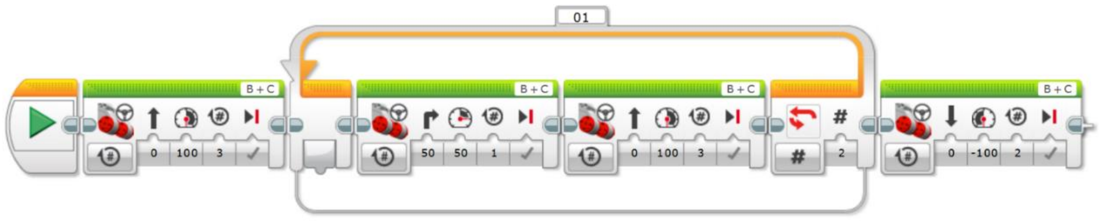
e.



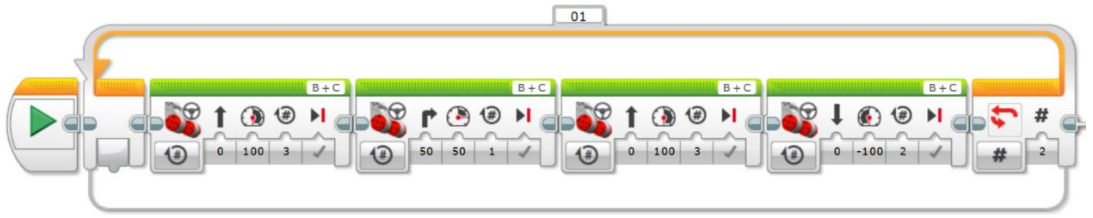
12. Which loop option simplifies this program?



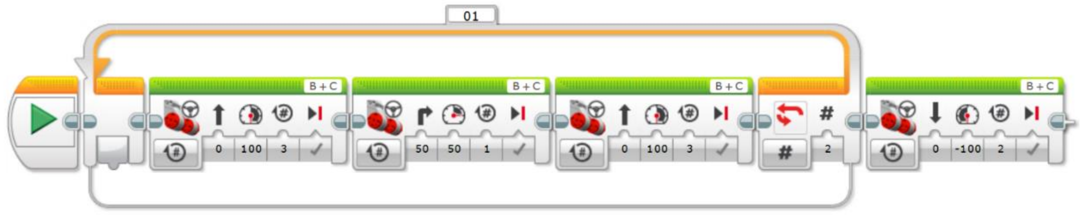
a.



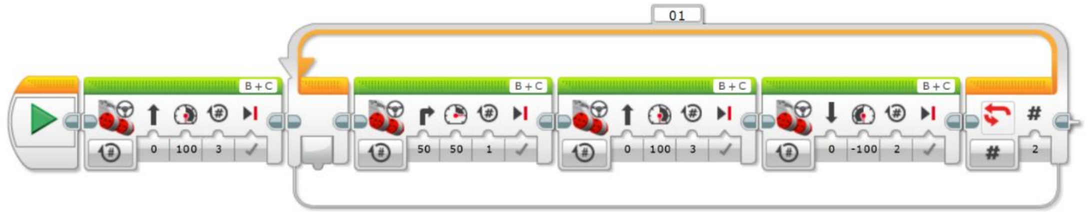
b.



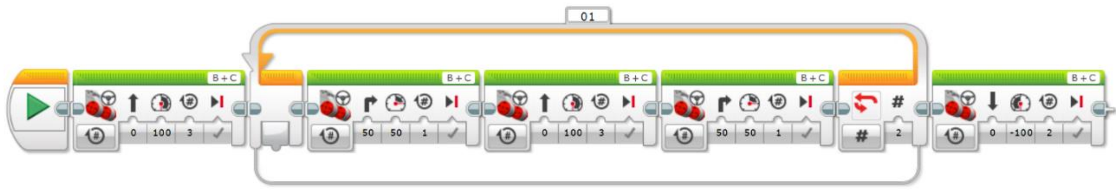
c.



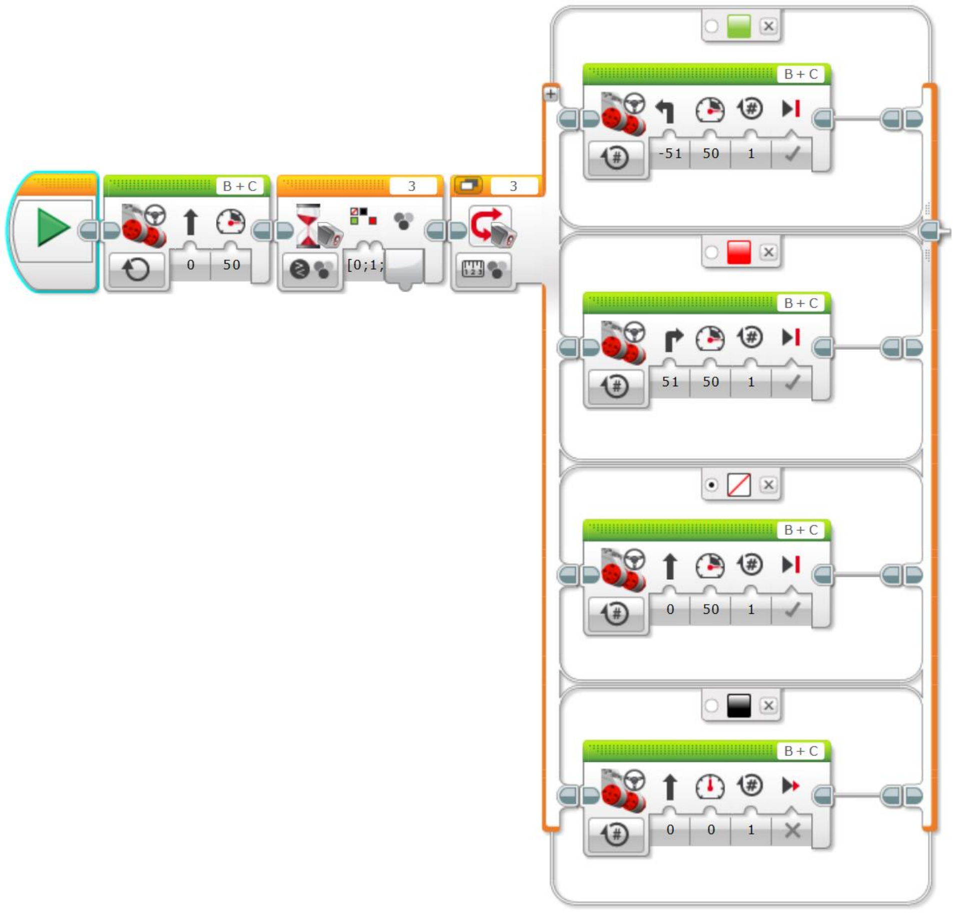
d.



e.



13. Given the conditional if/then statement, what will happen if the robot detects black?



a. It will turn left.

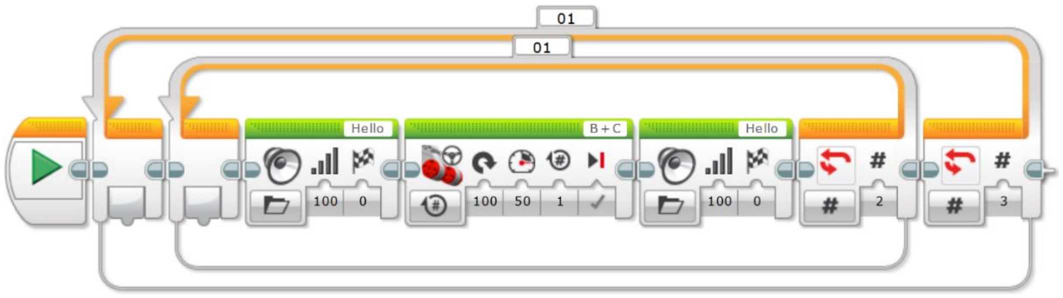
b. It will turn right.

c. It will continue moving straight until it detects either black, blue, or green.

d. It will continue straight for one rotation.

e. It will stop.

14. How many times will the following program say “hello” before ending?



a. 2

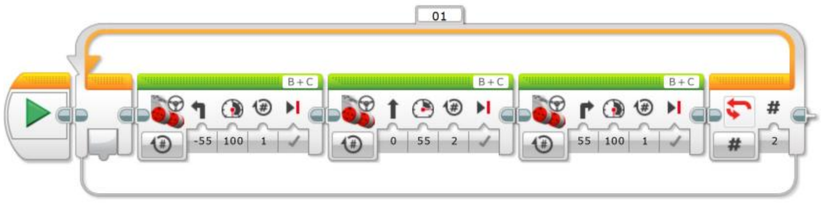
b. 3

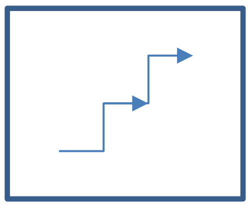
c. 5

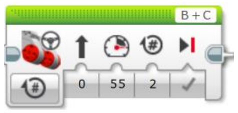
d. 6

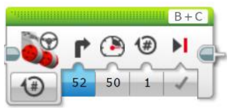
e. 12

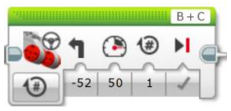
15. Finish creating an algorithm so that the car moves in the pattern on the ground as demonstrated in the graphic on the right.





a. Place  before the first programming block inside the loop.

b. Place  after the first programming block inside the loop.

c. Place  after the last programming block inside the loop.

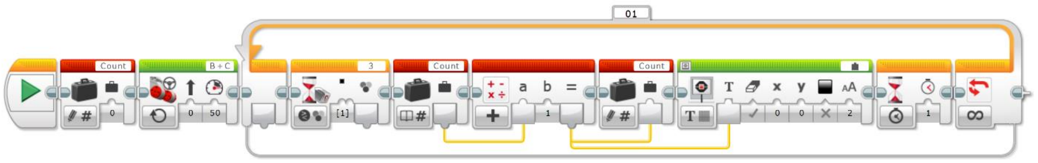
d. Swap each of the turn blocks within the algorithm.

e. Change the loop count from 2 to 4.

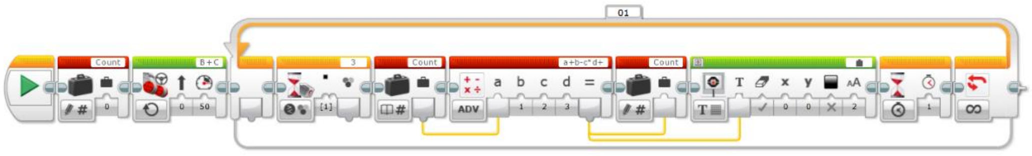
Variables

16. Which algorithm counts each black line it encounters forever?

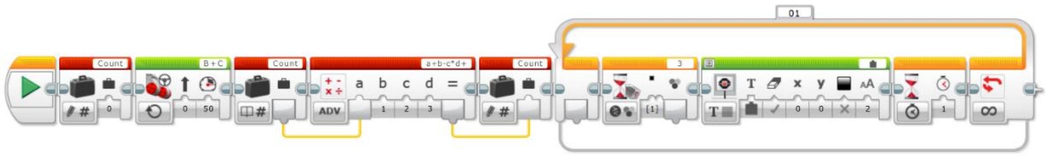
a.



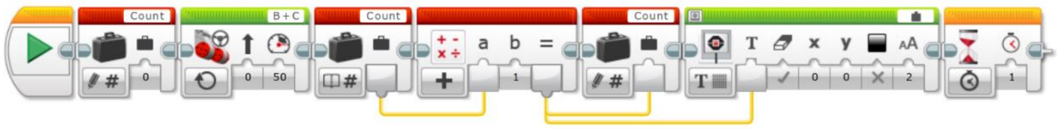
b.



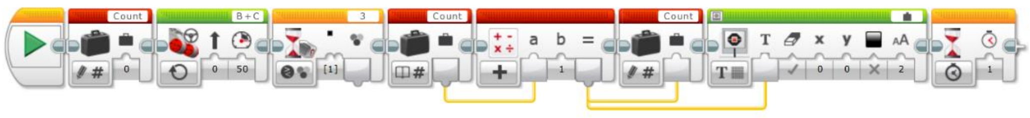
c.



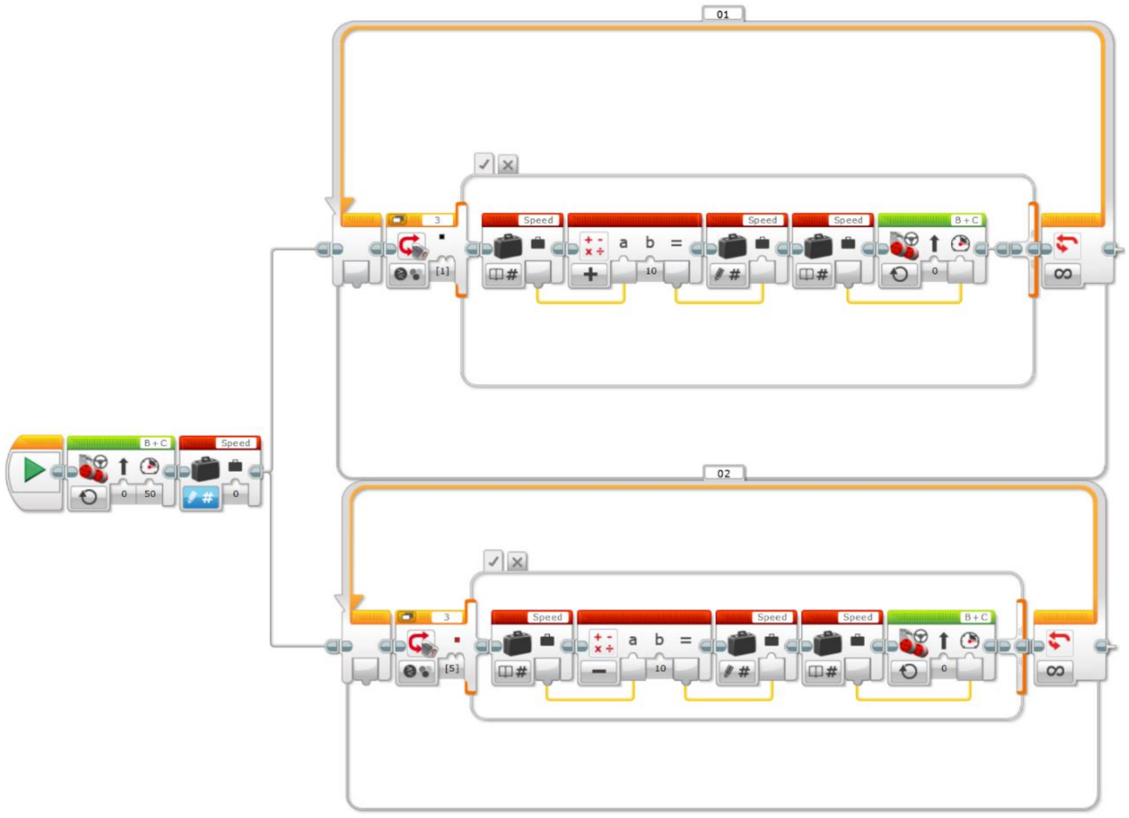
d.



e.



17. Given the pictured conditional if/then statement, what will happen each time the robot detects a black line?



a. It will speed up 10 power up to a maximum of 100 power.

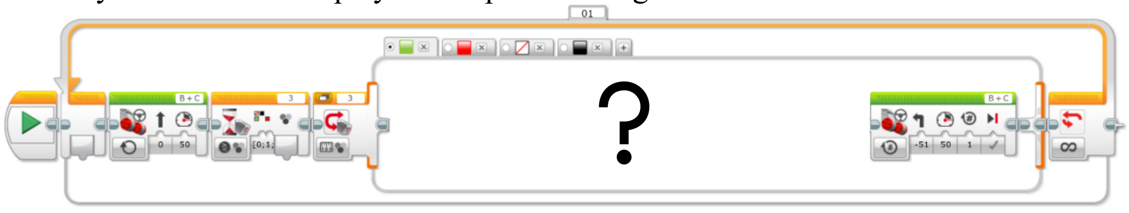
b. It will slow down 10 power up to a maximum of -100 power.

c. It will count by positive 10.

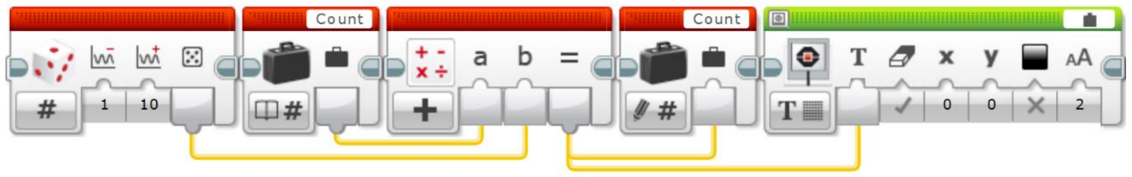
d. It will count by negative 10.

e. It will reverse.

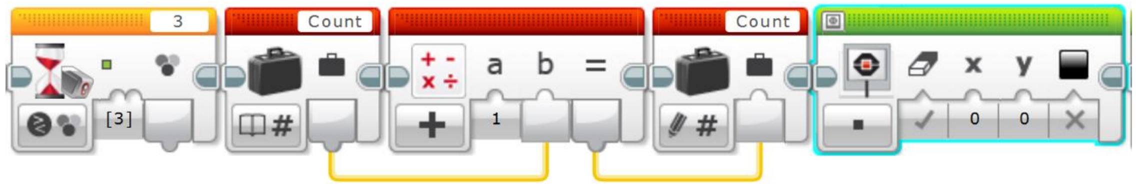
18. Your friend is building an algorithm which will increment a variable by one and turn left at each green line encountered. Choose the string of programming in which the variable increases by one at each green line encountered and displays the updated count textually on the EV3's display to complete this algorithm.



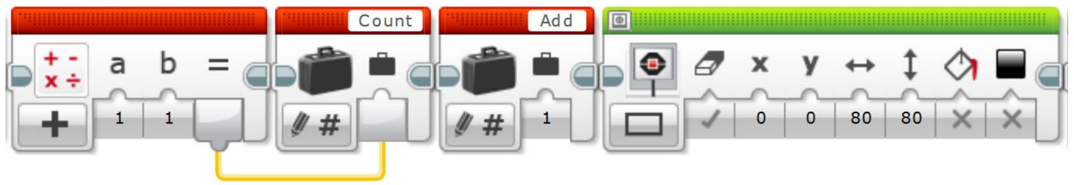
a.



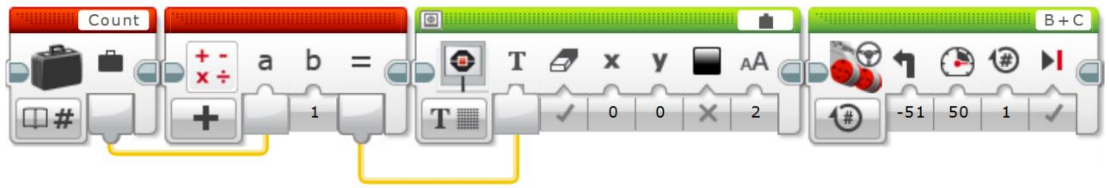
b.



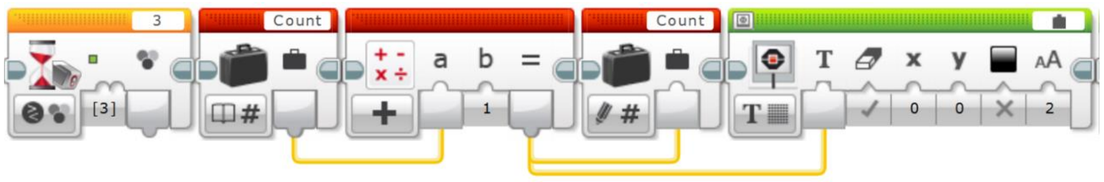
c.



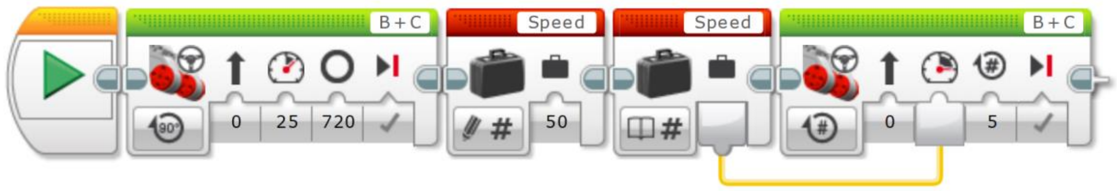
d.



e.



19. Given the variable, what will this program do?



a. Move at a power of 25 for 720º and then move at a power of 1 forever after that.

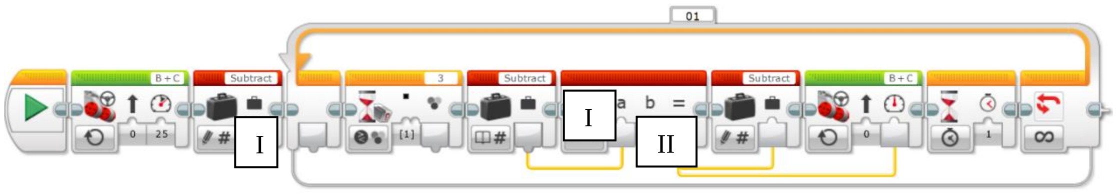
b. Move at a power of 25 for two rotations and then move 5 rotations at a power of 50.

c. Not move.

d. Move at a power of 50 for two rotations and then move at a power of 25 for 5 rotations.

e. Move at a power of 50 for 720 º and then slow down to a power of 1 for 5 rotations.

20. Create a program with a variable value of 25 which will subtract 15 power from the motor for each line it encounters.



a. (I) 25; (II) Subtract; (III) 25

b. (I) 15; (II) Add; (III) 15

c. (I) 25; (II) Add; (III) 15

d. (I) 15; (II) Subtract; (III) 25

e. (I) 25; (II) Subtract; (III) 1

APPENDIX C

COMPREHENSION ASSESSMENT ALIGNMENT TABLES

Table C.1. *Basic Procedures Assessment Question, Lesson Objective, and SC State Computer Science Standard Alignment*

Assessment Question	Lesson Objective	Computer Science Standard
1. If one meter is equal to 2160^o of turning on a wheel, which block set to number of rotations will move the robot half a meter?	Calculate odometry for a program	Standard 4: Develop a program to express an idea or address a problem
2. If (1 meter = 6 rotations = 2160^o = 4.25 seconds) at 50% power, which of these programs will move the robot exactly 7 meters?	Use different ways to program a robot to move a given distance	Standard 4: Develop a program to express an idea or address a problem
3. Arrange these pieces so the resulting program is executable and moves the robot forward for two seconds, backward for two rotations, and then forward for 720 degrees.	Create a program for the robot	Standard 1: Recognize that many daily tasks can be described as step-by-step instructions (i.e., algorithms).
4. How would you debug the block of programming below so that the robot moves backward for three seconds at 100% power and then coasts?	Test and debug a program	Standard 4: Develop a program to express an idea or address a problem
5. Which of these movement blocks would you add to build a program which moves the robot forward until it encounters a black line, then it backs up?	Create a program for the robot	Standard 1: Recognize that many daily tasks can be described as step-by-step instructions (i.e., algorithms).

Table C.2. *Advanced Procedures Assessment Question, Lesson Objective, and SC State Computer Science Standard Alignment*

Assessment Question	Lesson Objective	Computer Science Standard
6. Where would a robot running this programming finish at the end of the program?	Predict the outcome of a program	Standard 1: Design, evaluate, and modify simple algorithms (e.g., steps to make a sandwich; steps to a popular dance; steps for sending an email).
7. Where would a robot running this programming finish at the end of the program?	Predict the outcome of a program	Standard 1: Design, evaluate, and modify simple algorithms (e.g., steps to make a sandwich; steps to a popular dance; steps for sending an email).
8. Finish the program with the arranged segments to perform the action on the diagram.	Create a program to solve a problem	Standard 3: Decompose problems into subproblems and write code to solve the subproblems (i.e., break down a problem into smaller parts).
9. Identify the program designed to perform the action in the diagram.	Predict the outcome of a program	Standard 1: Design, evaluate, and modify simple algorithms (e.g., steps to make a sandwich; steps to a popular dance; steps for sending an email).
10. Your friend writes a program to move a car in a backward C shape, but the program keeps moving in an S shape. Identify which segment is incorrectly programmed.	Modify a simple program	Standard 3: Decompose problems into subproblems and write code to solve the subproblems (i.e., break down a problem into smaller parts).

Table C.3. *Control Structures Assessment Question, Lesson Objective, and SC State Computer Science Standard Alignment*

Assessment Question	Lesson Objective	Computer Science Standard
11. Which of the following loop sequences will say a different word after ever four turns?	Predict the outcome of an algorithm that uses control structures	Standard 2: Use and compare simple coding control structures (e.g., if-then, loops).
12. Which loop option simplifies this program?	Modify a simple algorithm using control structures	Standard 2: Use and compare simple coding control structures (e.g., if-then, loops).
13. Given the conditional if/then statement, what will happen if the robot detects black?	Predict the outcome of an algorithm that uses control structures	Standard 2: Use and compare simple coding control structures (e.g., if-then, loops).
14. How many times will the following program say "hello" before ending?	Predict the outcome of an algorithm that uses control structures	Standard 2: Use and compare simple coding control structures (e.g., if-then, loops).
15. Finish creating an algorithm so that the car moves in the pattern on the ground as demonstrated in the graphic on the right.	Create an algorithm using control structures	Standard 2: Use and compare simple coding control structures (e.g., if-then, loops).

Table C.4. *Variables Assessment Question, Lesson Objective, and SC State Computer Science Standard Alignment*

Assessment Question	Lesson Objective	Computer Science Standard
16. Which algorithm counts each black line it encounters forever?	Predict the outcome of an algorithm that uses control structures	Standard 2: Use and compare simple coding control structures (e.g., if-then, loops).
17. Given the pictured conditional if/then statement, what will happen each time the robot detects a black line?	Modify a simple algorithm using control structures	Standard 2: Use and compare simple coding control structures (e.g., if-then, loops).
18. Your friend is building an algorithm which will increment a variable by one and turn left at each green line encountered. Choose the string of programming in which the variable increases by one at each green line encountered and displays the updated count textually on the EV3's display to complete this algorithm.	Predict the outcome of an algorithm that uses control structures	Standard 2: Use and compare simple coding control structures (e.g., if-then, loops).
19. Given the variable, what will this program do?	Predict the outcome of an algorithm that uses control structures	Standard 2: Use and compare simple coding control structures (e.g., if-then, loops).
20. Create a program with a variable value of 25 which will subtract 15 power from the motor for each line it encounters.	Create an algorithm using control structures	Standard 2: Use and compare simple coding control structures (e.g., if-then, loops).

APPENDIX D

EXPERT REVIEWERS’ VALIDATION FEEDBACK

Reviewer 1

I already stole the whole thing. It progresses in difficulty quickly with Week 3 and 4 being pretty brutal. Teachers with foundational experience should be able to figure this out. I will probably use these in group work for students – assigning each question to a group and having students actually program each answer and run the bots to observe, then report the results.

This is a great example of a test with variety – construct/deconstruct, code/debug, matching, etc.

What kind of [redacted] helped write the K-8 CS standards?

For specifics goes:

- Overall the questions are succinct and unambiguous.
- My current students and even the [school redacted] CS kids might be confused by the diagram on question #8.
- At first glance it appears that the bot goes up, like straight up. Students should figure it out when analyzing the answers. For some reason, the diagrams for #9 and #10 are more clear to me – go left or right not up. The diagram for #15 is iffy and I can see students trying to jump the bot.

- The Lego screen capture for #11 is too small for this old guy. You may have to break them up or put them landscape on a separate page. I have this problem often – kids taking a test and complaining that they can't see what's in the boxes.
- For #18 "friend is building an algorithm which will add by one and turn left . . ." I would suggest "increment by one" or "increment a variable by one" For some reason, "add by one" doesn't jive with me.
- #20 is a [redacted] and I'm not sure I can figure it out. I like the idea of the answers corresponding to blank boxes in the code. I'm gonna steal that idea too.

Reviewer 2

My sincere apologies for not replying earlier. It's been a very difficult semester for me. Overall I really like this. I've been teaching an engineering course using the EV3s for the past year (that also uses our mazes). These questions would have been very helpful for my assessments.

I've attached my version of the assessment key that includes my comments. Please double-check my work, I'm a bit exhausted this afternoon. I'll also be around for the next two weeks if you want to follow up with me on my comments.

A couple of last comments. When I started up my version of the EV3-G software I got the attached message. It appears the LEGO Education is making a move to a newer version of their programming language. Also for the last semester I've been working with Python version of the software,

https://education.lego.com/en-us/support/mindstorms-ev3/python-for-ev3

It’s not bad. Just something to think about before going all in with the lesson plans and assessment instruments you’re developing.

- #1: I think the answer should be 3 rotations not 6 rotations. If 1 meter = 2160□ then half a meter would be 1080°. If I then take 1080° and divide by 360° I get 3. So I think item (a) is the correct answer.
- #9: The diagram implies that the robot performs a point turn at the junction (a pivot turn would also work). This occurs when the steering parameter is set to +/-100 or +/- 50 for a pivot turn. Most of the options include curve turns which will cause the robot to move forward in an arcing path. I don’t think any of the options are correct. Option (B) doesn’t work because it’s missing the final move forward segment.
- #17: Up to a maximum of 100. So after 10 lines it won’t continue making the robot move faster. The variable value will still increase, but the actual speed value in the final green block will max out at 100.
- #18: Not that affects the answer, but there is an extra floating bubble that says “Port: 3” on the image. It may be confusing.
- #19: The first Move Steering block is set to a power of 25 not 50.

APPENDIX E

PROGRAMMING MOTIVATION SURVEY

Demographic Information Please select the choice which best describes you.
Age: *0 – 100*
Gender: *Male – Female*
Classification: *Freshman – Sophomore – Junior – Senior*
Education major concentration: *Early Childhood – Elementary – Middle Level – Special Education – Physical Education*
I would rate my technology comfort level as: *Low – Medium – High*
I have prior programming experience. *Yes – No*
I have had prior programming instruction. *Yes – No*
I have prior experience programming a robot. *Yes – No*
I have had prior robotics instruction. *Yes – No*
Programming Motivation Please indicate your level of agreement with each of the following statements: *1* (strongly disagree), *2* (disagree), *3* (neutral), *4* (agree), *5* (strongly agree)
1. Programming is relevant to my life. *1 – 2 – 3 – 4 – 5*
2. Teaching programming would benefit my students. *1 – 2 – 3 – 4 – 5*
3. Learning programming is interesting. *1 – 2 – 3 – 4 – 5*
4. I am confident in learning programming *1 – 2 – 3 – 4 – 5*
5. I put enough effort into learning programming. *1 – 2 – 3 – 4 – 5*
6. I use various strategies to learn programming well. *1 – 2 – 3 – 4 – 5*
7. Learning programming will help me get a good job. *1 – 2 – 3 – 4 – 5*
8. Programming activities will enhance my students' learning *1 – 2 – 3 – 4 – 5*
9. I am confident I will do well on programming tests. *1 – 2 – 3 – 4 – 5*
10. Knowing programming will give me a career advantage. *1 – 2 – 3 – 4 – 5*

11. I spend a lot of time learning programming. *1 – 2 – 3 – 4 – 5*
12. Learning programming makes my life more meaningful. *1 – 2 – 3 – 4 – 5*
13. Understanding programming will benefit me in my career. *1 – 2 – 3 – 4 – 5*
14. I am confident I will do well on programming activities. *1 – 2 – 3 – 4 – 5*
15. I believe I can master programming knowledge and skills. *1 – 2 – 3 – 4 – 5*
16. I concentrate fully on what I do when I work on programming activities. *1 – 2 – 3 – 4 – 5*
17. I am curious about advancing my programming skills. *1 – 2 – 3 – 4 – 5*
18. I plan to incorporate programming into my teaching. *1 – 2 – 3 – 4 – 5*
19. I enjoy learning programming. *1 – 2 – 3 – 4 – 5*
20. I look for additional resources to improve my skills when learning programming. *1 – 2 – 3 – 4 – 5*
21. I enjoy teaching programming to others *1 – 2 – 3 – 4 – 5*
22. I can teach programming in my future courses *1 – 2 – 3 – 4 – 5*
23. My career will involve programming. *1 – 2 – 3 – 4 – 5*
24. I can write advanced programs *1 – 2 – 3 – 4 – 5*
25. I will use programming problem-solving skills in my career. *1 – 2 – 3 – 4 – 5*

Figure E.1. The Programming Motivation Survey adapted from the Science Motivation Questionnaire II © 2011 Shawn M. Glynn.

APPENDIX F

ADAPTATION OF SMQ-II

SMQ-II	**Programming Motivation Survey**
Intrinsic Motivation	
Learning science is interesting.	Learning programming is interesting.
I am curious about discoveries in science.	I am curious about advancing my programming skills.*
The science I learn is relevant to my life.	Programming is relevant to my life.
Learning science makes my life more meaningful.	Learning programming makes my life more meaningful.
I enjoy learning science.	I enjoy learning programming.
Career Motivation	
Learning science will help me get a good job.	Learning programming will help me get a good job.
Understanding science will benefit me in my career.	Understanding programming will benefit me in my career.
Knowing science will give me a career advantage.	Knowing programming will give me a career advantage.
I will use science problem-solving skills in my career.	I will use programming problem-solving skills in my career.
My career will involve science.	My career will involve programming.
Self-Determination	
I study hard to learn science.	I concentrate fully on what I do when I work on programming activities.*
I prepare well for science tests and labs.	I look for additional resources to improve my skills when learning programming.*
I put enough effort into learning science.	I put enough effort into learning programming.
I spend a lot of time learning science.	I spend a lot of time learning programming.
I use strategies to learn science well	I use various strategies to learn programming well.
Self-Efficacy	
I believe I can earn a grade of "A" in science.	I am confident in learning programming.*
I am confident I will do well on science tests.	I am confident I will do well on programming tests.
I believe I can master science knowledge and skills.	I believe I can master programming knowledge and skills.
I am sure I can understand science	I can write advanced programs.*
I am confident I will do well on science labs and projects.	I am confident I will do well on programming activities.
Grade Motivation	*Motivation to Integrate Programming into Teaching**
Scoring high on science tests and labs matters to me.	I plan to incorporate programming into my teaching.*
It is important that I get an "A" in science.	I can teach programming in my future courses. *
I think about the grade I will get in science.	I enjoy teaching programming to others. *
Getting a good science grade is important to me.	Programming activities will enhance my students' learning. *
I like to do better than other students on science tests.	Teaching programming would benefit my students. *

Figure F.1. The adaptations of the Programming Motivation Survey statements and subscales from the Science Motivation Questionnaire II © 2011 Shawn M. Glynn.
Note. * Indicates replacement.

APPENDIX G

INDIVIDUAL INTERVIEW PROTOCOL

Introduction

Hello (interview participant),

Thank you for taking time to sit down with me today. As you know, my name is Mr. Fegely. I am a doctoral candidate in the Education Department at the University of South Carolina. I am conducting a research study as part of the requirements of my degree in Curriculum and Instruction - Educational Technology, and I would like to invite you to participate.

I am studying programming comprehension and motivation among preservice teachers. If you decide to participate, you will participate in an individual interview about programming motivation. In particular, we will discuss your experiences with the programming activities performed in class over the past few weeks. You do not have to answer any questions that you do not wish to answer. The interview will take place at in this classroom and should last about 30 minutes. The session will be audio and video recorded so that I can accurately transcribe what is discussed. The footage will only be reviewed by members of the research team and destroyed upon completion of the study.

Participation is confidential. Study information will be kept in a secure location. The results of the study may be published or presented at professional meetings, but your identity will not be revealed. Remember, participation, non-participation or withdrawal will not affect your grades in any way.

I will be happy to answer any questions you have about the study now or later by phone or email. You may contact me at extension [redacted] or [redacted].

Thank you for your consideration. If you would like to participate, I will begin with the instructions for how this interview will operate.

I have prepared questions about your experiences with programming. Please answer them openly and honestly with substantial depth. Remember, there are no wrong answers. Please feel free to present your perspective even if you do not believe it is shared by myself or others. I have twelve main questions for you. Once I present the question, feel free to share your perspective and experiences. As the interviewer, I may interject to ask qualifying questions, but mainly I will be listening to your responses. Let us begin now.

Questions

1. What aspects, if anything, interested you in the programming activities?

- Prompt: Can you explain what you found interesting about those programming activities?

2. Tell me about your experiences with the programming activities in the course.

- Prompt: Which one(s) was(were) most enjoyable? Explain.
- Prompt: Which one(s) was(were) least enjoyable? Explain.

3. How do you think learning programming will influence your career after graduation?

- Prompt: Can you give me an example of how you feel learning programming will influence your career after graduation?

4. In what ways do you believe learning programming would be valuable to you as a teacher?

- Prompt: How has your opinion changed since the beginning of this course?

5. Can you tell me about a time when you felt learning programming was hard?

- Prompt: Why did you feel this way?
- Prompt: How did you overcome that situation?

6. Tell me about a time you put in extra effort over the past four weeks to research additional resources to help you during the programming activities.

- Prompt: How did you make the decision to seek additional resources?

7. Tell me about your current state of programming knowledge and skills?

- Prompt: How do you think it has changed since the beginning of this course?

8. What are your feelings on learning even more advanced programming?

9. Where do you position yourself in the continuum of adding or not adding programming activities to your classes? Why?

10. Tell me about your thoughts on how programming activities would fit into the grade level and subject area you will teach?

- Prompt: Can you please give me an example programming activity for the grade or subject area you will be teaching.

11. Which programming activities do you feel were effective in helping you learn programming?

- Prompt: What suggestions would you make to improve the programming activities in this course?

12. Do you have any questions for me?

That concludes our interview. I will share a copy of the transcript of this interview with you via email in the coming days. Please let me know if there is anything in the transcript which you feel does not properly communicate what you were trying to say. Remember, you can opt out at any time. Thank you for the time and effort you have put into answering these questions.

APPENDIX H

UNIVERSITY IRB APPROVAL

UNIVERSITY OF SOUTH CAROLINA

OFFICE OF RESEARCH COMPLIANCE

INSTITUTIONAL REVIEW BOARD FOR HUMAN RESEARCH

APPROVAL LETTER for EXEMPT REVIEW

Alex Fegely

[Redacted]

Myrtle Beach, SC 29579 USA

Re: **Pro00095457**

Dear Mr. Alex Fegely:

This is to certify that the research study ***Learning Programming Through Robots: A Mixed-Methods Study on the Effects of Educational Robotics on Programming Comprehension and Motivation of Preservice Teachers*** was reviewed in accordance with 45 CFR 46.104(d)(1), the study received an exemption from Human Research Subject Regulations on **12/18/2019**. No further action or Institutional Review Board (IRB) oversight is required, as long as the study remains the same. However, the Principal Investigator must inform the Office of Research Compliance of any changes in procedures involving human subjects. Changes to the current research study could result in a reclassification of the study and further review by the IRB.

Because this study was determined to be exempt from further IRB oversight, consent document(s), if applicable, are not stamped with an expiration date.

All research related records are to be retained for at least three (3) years after termination of the study.

The Office of Research Compliance is an administrative office that supports the University of South Carolina Institutional Review Board (USC IRB). If you have questions, contact Lisa Johnson at lisaj@mailbox.sc.edu or (803) 777-6670.

Sincerely,

[illegible]

Lisa M. Johnson
ORC Assistant Director and IRB Manager

APPENDIX I

RESEARCH SITE IRB APPROVAL

November 20, 2019

Alex Fegely

Education

[redacted]

[redacted]

RE: Learning Programming Through Robots

Alex,

It has been determined that your protocol #2020.97 is approved as **EXEMPT** by the [redacted] University Institutional Review Board (IRB) under the Federal Policy for the Protection of Human Research Subjects categories #1 & 2.

This approval is good for one calendar year commencing with the date of approval and concludes on **11/19/2020**). If your work continues beyond this date it will be necessary seek a continuation from the IRB. If your work is concluded before this date, please so inform the IRB.

Approval of this protocol does not provide permission or consent for faculty, staff or students to use university communication channels for contacting or obtaining information from research subjects or participants. Faculty, staff and students are responsible for obtaining appropriate permission to use university communications to contact research participants. For use of university e-mail to groups such as all faculty/staff, all students or other large groups

on campus permission must be first obtained by the researcher from the Office of the Provost after the research protocol has been approved by the IRB. Please allow at least one week to receive approval.

Please note, it is the responsibility of the Principal Investigator to report immediately to the IRB any changes in procedures involving human subjects and any unexpected risks to human subjects, any detrimental effects to the rights or welfare of any human subjects participating in the project, giving names of persons, dates of occurrences, details of harmful effects, and any remedial actions. Such changes may affect the status of your approved research.

Be advised that study materials and documentation, including signed informed consent forms, must be retained for at least three (3) years after termination of the research and shall be accessible for purposes of audit.

If you have any questions concerning this Review, please contact [redacted], IRB Coordinator, at [redacted] or extension 2978.

Thank you,

[illegible]

[redacted]

Director, Office of Sponsored Programs and Research Services

IRB Administrator

APPENDIX J

CONSENT FORMS

Informed Consent for Human Subject Research Participation

Introduction
My name is Alex Fegely and I am a faculty member [redacted]. I would like to invite you to take part in my research study entitled, *Learning Programming Through Robots*. You are free to talk with someone you trust about your participation in this research and may take time to reflect on whether you wish to participate or not. If you have any questions, I will answer them now or at any time during the study.

Purpose
The purpose of this research study is to evaluate the effects educational robotics have on programming comprehension and motivation of preservice teachers.

Procedures
During this research study, you will take motivation comprehension assessments, programming motivation surveys, and possibly be asked questions as part of an individual interview.

Duration
For this research study, your participation will be required for 5 weeks of in-class time.

Rights
You do not have to agree to participate in this research study. If you do choose to participate, you may choose not to at any time once the study begins. There is no penalty for not participating or withdrawing from the study at any time. If you are a [redacted] student, your decision to participate or not will have no affect your grade.

Risks
During this research study, no risks or discomforts are anticipated.

Benefits
By agreeing to participate in this research study you may help a better understanding of programming and its applications with educational robotics.

Confidentiality
Unless you provide consent to the contrary, the confidentiality of your participation in this research study, your responses or any individual results will be maintained by the PI and all members of the research team.

Note that confidentiality will only be violated when required by law or the ethical guidelines of the American Psychological Association. This usually includes, but may not be limited to, situations when your responses indicate that you, or another clearly identified individual, is at risk of imminent harm or situations in which faculty are mandated reporters, such as instances of child abuse or issues covered under Title IX regulations. For more information about Title IX, please see the University's webpage at: [redacted].

Sharing the Results
As the Principal Investigator on this research study, I plan to share the results of this study with my dissertation committee and by publishing in peer-reviewed journals and presenting at academic conferences. None of the material published or presented will have any identifying information.

Contacts
If you have any questions about this research study, please feel free to contact me by phone [redacted] or [redacted].

The Institutional Review Board (IRB) under the Office of Sponsored Programs and Research Services is responsible for the oversight of all human subject research conducted at [redacted]**. If you have any questions about your rights as a research participant before, during or after the research study, you may contact this office by calling** [redacted]**or emailing OSPRS@**[redacted]**.edu.**

****

Consent

I have read this form and have been able to ask questions of the PI and/or discuss my participation with someone I trust. I understand that I can ask additional questions at any time during this research study and am free to withdraw from participation at any time.

☐ **I agree to take part in this research study.**

☐ **I agree to allow my name or other identifying information to be included in reports, publications and/or presentations resulting from this research study.**

☐ **I DO NOT agree to allow my name or other identifying information to be included in reports, publications and/or presentations resulting from this research study.**

Participant's signature: ____

Date: ____

Photography, Video or Audio Recording Authorization

I hereby release, discharge and agree to save harmless [redacted], its successors, assigns, officers, employees or agents, any person(s) or corporation(s) for whom it might be acting, and any firm publishing and/or distributing any photograph, video footage or audio recording produced as part of this research, in whole or in part, as a finished product, from and against any liability as a result of any distortion, blurring, alteration, visual or auditory illusion, or use in composite form, either intentionally or otherwise, that may occur or be produced in the recording, processing, reproduction, publication or distribution of any photograph, videotape, audiotape or interview, even should the same subject me or my to ridicule, scandal, reproach, scorn or indignity. I hereby agree that the photographs, video footage and audio recordings may be used under the conditions stated herein without blurring my identifying characteristics.

If you have any questions about this research study, please feel free to contact me by phone [redacted] or [redacted].

The Institutional Review Board (IRB) under the Office of Sponsored Programs and Research Services is responsible for the oversight of all human subject research

conducted at [redacted]**. If you have any questions about your rights as a research participant before, during or after the research study, you may contact this office by calling** [redacted] **or emailing OSPRS@**[redacted]**.edu.**

I have read this authorization and have been able to ask questions of the PI and/or discuss my participation with someone I trust. I understand that I can ask additional questions at any time during this research study and am free to withdraw from participation at any time.

Participant’s signature: ____

Date: ____